



NOVA
NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTAMENTO DE ENGENHARIA
ELETROTÉCNICA E DE COMPUTADORES

OCTÁVIO MIGUEL SILVEIRA TEIXEIRA

Mestre em Engenharia Eletrotécnica e de Computadores

NETWORK INFRASTRUCTURES IN THE DARK WEB

MESTRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Universidade NOVA de Lisboa
Novembro, 2021



NETWORK INFRASTRUCTURES IN THE DARK WEB

OCTÁVIO MIGUEL SILVEIRA TEIXEIRA

Mestre em Engenharia Eletrotécnica e de Computadores

Orientador: Doutor Robert Kooij,
Professor Catedrático, Universidade Técnica de Delft

Coorientador: Doutor Luís Bernardo
Professor Associado, Universidade NOVA de Lisboa

Júri:

Presidente: Doutor Nuno Filipe Silva Veríssimo Paulino,
Professor Associado, FCT-NOVA

Arguente: Doutor Miguel Pupo Correia,
Professor Catedrático, IST

Orientador: Doutor Robert Kooij,
Professor Catedrático, TU Delft

Network Infrastructures in the Dark Web

Copyright © Octávio Miguel Silveira Teixeira, Faculdade de Ciências e Tecnologia, Universidade NOVA de Lisboa.

A Faculdade de Ciências e Tecnologia e a Universidade NOVA de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

ABSTRACT

With the appearance of the Internet, open to everyone in 1991, criminals saw a big opportunity in moving their organisations to the World Wide Web, taking advantage of these infrastructures as it allowed higher mobility and scalability. Later on, in the year 2000, the first system appeared, creating what is known today as the Dark Web. This layer of the World Wide Web became quickly the option to go when criminals wanted to sell and deliver content such as match-fixing, children pornography, drugs market, guns market, etc. This obscure side of the Dark Web, makes it a relevant topic to study in order to tackle this huge network and help to identify these malicious activities and actors.

In this master thesis, it is shown through the study of two datasets from the Dark Web, that we are surrounded by capable technologies that can be applied to these types of problems in order to increase our knowledge about the data and reveal interesting characteristics in an interactive and useful way. One dataset has 10 000 relations from domains living in the Dark Web, and the other dataset has thousands of data from just 11 specific domains from the Dark Web.

We reveal detailed information about each dataset by applying different analysis and data mining algorithms. For the first dataset we studied domains availability patterns with temporal analysis, we categorised domains with machine learning neural networks and we reveal the network topology and nodes relevance with social networks analysis and core-periphery model. Regarding the second dataset, we created a cross matching information web graph and applied a name entity recognition algorithm which ended in a tool for identifying entities within dark web's domains. All of these approaches culminated in an interactive web application where we publicly not only display the entire research but also the tools developed along with the project (<https://darkor.org>).

Keywords: Dark Web, Social Network Analysis, Core-Periphery Model, Machine Learning, Dark Web Monitoring, Natural Language Processing

RESUMO

Com o surgimento da Internet, aberta a todos em 1991, os criminosos viram uma grande oportunidade em passar as suas organizações para a *World Wide Web*, aproveitando-se assim dessas infraestruturas que permitiam uma maior mobilidade e escalabilidade. Mais tarde, no ano 2000, surgiu o primeiro sistema, criando o que hoje é conhecido como a Dark Web. Essa camada da *World Wide Web* tornou-se rapidamente a opção a seguir quando os criminosos queriam vender e entregar conteúdo como combinação de resultados, pornografia infantil, mercado de drogas, mercado de armas, etc. Este lado obscuro da Dark Web, torna-a num tema relevante de estudo a fim de ajudar a identificar atividades e atores maliciosos.

Nesta dissertação de mestrado é mostrado, através do estudo de dois conjuntos de dados da Dark Web, que estamos rodeados de tecnologias que podem ser aplicadas neste tipo de problemas de forma a aumentar o nosso conhecimento sobre os dados e revelar características interessantes de forma interativa e útil. Um conjunto de dados tem 10 000 relações de domínios que vivem na Dark Web enquanto que o outro conjunto de dados tem milhares de dados de apenas 11 domínios específicos da Dark Web.

Neste estudo revelamos informações detalhadas sobre cada conjunto de dados aplicando diferentes análises e algoritmos de *data mining*. Para o primeiro conjunto de dados, estudamos padrões de disponibilidade de domínios com análise temporal, categorizamos domínios com o auxílio de redes neuronais e revelamos a topologia da rede e a relevância dos nós com análise de redes sociais e a aplicação de um modelo núcleo-periferia. Em relação ao segundo conjunto de dados, criamos um grafo da rede com cruzamento de dados e aplicamos um algoritmo de reconhecimento de entidades que resultou em uma ferramenta para identificar entidades dentro dos domínios da Dark Web estudados. Todas estas abordagens culminaram em uma aplicação web interativa onde exibimos publicamente não apenas todo o estudo, mas também as ferramentas desenvolvidas ao longo do projeto (<https://darkor.org>).

Palavras-chave: Dark Web, Análise de Redes Sociais, Modelo Núcleo-Periferia, Aprendizado Máquina, Monitoramento da Dark Web, Processamento de Linguagem Natural

CONTENTS

List of Figures	xi
List of Tables	xv
1 Introduction	1
1.1 Crime on the Internet	1
1.1.1 The Silk Road Market	2
1.1.2 Operation Torpedo	4
1.2 Problem Statement	5
1.3 Research Structure	8
2 Dark Web	11
2.1 World Wide Web Structure	11
2.2 Dark Web	13
2.3 Dark Web Softwares	14
2.3.1 Freenet	15
2.3.2 Tor Project	18
2.4 Dark Web Models Approximations	31
2.4.1 Social Network Analysis	32
2.4.2 Core-Periphery Structures	39
2.4.3 Machine Learning	44
3 State of the Art	49
3.1 Using Unsupervised Learning Algorithms on the Dark Web Forums	49
3.1.1 Process	50
3.1.2 Final Results	53
3.2 Predicting cyberattacks within Dark Web forums	54
3.2.1 Process	54
3.2.2 Final Results	57
3.3 Building and Analyzing Tor web graph	60
3.3.1 Process	61
3.3.2 Final Results	63
3.4 Overview	67

CONTENTS

4 Dataset 1	71
4.1 Data Collection	71
4.2 Data Analysis	72
4.2.1 MySQL Data Structure	73
4.2.2 Scraping Domains	74
4.2.3 Optical Character Recognition (OCR)	76
4.2.4 Temporal Analysis	78
4.2.5 Automatic Classifier	81
4.2.6 Social Network Analysis	93
4.2.7 Core-Periphery Analysis	94
4.2.8 Domains Detection	96
5 Dataset 2	99
5.1 Data Collection	99
5.2 Data Analysis	101
5.2.1 Cross-Matching Web Graph	102
5.2.2 Name Entity Recognition (NER)	105
6 Web Application	109
6.1 Technology	109
6.2 Development	110
6.3 Results	111
7 Conclusion	121
Bibliography	125

LIST OF FIGURES

2.1	Difference between Clear Web, the Deep Web and the Dark Web on an analogy with an iceberg. Retrieved from [53]	11
2.2	Simple representation of the tunnelling technique used on VPNs where the web server will see incoming packet from someone with IP address 82.154.185.219 making the client's IP address protected	14
2.3	Request example that shows system recovering from dead-end at step 3 and from a loop at step 7. Retrieved from [18].	16
2.4	Example of an onion routing circuit using 3 onion routers each one with its respective public key.	19
2.5	Onion routing control cell structure and relay cell structure, respectively. Retrieved from [6].	20
2.6	Example of an onion routing after establishing the circuit to the web server.	22
2.7	Message encrypted at a client showing multiple layers of encryption. Retrieved from [41]	23
2.8	Example of a connection within the Tor network using hidden services	26
2.9	Example of how Shallot would behave when asked to generate an onion address with the word "test" on it. Retrieved from [32]	27
2.10	Hidden Service architecture example	28
2.11	Comparison between a random network (A), where every node is dispersed in space, and a scale-free network (B) where some nodes have more connections to the rest of the network. Retrieved from [16]	33
2.12	Power Law Distribution Representation	34
2.13	Example of a network graph with 7 nodes	34
2.14	Example of a small world network. Retrieved from [51]	38
2.15	Example of a network with a core-periphery structure. Retrieved from [12].	39
2.16	Comparison between a single core-periphery structure (a) and (c) with a multiple core-periphery structures (b) and (d), respectively. Retrieved from [34].	42
2.17	Core-periphery structure of the karate club network. On (a) it was used the Borgatti-Everett algorithm, (b) the two-step algorithm and in (c) the authors of [34] algorithm.	43
2.18	Neural Networks Neuron Model. Retrieved from [59].	45
2.19	Neural Network Diagram with 3 layers. Retrieved from [15].	46

LIST OF FIGURES

3.1	Network displaying the five categories created by the twelve metrics used in the study [45]. Retrieved from [45]	52
3.2	Network created from forum Ansar AlJihad Network where each node size shows its HITS hub score. Retrieved from [45].	53
3.3	Merge operation of the network with important nodes highlighted in grey. Retrieved from [50].	56
3.4	Results obtained from the learning algorithm in 1 week time window. Retrieved from [50].	58
3.5	Results obtained from predicting malicious email attacks within the data range with more cyberattacks. Retrieved from [50].	59
3.6	Graphs showing degree distribution scores for each type of network graph. Retrieved from [9].	63
3.7	Graphs showing betweenness centrality scores for each type of network graph. Retrieved from [9].	64
3.8	Graphs showing the betweenness centrality (a) and degree (b) score upon removal of important nodes on Page Graph undirected network. Retrieved from [9].	64
3.9	Statistical distribution of the cosine similarity in Page Grap. Retrieved from [9].	65
3.10	Statistical distribution of the cosine similarity for every single category studied. Retrieved from [9].	66
4.1	Web Crawler Diagram where the crawler represents the software, the seeds represent the starting domains used to access the web servers which the “bug” will crawl to find hyperlinks and discover new domains.	72
4.2	Excel file provided with the dataset 1.	72
4.3	Entities relation diagram of the first four tables created on the MySQL database.	73
4.4	Portion of the hyperlinks table with the results from the scraping process.	75
4.5	On this image, we can see the first hyperlink of the dataset with little and non-relevant text on the “extra” field, which is the one with the text removed from its index web page source code	75
4.6	MySQL table created to save the texts extracted from the images.	77
4.7	Entries from the hyperlink_images table for the URL id = 1 with 6 images where the OCR was applied.	77
4.8	One of the entries (row id = 21) in detail where we can confirm that the text extracted from the image clearly passes the message of this website purpose	78
4.9	Table created to store the measures from the temporal analysis script.	79
4.10	Measures between March 23rd and April 29th showing some variability in the availability of the website with ID 2420.	80

4.11 Availability web graph from the dataset 1 with measures taken from 21-03-2021 till 24-03-2021.	81
4.12 Small portion from the DUTA 10K dataset	82
4.13 DUTA 10K Main Class Distribution	83
4.14 DUTA 10K Sub Class Distribution	83
4.15 MySQL tables created to prepare the DUTA 10K dataset for the training	84
4.16 Portion of the hyperlinks table after running the scraper	85
4.17 Portion of the hyperlinks_training table after running the scraper which shows the exact same structure as figure 4.16 apart from the language, main class and sub class fields	85
4.18 Index web page of a hyperlink found by seed with ID 178.	89
4.19 Top 20 words found on the text scraped from URL id 178	89
4.20 Automatic Classifier result on the left for the main class and on the right for the sub class	90
4.21 Top 20 words found on the text scraped from URL id 1	91
4.22 Automatic Classifier result for the main class	91
4.23 Automatic Classifier result for the sub class	92
4.24 Result of the single-core periphery structure found for the dataset 1. The left graph shows the entire network, the middle one shows the first 150 nodes and the right graph is the middle graph with an offset of 150 nodes on the x axis	96
4.25 MySQL extracted_hyperlinks table created	97
4.26 Portion of the extracted_hyperlinks table after searching for domains in the dataset	98
5.1 Figure 4.1 completed with the scraping process which was used for the dataset 2.	100
5.2 Data structure provided for each of the 11 domains scraped.	100
5.3 Crypto excel file example from the dataset 2.	101
5.4 Email excel file example from the dataset 2.	101
5.5 PGP excel file example from the dataset 2.	101
5.6 Portion of the structure of the domains	102
5.7 Portion of the structure of the emails	103
5.8 Portion of the domain emails relation structure	103
5.9 Cross-Matching Web Graph with the respective labels for each type of node. In yellow we have the 11 domains, in grey 8 PGP keys and 39 cryptocurrency wallets and in rose the 134 emails discovered.	104
5.10 On the left, we have the domains table that will contain the 11 domains from the dataset. In the middle, we have a domain_web_pages table which contains the information provided in the pages excel file. Finally, at the right, we have the table domain_ner, which will serve as an object to store the results from the NER for the respective domain.	107

LIST OF FIGURES

5.11	Portion of the results saved on the domain_ner table with the identification of the domain and web page to which it belongs to, the type of entity and the value.	108
6.1	Temporal Analysis result represented by the average availability of each node. Brighter green represents nodes that have an higher average availability and darker green the ones with lower average availability.	112
6.2	Main Class distribution from the automatic classifier applied on the dataset 1.	113
6.3	Sub Class distribution from the automatic classifier applied on the dataset 1.	113
6.4	Languages distribution showing a predominance of the English language. . .	114
6.5	Dataset 1 web graph with the node that owns the biggest part of the edges highlighted in white.	115
6.6	Dataset 1 web graph with the node that has the top measure for Closeness Centrality, Eigenvector Centrality, Page Rank and in-degree Centrality. . . .	115
6.7	Degree distribution from the dataset 1 network showing similarities with a power law distribution.	116
6.8	Core-periphery result of the single layer variant (discrete model). Red represent nodes belonging to the core while faded red nodes represent the periphery.	116
6.9	Core-periphery result of the multiple layer variant (continuous model). Red represent the nodes belonging to the first core-periphery pair and yellow represent the nodes belonging to the second core-periphery pair.	117
6.10	Hyperlinks destination found on the texts scraped from the domains analysed in the dataset 1.	118
6.11	Cross-Matching information web graph of the dataset 2.	118
6.12	Name Entity Recognition indexation tool.	120

LIST OF TABLES

2.1	Adjacency matrix resulting from the example in the figure 1.15 [12].	40
2.2	Ideal adjacency matrix showing core-periphery structures [12].	40
3.1	Table showing the metrics from the clusters identified on the forum Ansar AlJihad Network [45].	54
3.2	Table showing the groups of features and their description [50].	56
3.3	Table revealing the connected components after the cuts in figure 3.8 [9]. . .	65
3.4	Comparative table with the three research works explored	67
3.5	Comparison of the different techniques used from 10 different works [9]. . .	69

INTRODUCTION

This work is licensed under the Creative Commons Attribution-NonCommercial 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/4.0/>.

This section will introduce some relevant points important to understand before diving in-depth into the problem we're trying to deal with. It's essential to define some concepts and give a background under the roots of the project to help the reader follow the steps taken to achieve the final objective. As so, we will start with a summarised but detailed description of why this is such an important matter, following with important notions to carry in mind, and finishing with the statement of the problem.

1.1 Crime on the Internet

Cybercrime is a term that is, unfortunately, more and more often buzzing in our ears as it is such a serious topic related to networks, and that is also very sensitive as some forms of practices of these offences include taking human lives. The term "cybercrime", in simpler words, means the use of the Internet to elaborate and create criminal activities that can go from hacking other users and stealing their private information to creating criminal markets to sell drugs, guns, child pornography, organs, sensible videos as decapitations and even trafficking human beings...

So a question that can surge in our minds is who deals with these kinds of crimes and what cooperations exists nowadays to mitigate these condemnable acts?

Before answering it, it's essential to understand the origin of cybercrime. This term has been around since 1980, and the biggest problem of cybercrime is that it started to evolve at a dramatic pace and new forms of cybercrime appear year after year with little or non-existent information on how to deal with these crimes. The reason is that

they are supported by a platform that, apart from being hugely enormous and growing exponentially, allows users to use encrypted channels while being anonymous and at the same time working with this borderless and distributed environment that has a lack of regulations [43]. The fact that cybercrime evolved so quickly got law enforcement agencies unprepared, and even at this very moment, there is a lack of jurisdiction internationally because most of the time, these crimes are based across borders, and law enforcement units have no jurisdiction to prosecute such acts that by nature are hard to obtain evidence of [43].

In general big agencies like the Federal Bureau of Investigation (FBI), Drug Enforcement Administration (DEA), Secret Services, Europol, Interpol and law enforcement agencies from some countries such as the Dutch National Police have created departments to deal with these crimes and have done some operations that were completed as we will see in a while.

It is essential to understand that these crimes are the toughest ones for law enforcement agencies and judicial entities as it reveals to be hard to track these individuals and their infrastructures, and they will keep continuously developing hidden on the exponential growth of the Internet. If actions are not taken to mitigate these horrendous practices, this will turn into a race without a finishing line. Therefore, law enforcement agencies must keep up with the developments and the increasing rise of new criminal forms living under the Internet [43].

Another critical aspect to take into consideration when studying these criminal activities is how they exchange goods? The technology that has solved this issue for criminals is under the hood of cryptocurrencies by using a transparent anonymous, and immutable chain of records named blockchain. The appearance of these cryptocurrencies, almost like with the appearance of the Internet, had good intentions, create a distributed payment network that is not controlled by governments. This combination of hidden encrypted networks and a payment method that allows anonymity while secured transactions make the law enforcement agencies work even harder [36].

Through the years, there have been a few major operations to deal with some of these criminal activities living in the Dark Web. We will explore two operations that were publicly very well known, and that proves and shows in a practical form what have been said till this point.

1.1.1 The Silk Road Market

In 2011 an illegal market surged on the Dark Web with the name Silk Road, two years after the first and most famous cryptocurrency emerged, Bitcoin¹. This black market was more like an eBay platform where users could sign up as sellers or buyers and make

¹Bitcoin was the first cryptocurrency emerged in 2009 that was shown to the world with a paper from its creator, Satoshi Nakamoto, that can be seen at <https://bitcoin.org/bitcoin.pdf>

transactions anonymously, and just to have an idea of how big this was, it had a total sales volume of over 1.22 million dollars per month [17].

This was one of the most famous cases in the Dark Web that had originated multiple documentaries, researches and even movies about it as it's a complete story that reveals how hard it is to track websites that are hosted on the Dark Web, and how complicated it is to execute these complex operations that sometimes needs an opening from negligent actions by the owners of these criminal services [31].

The Silk Road website was the first one to combine the powerful synergy between the Dark Web and cryptocurrencies, opening the door to other illegal markets to follow their steps inside the Dark Web. Someone led this illegal market with the nickname "Dread Pirate Roberts", and this account from the Silk Road website was linked to a person named Ross Ulbricht that for what is known, created the platform. The Silk Road offered a forum where the community could send public messages to each other, and through analysis made on these posts, it was clear that the goal of the administrators was to create a market for drugs that could not be under the control of law enforcement units and that promoted freedom. The administrators also claimed that the website would not allow selling child pornography films or hitman jobs.

The operation behind this website was made through high levels of research and information gathering that gave some leads to whom could be the person behind the "Dread Pirate Roberts" account. In 2013 on a forum available on the public World Wide Web, law enforcement agencies found a post from a nickname "altoid" promoting the Silk Road website that was running on the Dark Web. Further investigations revealed that this nickname was linked to a Gmail account "rossulbricht@gmail.com". This discovery, together with a package seized by US Customs containing fake id documents such as driver's licenses and passports, led law enforcement finally to Ross Ulbricht's location. At his arrest, it was possible to neutralize Ulbricht's laptop and copy all of his files into what would later become digital pieces of evidence to prosecute him for being the head behind the Silk Road black market. At the time of arrest, the FBI had already access to the real server hosting the Silk Road website, but there was never a consensus on how the FBI discovered the real IP address from a website hidden on a Dark Web service. On the official declaration, they argue that due to a flaw in the CAPTCHA system, to prove to users that they're not robots, the FBI could retrieve IP packets containing the real IP address of the Silk Road server located in Iceland [31]. This investigation was way more complex than what was retracted here as this was maybe the first big operation in terms of Dark Web criminal groups, and it raised a lot of concerns about how can the law enforcements use their forces to find criminals as it is very suspicious that they found the real IP address through leakage from the CAPTCHA validation system. This story had a little bit of everything. Even a DEA agent named Carl Force was convicted of extortion and money laundering due to receiving payments from Ulbricht in exchange for inside information about the investigation. Nonetheless, the website was closed in 2013 and Ulbricht was sentenced to life imprisonment in 2015.

1.1.2 Operation Torpedo

Another interesting story is the “Operation Torpedo”, where the Dutch Police found a server in Nebraska in 2011 that was serving child pornography through the Dark Web under the name of Pedoboard. This operation had crucial help from a tool developed by the FBI named Network Investigative Technique (NIT)² that the federal government developed. This malware is basically a backdoor³, which gives access to someone’s personal files, location and even devices such as webcams. These capabilities shown from the FBI malware turned to become a use of force that could catch innocent people and criminals without distinction. That is one of the reasons for the appearance of hidden services that allow anonymity, as we will see further in this project.

So after the Dutch Police found the real IP address of this website, hosted in Nebraska, through a mistake from the owners of the web page as they left the administrative account without a password, the Dutch Police provided the information to the FBI as they were the ones with jurisdiction in the server’s local area. Further investigation from the FBI discovered that this server belonged to a web hosting company and that a 31 years old Aaron McGrath was the one owner of this website. It happens that Aaron McGrath was not only hosting one child pornography website on the Dark Web but two at the website hosting company where he worked and one more at his home. The FBI decided to surveil this man, and after a year, in 2012, in cooperation with the Justice Department, they decided to appear and take control of the servers that, with a warrant for each of the websites from a federal magistrate, were changed by the FBI to delivery the NIT malware to users that access it. This allowed the FBI to identify over 25 users of the website, and apart from this operation being a success, it raised concerns on when should and should not be used this malware software by the FBI as the success of this operation could lead the FBI to deploy this malware more often and failing the real objective that is identifying these criminals and not innocent people like journalists and activists that just want a right they own that is privacy[46].

There are other examples of successful operations that show attempts from law enforcement agencies to take down these criminal groups and individuals. However, we just wanted to give an idea of how complicated those are and how dependent they are on human error factors due to a lack of methods to apply on these networks as there is not much information about the Dark Web structure.

²More information about how it works can be found in <https://journals.sas.ac.uk/deeslr/article/view/2475>

³More information about backdoors can be found in <https://www.wired.com/2014/12/hacker-lexicon-backdoor/>

1.2 Problem Statement

This project was conceived at the Delft University of Technology under the Erasmus Traineeship Programme with Professor Robert Kooij, a part-time, full Professor at Network Architectures and Services group at Delft University. The project had also a contribution from Professor Luis Bernardo, who was the co-supervisor and the connecting element from NOVA University of Lisbon and Delft University of Technology.

The goal of this project is to extract relevant information about network structures within the Dark Web using two datasets provided by CFLW Cyber Strategies⁴. These two datasets consist of data crawled from the Tor network that allowed us to study real data and apply some theories that we explore in the next chapters.

The datasets are extremely important as they are crucial for the development of these kinds of projects. To elaborate, if the datasets from which we conduct a study are not relevant, do not have interesting patterns, or are incomplete and lacking information, it can compromise the study conclusions. Nevertheless, apart from being unpredictable, it should never be an excuse for the quality of the analysis and the processes used to study it.

- The first dataset consists of 10000 domains relations from the Dark Web indexed by the crawler that used some domains as “seed” to explore and find more and more onion links. By “seed”, we mean that these 10000 domains were extracted from some onion links that were picked beforehand, like wikis and link directories.
- The second dataset consists of 11 domains that were crawled more in detail, including a market named Tochka Market that started operating in early 2015, which is no longer online. Nevertheless, the information gathered from all of those 11 domains follows the same structure: a copy of thousands of web pages within every domain (in .html format), crypto information, emails information, web pages crawled list, Pretty Good Privacy (PGP)⁵ information and some information regarding the web server.

With these two datasets, we tried to extract relevant information from the Dark Web and apply the known models of the Dark Web while also trying new approaches such as combining them with some machine learning technologies. Nevertheless, before showing the work developed with this data, we show some of the recent works done on this area with similar data that helped us by enlightening us regarding state of the art techniques used previously.

⁴CFLW Cyber Strategies is a dutch company focused on finding and providing cyber security solutions. More information at <https://cflw.com/>

⁵The Pretty Good Privacy is an encryption system that allows two users to exchange information safely by using symmetric cryptography and asymmetric cryptography techniques.

Our goal, as stated before, is to use state of the art techniques and approaches to explore and retrieve valuable information from two datasets provided with data from the Dark Web.

To make a brief recall of what consists the datasets we have at our disposal two types of data extracted from the Dark Web:

1. Dataset with ten thousand hyperlinks relations created with a Dark Web crawler that has the following structure:
 - **FROM:** <http://kpvz7ki2v5agwt35.onion>
 - **TO:** <http://deepro33jjvfb3n7.onion>

So, we have 10000 structures, like the one above, where the “FROM” is from where the crawler navigated to get the hyperlink depicted in the “TO” field. Most of the hidden services from the “FROM” field are web pages such as wikis and link directories from the Dark Web.

Now, what can we do with this data? The possibilities are many but we focused on understanding the relationship between those domains. One aspect that could be relevant to understand within this dataset is: what if we could add more data to those hyperlinks? For that, we could build a simple scraper in python to navigate through every hyperlink, check if it is still active, and if so, gather information such as the title of the web page, description and some keywords. After that, with all of that information organised in a database, such as in a MySQL server, we can categorise hyperlinks, create a web graph and calculate some measures from social network analysis. With that information, we built an interactive interface on a web application using technologies such as JavaScript that will allow the visualisation of a web graph representing this dataset. We could also see it from a different perspective by adding filters to highlight nodes by category, the status of availability, etc.

Another interesting technique we also explored was the use of core-periphery structures to model the data and also represent it on the same web application as a comparison between approaches. As this study could have had multiple directions, we did not strict ourselves to what we described here, but those were some requirements we established as our goal for the study of this dataset.

2. The second dataset is composed of 11 structures, which are basically 11 hidden services from the first dataset crawled in detail. The structure for each hidden service has:
 - **Emails**

- **PGP Keys⁶**
- **Cryptocurrency Wallets Addresses**
- **Web Pages (in .html format)**

So basically, a scraper went to those 11 hidden services, one at a time, and extracted all the emails he could find there, all the PGP keys, cryptocurrency wallets and not only indexed all the web pages but also downloaded every single web page found within a hidden service into a folder.

The idea here is to correlate the data between domains to cross information between them and categorise them. We also looked for some interesting web pages within this dataset to explore what natural language processing algorithms could be interesting to analyse.

To summarise, there is a dataset with 10 000 relations from the Dark Web, which gives a more global perspective of the Dark Web, and a dataset with just 11 domains from the Dark Web that have a lot of specific information about those domains and all the web pages they have, so it gives a more local and closed perspective on the Dark Web. These datasets are great because they allowed us to have two different perspectives and give us the possibility to act according to the perspective we're working on. With the first dataset, we explored what interesting analyses could be used and are relevant to analyse big datasets of that nature, in the other hand with the dataset 2 we explored what technologies and analyses are interesting to study small numbers of domains but with a lot of content.

In essence, we were given the chance to study a dataset that gives a big perspective of the Dark Web, with a lot of domains and relations between them (dataset 1), and to study a small set of domains focusing more on the content of them and the relation between the content (dataset 2).

From this project, we could extract and work the raw data given by both datasets, and turn it into useful information revealing detailed information about the Dark Web domains associated, and, most importantly, displaying them on a useful and interactive form by implementing a web application that, more than showing the results, it can be also seen as an object for explaining the whole research while providing the tools developed and the processed data in an interactive way.

For the datasets in study we highlight the following contributions:

- The majority of the domains from the datasets were about pornography which reveals one of the major types of content in the Dark Web, which most of the time relates to child pornography.

⁶The Pretty Good Privacy is an encryption system that allows two users to exchange information safely by using symmetric cryptography and asymmetric cryptography techniques.

- The implementation of the web application revealed to be a strong tool to display results and share the research publicly while sharing tools that allow users to easily dive into the processed data.
- Images are an important resource for the Dark Web web which make them an important object of study for future works.
- The availability of these domains living within the Dark Web are prone to some instability which reveals to be an interesting characteristic to study.
- The network created with the data revealed a topology with low in-degree centrality and an edge distribution similar to a power-law distribution which is also common on these networks due to their nature.

1.3 Research Structure

This research will start by having two theoretical chapters followed by four chapters focused on the practical work developed.

The first two chapters are important to give a background about the Dark Web and explain how it works as well as explaining the latest researches done on this area for us to understand at which point is the study of the Dark Web.

The remaining four chapters of this project will handle the practical work developed during the duration of this project. We broke this chapter into two initial chapters where we explain the analyses and methods used to tackle the first and the second dataset, followed by a chapter where we explain the implementation of the web application as well as the results and tools displayed there. To finalise we end with the usual chapter with the conclusions pointing out the contributions of the research.

- 2 chapters with a theoretical study relevant for the research
 1. Chapter about the Dark Web technologies where we will explain in detail how the Dark Web is organised, how it works, examples of current darknets and some of the methods used to study these hidden networks.
 2. Chapter where we will explore state of the art researches to understand which technologies were used before to reveal information about these networks, and how they were implemented.
- 4 chapters about the practical part of the research
 1. Chapter focused on the study of the first dataset in which we will explain each method and approach taken to handle it. We explain the objective of the approaches, the technologies used, and the whole process from start to end including some of the first results and observations.

2. Chapter focused on the study of the second dataset in which we will explain each method and approach taken to handle it. We explain the objective of the approaches, the technologies used, and the whole process from start to end including some of the first results and observations.
3. Chapter that explains the implementation and the structure of the web application as well as the tools and results obtained from the study of both datasets.
4. Last chapter in which we point all the contributions of this project while making a brief recall of the initial objectives and the final results on which this project culminated.

DARK WEB

This chapter serves mainly to give a background about the technology in study, the Dark Web. In this chapter, we will explore the Dark Web structure and technology as well as explain the first darknet and the most used darknet at this moment. This makes this chapter extremely important as it reveals the main characteristics of these networks which are important to understand, as they make these networks special and singular within the World Wide Web.

2.1 World Wide Web Structure

The World Wide Web, a system that allows content exchange through the Internet, subdivides into three different parts. Those three parts are known as the Surface Web (usually mentioned as the Clear Web), the Deep Web and lastly, the Dark Web.

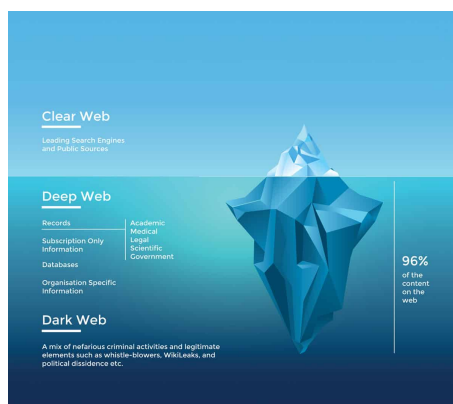


Figure 2.1: Difference between Clear Web, the Deep Web and the Dark Web on an analogy with an iceberg. Retrieved from [53]

The Surface Web, in analogy with an iceberg, is just the tip of the Internet. It is

composed of all web pages that are accessible to anyone without restraints. In other words, we can characterise the Surface Web as the part of the World Wide Web that can be indexed by search engines such as Google. Search engines use crawlers¹ that are programs that go through thousands of domains and explore every web page available on those domains to extract information and index information.

On the opposite of it, there is the Deep Web, the part hidden under the “sea” in the analogy with the iceberg. This analogy makes a perfect fit for this explanation in the eyes of search engines as the Deep Web is the part that these engines cannot crawl, and it is formed of the big portion of the World Wide Web. So any web page that is not crawled, either because credentials protect it or either the web server instructs it not to be, can be considered as the Deep Web. If a web page is protected by credentials such as our online banking account profile or our Facebook account feed is easy to understand why these crawlers cannot retrieve information from that private web page. However, there is another scenario that does not allow crawlers to visit certain web pages. This happens because the web server instructs them not to crawl certain pages by a file optionally added to the root of a web server named robots.txt. This is known as a Robots Exclusion Standard proposed by Martijn Koster in 1994 [35]. This consensus is not an official standard, but quickly everyone adopted this good practice as it saves crawlers time and helps search engines be more efficient. As a curiosity, there is also a non-official standard for adding not only a robots.txt file but also a humans.txt file where developers can add the names and contacts of the people that helped to build the web page in the form of response to the robots file saying that humans are also important and deserve a file designated for them.

The Dark Web, just like the Deep Web, cannot be crawled despite being publicly available to anyone, but in this case, it is just because it works on a different part of the Internet, and so it cannot operate under a usual browser. Therefore we can characterize the Dark Web as a portion of the Internet, more specifically a portion of the Deep Web, that to be accessed needs special software to handle requests. The difference in these overlay networks that together form the Dark Web, also known as Darknets², are different from the rest of the World Wide Web because they have their specific forms of routing, encrypting, handshaking and lastly, exchanging data providing anonymity and secured communications.

It is estimated that 96% of the web is actually composed of web pages that live “under the sea” in the Deep Web, this is really easy to understand as web applications nowadays serve purposes that are mostly hidden behind accounts, and those protected pages cannot be indexed as explain before [33]. On the other edge, almost reaching the bottom of the ocean, there is the Dark Web that can only be accessed through specific software as it

¹More information on how crawlers works can be found at <https://www.cloudflare.com/learning/bots/what-is-a-web-crawler/>

²Darknets are networks built-in top of the Internet that require specific software to be accessed. All existent Darknets together make the Dark Web.

does not support the paradigm of the Internet in terms of handshakes and encryption as it is a customized version that requires these specific programs capable of “speaking” according to the type of Dark Web architecture we’re trying to use. There are a few other types of software that live in the Dark Web ecosystem that we will explore further in this document. However, we will focus and specialize on the ones that are used the most, but before diving into those, we need to explore more about the Dark Web.

2.2 Dark Web

The Dark Web nowadays is seen mainly by its obscure side that besides being a small part of it, it reveals obscene, criminal activities that survive due to the characteristics of the Dark Web. This ease of deployment of criminal activities on these infrastructures turned the attention to the Dark Web in its negative sense. The Dark Web was built with the intuition to create a public platform that allows users to browse content privately, anonymously and securely without being worried about government surveillance or any other type of entities that may have the ability to track the activity on the Internet. This made this platform welcoming for activists and people in oppressed countries to share ideas and opinions without being publicly exposed.

Nonetheless, as with most of the inventions, users saw a big opportunity to make a profit from it and started to use this platform with the wrong principles. Actually, it is an interesting topic of criminology, understanding what makes people searching for any type of opening to start a criminal activity, either if it is just for making money either if it is just for pleasure, and for what is known there are not many pieces of research in understanding the similarities between the Dark Web criminal groups and the “offline” criminal groups. However, understanding some theories behind criminology can help us understand how to reduce the incentive for these groups to operate under the Dark Web ecosystem. There are explanations based on social disorganisation theories that propose a theory based on the facts of not having a centralised party managing and controlling the environment, the existence of instability and heterogeneity, leads people to trust in such infrastructures to deploy devious activities [39].

“Protecting political dissidents, privacy advocates, and whistle-blowers should not come at the expense of empowering child abusers, arms traffickers, and drug lords.” [36].

There are other solutions to provide anonymity and safe communications, like, for example, using a Virtual Private Network (VPN). It can be useful to allow communications between different protocols (i.e. communication between an IPv6 network and an IPv4 network) to allow users to hide their IP or work under some specific network that can be, for example, an enterprise network.

VPNs allow users to forward their traffic through a VPN server so that the web server that a user wants to visit can only see that the originator from a request is the VPN server and not the client that made the request. This works using a network technology named tunnelling that allows encapsulation of data inside other IP packets by repackaging the

original packet. The tunnelling technique can be split into three different protocols: Point-to-Point Tunneling Protocol (PPTP), Layer 2 Tunneling Protocol (L2TP) and IPsec.

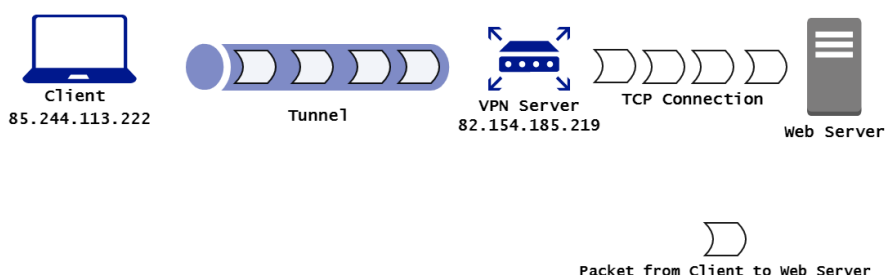


Figure 2.2: Simple representation of the tunnelling technique used on VPNs where the web server will see incoming packet from someone with IP address 82.154.185.219 making the client's IP address protected

This solution provides speed and security as we are only adding an extra stop for our traffic at the VPN server, and it is very difficult to correlate the traffic that a VPN server is outputting with the traffic their receiving. However, the problem behind VPNs is that it doesn't protect the servers in terms of anonymity and the clients must trust the owners of the VPN server as they can see the connections established through their server. So this solution doesn't eliminate the hypotheses of the owner sniff into its client's connections, and it also raises this week point where exploitation on a server like this can lead to a leakage of history logs and expose users to their activities [4]. An interesting fact is that this has happened before, and the owners of those attacks often sell that data on the Dark Web.

2.3 Dark Web Softwares

As the Dark Web is built within the Internet but with its specific protocols and software, there is no such thing as general software to access the Dark Web like the web browsers we use to access the Surface Web and the Deep Web. This idea of using global infrastructures of the Internet to built-in some extra layers to guarantee high levels of encryptions and anonymity allowed developers to create software that would open the door to what is intrinsically the Dark Web. Actually, we will understand that later, but every technology behind the different Dark Web systems we will analyse is built on top of techniques and technologies that existed long before these Darknets appeared. People understood that by gathering the right technologies, they could build different pieces of software with strong protocols that would result on platforms to more than nothing guarantee privacy and anonymity.

The most popular networks that live inside the Dark Web are Freenet, I2P and the Tor Project. However, in this study, we will just approach the first darknet released at the beginning of the 21st century and the most popular darknet at this moment, and that changed the definition of the Dark Web, the Tor Project.

2.3.1 Freenet

The Freenet³ was the first software that opened the door to the world of the Darknets in the year 2000. In a brief description, Freenet is an open-source solution that allows users to share data between them using “freesites” that are only accessible through the Freenet software.

The Freenet works on a peer-to-peer (P2P) network that has the ability to self-organise its structure. Surely the creators didn’t want to follow the recent example, at that time, from Napster, a file-sharing software that, apart from being also over a P2P network it was dependent on a central server that would handle requests and redirect them to the right computer making it possible to law enforcements close the service by shutting down the central server that worked as an index for Napster’s content [38]. The Freenet on the other hand works on a decentralised P2P topology that allows it to be more robust and eliminate dependencies on a single point of the network as there was with Napster.

A question that might surge is that if this is software to increase security and anonymity, how can we trust a P2P network that can easily integrate malicious users? This is assured with strategies to guarantee data integrity such as hashing algorithms and also strategies to prevent jeopardising users anonymity [18].

So how do Freenet works? When we download its client software and run it for the first time, we become a Freenet participant, allowing the entire network to store data at our computer. A node is identified by asymmetric encryption, in other words, by a pair of public and private keys. Users have an option to add new files by sending an insert message with the encrypted file with the owner’s private key and a globally unique identifier⁴ (GUID) to a set of nodes, having, therefore, the designation of publishers. The GUID key used here is generated through hashing the file content using SHA-1⁵ algorithm providing data integrity protection to the respective file. Now when some user wants to retrieve a file, a message containing the GUID from that file must be sent, and the network will reach the node where it is stored and send it back to the user who requested it. Between these requests, either for adding a new file either for requesting a file, there is a protocol running that allows anonymity and security for recipients and senders. So messages, instead of travelling from the sender to the recipients, must pass through middle nodes to assure that no request can be linked to any user. For this to work, each node in the software only knows about its direct neighbours, resulting in that even the node before the recipient and the node after the sender cannot tell if the next or the previous node is the recipient or the sender, respectively [18].

³More information can be obtained at [http:// www.freenetproject.org/](http://www.freenetproject.org/).

⁴128 bit number that is unlikely to be duplicated besides not being managed by a central authority. More information at <http://guid.one/guid>

⁵Secure Hashing Algorithm that delivers a 160 bits message digest. There are also the SHA-2, and the SHA-3 that are stronger and deliver a message digest with 224 bits or more depending on the implementation [19]

Searching Files In terms of indexing data, as told before, Freenet doesn't use a centralised system as Napster does. Instead, it makes a search in-depth among the existent nodes of the network. So to reach a file, we first need to know who can deliver it, and for that, each node must forward the request to another node that the current node believes is closest to the destination. To do that, each node has to hold a routing table with the addresses of its neighbours and the GUID keys they hold. With that information, a node can check if they own a file according to the GUID file requested, and if so, they return an identifier with itself as the data holder. If not, they must pass to the neighbour that has the closest GUID key to the one requested. After the request reaches a node with a copy of the requested file, it starts a new procedure of returning to the origin of the respective file. Through the way back to the sender, each node can cache a copy of the file locally to increase speed on future requests. On this path back to the sender, each node must replace the holder tag (a tag that identifies the data holder) with their identity [18].

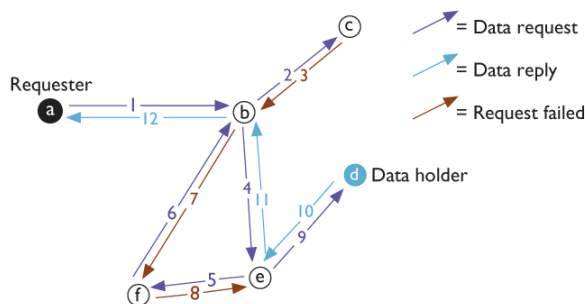


Figure 2.3: Request example that shows system recovering from dead-end at step 3 and from a loop at step 7. Retrieved from [18].

In the figure above, we can see an example of a possible scenario of a request for a file. This figure shows how data travels through the network showing two situations of failure that are solved by the protocol, respectively, a dead-end and a loop situation [18].

Data Store A datastore is a directory created on each node that will serve as storage available to the network. The Freenet protocol does not allow users to have control over what is stored on this directory. A concern that can be obvious is how data is stored on a given node? We can be talking about criminal data that can be stored on our computers and making us a point of sharing dubious activities. Therefore data stores data can be encrypted, making it invisible from the nodes that own that data making the data store consisting of GUIDs that are attached to the files it stores in the eyes of the node that is holding it [18].

Improvements Over the years there have been some improvements on the Freenet software, and at the time this document is being written the current stable version is the 0.7.5 and was released in 2014. One of the biggest changes on this software was on the release of version 0.7 in 2008, where the software allowed users to work under two distinctly

operations modes: opennet and darknet. The opennet mode was the basic one where all nodes are strangers and connected to the whole network. On the other hand, the darknet mode allows users to manually insert which nodes they want to connect to, in the form of a friend-to-friend operation mode where we decide whom we allow swapping requests with us. This makes the opennet mode less secure than the darknet as we do not control who is connected to the network, raising some risks for users using this mode of operation.

Freenet has much more details that could be explored, but we're not focused on understanding step by step how this specific software works as we just want to show how was the first system that opened doors to the Dark Web and understands in general lines how it allowed users to exchange data anonymously. Therefore, in summary, Freenet offers a file storage service that is guaranteed by all the anonymous participants that share a piece of their storage to create huge disk space for the community. It is slower as expected compared for example, with regular browsers in retrieving information as there are extra steps here in order to find where the data is located and in scenarios where the data is located in different nodes because if files are too big, they can be split and stored in pieces in different nodes, there will be an extra effort to link the different parts of the content to retrieve the whole file [42].

Overview In terms of accomplishing what it offers the Freenet offers high levels of security, and as we will see further in this document, this software offers the best results in terms of anonymity and accessibility as data from a publisher can still be available even after the publisher go offline. This architecture has a cost, and it comes in three forms: it is not that easy to use by the general population, the speed in terms of obtaining content is not great, and by last it can only serve static content meaning JavaScript cannot run on these pages as well as server-side scripting as there is no central server[42].

Apart from being one of the most secure forms of accessing the Dark Web, the Freenet has some weaknesses that were explored in a study developed in 2012 that approached an attack based on routing table insertion (RTI) attacks [7]. This study shows a technique over the opennet mode of the Freenet that required some knowledge from the network topology that can be achieved by searching for documents using a specific function from the Freenet implementation named `probed`, that is an undocumented debug function that returns all the nodes a request passes through. This is a big flaw that helps the effectiveness of the RTI attack because it helps to find a target node that we want to break some of his anonymity. The idea behind the RTI attack is that by creating fake nodes surrounding the target node, it will be possible to identify if a request is from the target and if the target holds a specific file because all in and out messages to the target have to pass through the attack nodes [7].

Another study presented in May 2017 explores a technique to tell if a neighbour node is the receiver of a file or if he is just forwarding the requests to other nodes [14]. This

study uses a Bayesian framework⁶ that allows a passive node just by analysing the traffic to determine if its neighbour is the one downloading a file or not.

2.3.2 Tor Project

The next software that allows users to visit the Dark Web is designated as The Onion Routing⁷ (Tor). This is the most used darknet nowadays, and it has to do mainly due to its nature and its ease of use because it looks like a usual browser, and from the user perspective, it acts exactly like one. This is an important aspect of the Tor network because not only does it allow users to use the Dark Web easily compared for example with Freenet, but it also improves security as a network with higher traffic density makes it harder to track and monitor users, ending in some strategies like time analysis having less efficiency as we will study further on this document [8].

The Tor Project is based on a network solution found in the 90s by the U.S. Naval Research Laboratory to solve the question: how can we have a network that does not reveal who is talking to whom, assuring anonymity and secure communications? The solution has one name, and it is called the Onion Routing⁸. This technique allows onion networks (i.e. networks that use the onion routing technique) to be a safe environment for clients when it comes to privacy and anonymity. On the regular browsers we use, like Google Chrome or Mozilla Firefox, we connect to web servers to retrieve pages by using TCP connections to their port 80. This TCP connection will exchange packets that will contain data that can be encrypted if we are using SSL/TLS protocol and other information such as the IP address from the client. With this, internet service providers (ISPs) can easily see who is a user talking to. However, by using the Tor browser, we can also hide the information that identifies a client, such as the IP address. It also allows servers to stay anonymous on what is known as hidden services. Before we dive in-depth into how the Tor Project works, we will first take a look at its main technology under the hood, the onion routing.

2.3.2.1 Onion Routing

This technique provides connections that are resistant to eavesdropping and traffic analysis by making connections pass through different onion routers before reaching the destination. In other words, we can see this as a client that asks someone else to exchange data with a server on his behalf. This “someone else” will be onion routers that are publicly available on the network, and therefore no one can blame them for being searching for some kind of content. It is important to note that we will only consider the most recent

⁶Bayesian framework is an inference technique that allows through probabilities create a system that englobes pieces of evidence to give an informed probability in an outcome

⁷Tor Project history and origin can be seen at <https://www.torproject.org/about/history/>.

⁸Information about Onion Routing in the eyes of its creators can be seen at <https://www.onion-router.net/>

version of this technique as the original one suffered some modifications. After all, this technique was developed 20 years ago. So how exactly this technique works?

The best form of explaining how it works is by imaging the scenario of a client that wants to talk with a web server, for instance, to obtain a web page. First, it has to establish a path of onion routers that will act as a middleman transporting the data that will be exchanged between the server and the client.

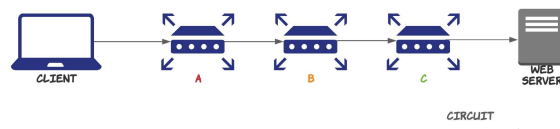


Figure 2.4: Example of an onion routing circuit using 3 onion routers each one with its respective public key.

Before we dig into the example, let us establish some concepts:

- The path, also referred to as a circuit, is composed of onion routers that only knows about their successor and predecessor. Therefore only the node that has a direct link to the client knows his IP address.
- Every node will have a public/private key (RSA algorithm) that will allow them to authenticate themselves and guarantee that there is no one in the middle of the connection.
- To protect the client, he will not have a public/private key, so to check the integrity of the data it will be used a hashing algorithm such as SHA-2, but we will see that in a few minutes.
- On these networks, there has to exist a directory server or directory nodes that have special information about existing nodes in the network, such as their IP addresses and their corresponding public keys.
- A circuit can have a different number of onion routers, but we will consider that a circuit has 3 onion routers as it is how the Tor Project implemented it.
- The first node will be designated as the guard node, the last as the exit node and the intermediate nodes are designated as relay nodes.
- To encrypt data, the onion routing technique uses the Advanced Encryption Standard AES⁹ symmetric key algorithm with the counter mode of operation.

⁹More information about AES can be seen at <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>

Cell Structures A request is sent in a cell that is a fixed-length packet that can either be a control cell or be a relay cell according to what command is passed along the cell. A control cell is a cell that should be interpreted by the node that receives it, and its payload has data according to the command field it holds, in the other hand, the relay cells are characterised as so when on the command field it is passed a relay instruction making the payload of these cells a combination of fields (relay header fields + data field) as we can see on the next figure. It is important to mention that the control cells only have the data field encrypted, and relay cells have their data as well as their relay header encrypted. [21].

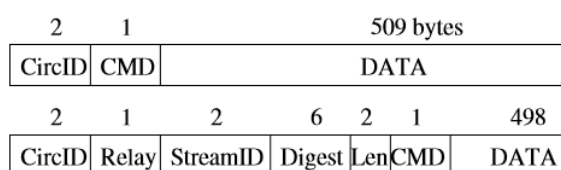


Figure 2.5: Onion routing control cell structure and relay cell structure, respectively. Retrieved from [6].

As we can see in the figure above, the control cell has a header composed of two fields: the circuit identifier (circID) that allows identifying a circuit and will be different for each link (i.g. the link between the client and router A can be designated as CCA but the link between the router A and the router B can be designated as CAB), the second field of the header is the one named command (CMD) that will define what operation the sender wants the receiving node to do with the data field that will be the payload of the message. Once again, on these cells, only the payload is encrypted using AES.

As we can see below the control cell structure, we have the relay cell structure, which is composed of 7 fields. The first one, just like with the control cell, carries information about the circuit ID that the cell is travelling on. The following field works exactly like the CMD field on the control cell structure, but, as we are talking about relay cell structures on that byte, following the circID will always indicate a relay cell, which is why it appears on the figure as “Relay”. Following this field, we have the relay header that is composed of 4 fields: the stream identifier (stream ID) that can identify which stream does this package concerns to as there can be multiple TCP streams under the same circuit, the integrity check field (Digest), the size of the payload (Len) and the operation to be executed on the relay node (CMD) with the data. At last, we have the payload that is only composed of the data field. In terms of encryption, these cells only encrypt the four fields of the relay header and the payload, leaving the circID and the Relay field with no encryption.

It is essential to understand that the stream ID allows the client and the exit node to know to which server are they receiving/sending messages because a client can have multiple stream TCP connections over the same circuit. The circuit identifier allows the relay nodes (including the exit node and the guard node) to know where does a cell should

go because a node can be serving multiple clients at the same time, or even other nodes because an entry node on a circuit can be a relay node on another circuit or even an exit node on other circuits.

So bearing in mind that these routers exist only to relay data, the client has to request a circuit to the directory server to establish a secured connection to each of the three onion routers that will be assigned for his connection. After the client obtains the IP address and the public key of the three routers he was assigned to, he must pick them randomly and start negotiating with each one of them a symmetric key that we will designate as K to allow the encryption of the data.

Establishing a circuit So the process to establish the circuit will start with the client creating a shared key between him and router A. To do that, the client will use the router's A public key to perform the Diffie-Hellman key exchange in order to create a shared key, K_A , that will be used to encrypt the data payload using AES. The client will also send a generated unique circuit ID to identify that link. We will designate the first link as CCA (in an abbreviation of the circuit-client-router A). This circuit ID will be stored on router A with the K_A in order for the onion router A to be able to communicate using that link. The router A after receiving the request it sees that it is a control cell with the CMD "create", so he understands that someone wants to create a circuit with him, so it will apply his private key on the data payload, calculate the shared key according to Diffie-Hellman half key sent from the client on the data payload and reply back to the client in plaintext his half of the key so the client can also generate the same shared key according to the DH algorithm. It is essential to highlight that the reply is sent in plaintext to avoid using the client's public key, overall we are trying to avoid sharing as much information as possible from the client, so it becomes harder to track where it is. However, how can the client know if someone modified the message? It is solved by router A adding a hash of the shared secret key generated on the message reply so the client, when receives it and calculates the shared secret key with A, can verify that the hash of the key matches the hash sent on the reply message [3].

Then the client picks another IP address and its respective public key from the two left of the three obtained from the directory server and creates a request using the Diffie-Hellman method again to create a shared secret key with the next router that will be router B, a relay node. To achieve that, the client must encrypt his generated DH half key with the public key of router B, and then, it will add that result into the data field of a relay cell that will have a CMD defined as "extend" to extend the circuit. This relay cell, excluding the circID and the CMD, will be encrypted with another outermost layer of encryption with the shared key K_A , previously negotiated with the onion router A. With the request created, it is sent to the router A that will see that there is a relay cell sent from the client that he shares a key with, and he knows that it came from the client because through the circID he received, he knows to which circuit the received cell belongs to. After decrypting it the router A will observe that the relay cell has a CMD saying that

the client wants to extend the circuit, so what the router A (entry node) does is create a control cell with a new generated unique circuit ID that we will call as CAB (circuit - router A - router B), add the command “create” on the CMD field, copy the data field from the received relay cell to the data field of the new control cell that will contain the DH half key from the client so the router B can establish the key with him. The router B will find that a client wants to extend a circuit with him, so he will decrypt the data using his private key, generate the shared key KB, calculate the hash of the generated key and send a reply back to the entry node with the circuit ID CAB and his half key of the Diffie-Hellman with the hash of the new generated key KB informing that the circuit was created. The entry node receives the reply from the circuit CAB saying circuit created, creates a relay cell informing that the circuit was extended, adds the data received from router B to the relay cell, encrypts it with his KA and relays it to the client that will decrypt the payload with KA, generate the shared key with router B and compare with the hash to assure that the data was not manipulated.

At this point, we’ve established a key with router A, KA, and a key with router B, KB. Then the last step is to repeat it again to establish a shared key with KC.

So again, the client using the Diffie-Hellman algorithm, encrypts a message with his DH half key using the router’s C public key and create a relay cell with the command relay extend encrypted with KB followed by KA and forward it to router A with circuit ID CCA.

Router A sees that it is a relay cell with command relay data that must go to router B, so it decrypts the data field using KA and relays it to router B. This router receives the message, applies his KB and observes that it have a command relay extend, so he must create a control cell with the new circuit ID that we will call CBC and send the data from the replay cell to router C. Finally, the router C receives the message from B, generate the shared key with the client, calculates the hash of the key and sends back a reply with the hash and his half key for the Diffie-Hellman. Now it goes back through routers B and A, where each of them encrypts with their respective previously established shared key. The client receives the reply, uses KA, KB and then, with the reply from the router C, generates the shared key, calculates the hash of the shared key and checks again if this is matched with the hash provided on the message from the router C.

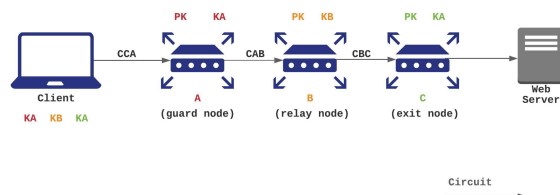


Figure 2.6: Example of an onion routing after establishing the circuit to the web server.

Communicating with a server After having the circuit established, as we can see in the figure above, the client is now able to make a request through the onion network. For that, he has to create a relay cell with the command “begin” that will transport information about which server (IP address) and port that the client wishes to connect and encrypt this message with three layers of encryption using KC followed by KB and lastly KA. The router A receives a cell from the circuit CCA, so he knows that he came from that specific client, and after observing that it is a relay cell, he applies his shared key with the client, KA, and removes the first layer of encryption. Now the client forwards his encrypted cell to the circuit he has created. This router observes that he cannot understand the decrypted result, so he forwards it to the next hop in the circuit connected with the CAB. Before forwarding to router B, the router A changes the circID field to CAB. Then, router B receives the cell and does the same as router A. When the cell arrives at the exit node (in our example, the router C), this node observes that after decrypting it with KC it has a relay command “begin”, so the exit node starts a TCP connection with the server specified on the payload. The reply from the server will go back to the exit node that will create a relay cell with the relay command “data” and encrypt it with KC. Then it passes it to router B that will encrypt it with KB, and again the same for router A. When it reaches the client, he decrypts all the layers, checks that the connection was established, and can now send another relay “data” command through the circuit to communicate with the web server whilst assuring his anonymity. These layers of encryption (three in our example) makes this technique be compared with an onion as there are multiple layers protecting the interior, and a perfect example of that can be seen on the figure below showing a message created at a client ready to start being “peeled” of by the circuit it is going to.

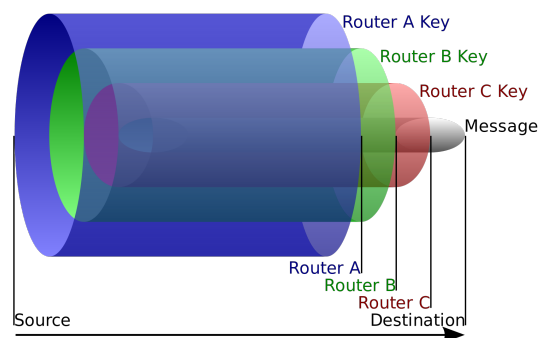


Figure 2.7: Message encrypted at a client showing multiple layers of encryption.
Retrieved from [41]

Few notes to help solidify some concepts that might be more confusing:

- There are two types of encryption used during the circuit establishment. First, we use RSA to encrypt the Diffie-Hellman half key that travels inside the data payload. Second, and after we establish a shared key with the nodes, we use AES to encrypt

the payload of a control cell or the relay header and the relay payload from a relay cell.

- An onion router can be working at the same time as an entry node, relay node or even an exit node for different circuits, therefore the circuit identifier (circuit ID) goes on every cell, either if it is a control cell either a relay cell, because without that the nodes that receive that cell wouldn't know from where it came and therefore wouldn't be able to remove the correct layer of encryption and pass it to the next node on the circuit. It is also important to understand that the circID field is changed when passing through any node because every link on a circuit has a different circID.
- As there can be different streams using the same circuit, and the client needs to know to which stream/connection does a request belongs, there is a stream identifier (stream ID) that will be mapped on the exit node according to the connection so when an exit node receives a reply from the server he knows to which circuit it belongs to and to which stream.

2.3.2.2 Onion Network

Now that we have this background on how the onion routing works let us study how is it applied to the most famous onion network, the Tor Project. When we first install Tor on our machines, we will also be running a localhost server named Onion Proxy (OP) that will act as an intermediate between the data exchanged from our applications to the onion network. This OP, on load up, gathers information about Tor's relays from the directory servers and creates what we saw on the explanation about onion routing, a circuit with 3 nodes [10].

As so, we use all the mechanisms that we would use on a normal browser to achieve a web server: we write the address we want to reach, our browser performs an address resolution through DNS, performs a TCP handshake with the IP retrieved from the address resolution and finally, we start communicating with our server through HTTP requests or HTTPS if we are using SSL/TLS that requires an additional exchange of information to generate a shared encryption key to allow secure communications. However, this is just if we want to remain anonymous and access web servers that are public and visible to anyone, because if we want to use the Dark Web we have to be using servers that are hidden, and therefore mechanisms such as DNS do not work, and there is a different form of establishing a connection with these servers that are known as hidden services.

To clarify some thoughts, we can use the Tor Project implementation of the onion routing to navigate anonymously through the web, even if we are talking about the Surface Web, Deep Web or Dark Web. However, in the perspective of the web servers, those will only gain this ability to be "invisible" if they are hosted within the hidden services that the Tor Project provides, which are an extension of what we've seen on the

section about the onion routing in the sense that we will have 3 hops to hide the client as usual and we will also have another circuit with 3 hops to hide the web server and the client meets the server in the middle on a point labelled as a rendezvous point. We will see that in detail in the next pages. The important here is to remember that Tor clients are always protected with their 3 hop circuit. However, this is not true for web servers as those are only protected when deployed on the hidden services layer that lives in the Dark Web.

Tor Nodes So the Tor Project is a big onion network that currently has almost 7000 nodes serving interchangeably as guard nodes, relay nodes and exit nodes [57]. Anyone can run a relay at their homes, and Tor relies on these volunteers that provide bandwidth and support for the Tor network. Nevertheless, it is advised that volunteers do not run an exit node at their homes because, as we have seen, on the onion routing technique, these nodes are the ones that communicate to servers and therefore, they can be identified and seized. Instead, Tor recommends that users should run these types of nodes on hosting facilities in the most transparent form.

Another relevant aspect of these nodes is that, as we observed before, they are listed on a directory server in order to be publicly available to Tor users to create their circuits, but if their information is publicly available, why aren't they taken down? The real answer is that they can be taken down by governments and Internet Service Providers (ISPs) if they want to. In countries such as China, an oppressed country, the government blocked all the nodes provided by the Tor publicly directory server. However, Tor's mission is to give freedom and speech to people who live in such countries. To solve this inconvenience, there is a different type of nodes designated as Bridges that are in every aspect similar to the other relays we have seen. However, they are not available publicly by directory services [54].

So to stop Tor from running on these censored countries, there are two forms: block the connections to the known IP addresses from Tor relays that can be obtained from the directory server, or analyse traffic on the network to identify the Tor protocol [55].

Tor Bridges In comparison with Tor's relays, there are at this moment around roughly 1750 bridges that allow users to access the Tor network no matter where they're living. Besides these bridges being hidden publicly from governments, they can also be blocked if Tor protocol is being detected on the packets that go to these bridges. There are software tools like the CapLoader¹⁰ that is a software capable of analysing traffic and, with the right settings, identify protocols such as Tor through statistical analysis. This software uses Deep Packet Inspection (DPI) in order to classify the traffic by protocol, and if they detect that a connection is running on the Tor network, these routers can be blocked by the government. A solution to prevent software such as CapLoader from identifying

¹⁰Software designated to analyse huge amounts of captured network traffic. More information at <https://www.netresec.com/?page=CapLoader>

traffic using Tor's protocol is a protocol known as the Obfuscator (Obfs4) that offers obfuscation layers to prevent protocols from being identified through message contents. This protocol is one of the most effective ones, but there are others such as meeko, format-transport encryption (fte) and scramble suit that are available to use with Tor. These protocols are part of a mechanism known as pluggable transport (PT), and they, in summary, transform the traffic from a connection between a client and a bridge to packets that look usual in the eyes of the network and therefore making it difficult for the detection of Tor communications on the network [55]. Apart from being the state-of-the-art of these techniques, the Obfs4 has some researches oriented to understand how can it be surpassed [29].

For a user to use a bridge relay, he must first find it through 3 different options: send an email to bridges@torproject.org, request them online through the BridgeDB website (<https://bridges.torproject.org/>) or request them on the Tor browser settings.

Hidden Services The main reason why Tor is so important and used is due to what is known as the hidden services. The hidden services were introduced to Tor in 2004 and allow web servers to accept connections and exchange data without revealing who they really are. This is useful as some owners of these servers don't want anyone to know where they store and provide content that can be sensible.

To explain the hidden services, we will keep with the example shown in figure 1.3. Now, not only the client will have his IP address hidden, but also the server will gain this ability. For that, the server will have to run, just like the client, behind an onion proxy and establish a circuit for him to meet the client in the middle on a node designated as a rendezvous point [10]. Nonetheless, it is important to understand how addresses work on hidden services as we're talking about servers that don't want to share their IP address.

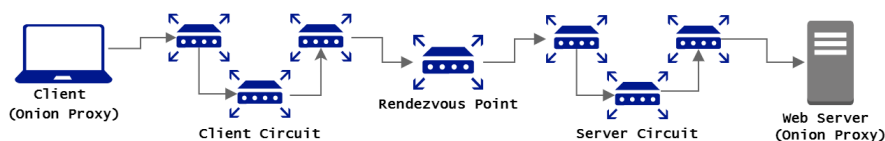


Figure 2.8: Example of a connection within the Tor network using hidden services

Onion Addresses As these servers work under a specific form, they are identified by being under the root domain .onion. This allows protecting these servers from the Domain Name System (DNS) that is used to make address resolutions in the other parts of the World Wide Web. These domains are created by hashing RSA's public key of a web server with SHA-1 and using the first 10 bytes of the result by encoding these bytes in a 16-character base32 string [10]. As you may imagine, this will result in identifiers that are not human friendly, at least from the reading perspective. There are also tools such as

Shallot¹¹, which allows generating an RSA pair of keys that would result in an onion address with some specific text on it by brute-forcing onion addresses. Below, in figure 1.10, we can see an example of running this tool to generate an onion address with the word “test” that took 99133 tentatives. As a curiosity, to generate an onion address with a specific word of 14 characters would take roughly 2.6 million years [32].

```

$ ./shallot ^test
-----
Found matching pattern after 99133 tries: testvztz3tfoiofv.onion
-----
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQC3R85m6NQA1ZjaYqvz1hvFIjbl4RtdJb68h1C9xEBkvfr/BG
8Z5vDiUzdbDt8mEBuZUDanx80uGJvbXTgmczX0UlkE0gGiZ8RKpnsbKaf/EJNrIw
T7MSXQmWncm22nDeViV7fvy+Usya12RE5cdVCFsPtEbVZqCum1KkEgCyFwIDBAZ7
AoGBAJSa2cGuru/XhzJAEAIwH2bgPDnum9T/srOYxUKW6afHZe0u5S4Cc1wb+xb/
pG0tzn71XZfCKMfiVdxB/f3XTcRrYB2VnBoNtoTD7WfH6DksdDf4zunqiEjvx19K
R+tKxmf7OedrRt8wIhUmFd1E2Q9nbTHI6icdB4kR4QkYKZzAkEA5M6samK7+495
6SwPRXiePIs7sHKWuxdCrG7kWR5RNjrv2Cc6YwK46TPcaXBcRfM4eq9+9PGoKi0IO
gSpOZ5vRYQJBAM0QAZYTZ6ApD014x372MX1ZNofuYL/+XF8ZPZV6Sh4+9MUBuNPb
yL7BENDr6pX4Zm60epvAphhCa4vGno2pHncCQQCQnfhUCHANU4bjtX4E0oI63WDq
UwB0eIWxu0YvGt7Z25Dg9CNz/ax8UZOj6VyKxLRbR9+K3mNrNgaopw+ZDKzAkEA
ttgTK1ALe+3v+5H+Ez1SvFPREDfCHihrfD1Ipc5ziCY9ixTArgdyZvk+Pi+AMBVV
sL2HwvjRLEAgRc1vKfkWwJAFtM+BIGRM5me+fMALuBBEtKnBj6maf1syucErEb0
pIIBkovF5oyW031SBmtStEIANnkH0g8aXqjcgPKusDN7CQ==
-----END RSA PRIVATE KEY-----

```

Figure 2.9: Example of how Shallot would behave when asked to generate an onion address with the word “test” on it. Retrieved from [32]

We might wonder why does Tor uses this form of providing domains that are so difficult for human beings to memorise. The answer has to do with the fact that these servers have to be certified in order to prove that they own some domain. Outside the Dark Web, this is done by central authorities that manage how the domains are assigned. Together with the DNS, it allows solving queries for specific domains informing whether they exist or not. In the Dark Web as we have seen before there is no DNS because that would compromise all the effort for hiding servers as they would be listed on central authorities with their IP address, therefore the point of creating onion addresses by using the hash result of the server’s public key is that we can easily verify that some domain belongs to a server if the public key hash matches the domain address. So the proof of domain is done through math operations instead of through central authorities.

Deployment of an Onion Web Server For a web server to be online on the Tor network through hidden services, it must complete a few steps. The first is, as explained in the onion addresses paragraph, to generate an RSA pair of keys and, with the first 10 bytes of the hash of the public key using SHA-1 and base32 encoding, create its .onion address [11]. Then the onion proxy of the web server has to choose some nodes of the Tor network and establish a circuit to each of them. These nodes will act as introduction points that will allow clients to connect with the web server, but we will get to that in a second. After having a circuit established for those nodes, the server’s OP has to generate two service

¹¹ Tool that allows generating onion domains with some specific text on the .onion domain. More documentation can be found at <https://github.com/katmagic/Shallot>

descriptors and send them to 6 hidden services directories servers that will be chosen during the generation of the services descriptors. The descriptors are files that hold an identifier, information about the list of introduction points and the hidden service's public key of a specific onion address. These hidden services directories are relays that can handle requests asking for descriptors so a client can know to which introduction point he should make a request to communicate with a server. [11].

Establishing a connection with a hidden service To start a communication with a web server that is deployed on a hidden service, we first have to find its onion address. The client with the onion address of a server will compute the descriptor identifier and the list of the hidden service directories that hold the server's descriptor [60]. With the descriptor ID and knowing in which hidden service directories to find the specific .onion hidden server, the client makes a request to a hidden service directory for that descriptor ID to get more information such as the introduction points of the hidden service [60].

So now the client knows where the introduction points of the server are located, and with that, he first establishes a circuit with a rendezvous point, also sending a cookie designated as rendezvous cookie, that is a random 20-byte value [60]. This cookie allows the onion router of the rendezvous point to link the circuit with that cookie. After having established this circuit, the client will now interact with one of the hidden service introduction points, sending a cell containing the IP address and rendezvous's fingerprint¹², the hash of the public key of the web server, and the rendezvous cookie [60]. The introduction point when receives this cell he check on its directory if he servers that server through the hashed public key sent on the message. If so, he introduces to the web server that someone wants to speak with him on the specified rendezvous point with a specific rendezvous cookie on an encrypted cell with the server's public key. The web server decrypts that message with its private key, and with the information on the cell, he can know to build its own circuit to the rendezvous point. At this point, the rendezvous point will just exchange relay cells between the circuit with the client and with the server [60].

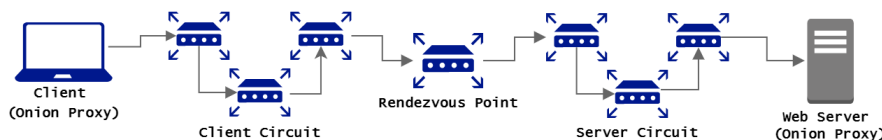


Figure 2.10: Hidden Service architecture example

In comparison with the usage of Tor without the hidden service layer, we can see that after establishing the connection, there are 3 more hops on the way that serves to hide the web server from the client. In this way, the client does not know the real IP address from the server neither the server knows its client IP address [60].

There are a few aspects that, at this point, might be not so clear. For instance, who are these hidden services directories servers? And how are they established? The answer is

¹²The fingerprint in the Tor Project is the result of hashing a relay public key using SHA-1.

that these servers are just normal nodes with a special flag attributed that allows them to run also as hidden services directory servers [10]. These nodes must be running for at least 25 hours in order to gain that status, and it is assigned by Tor authorities, named directory authorities, which are nodes that have special information about the network [10]. This can also be pointed as a weakness from the Tor Project because if these special nodes are detected and seized, it could mean the end of the Tor network. To avoid letting such important nodes be running on strangers hands, the Tor Project hard-coded their directory servers with known and trustable nodes, most of them running under the nodes owned by developers and collaborators. There are, at the time this is written, 9 directory authorities¹³ servers that are spread through the United States and Europe.

Another important aspect that wasn't mentioned and that is important is the fact that service descriptors have to be updated every 24 hours from the hidden services, meaning that a hidden service has to send its descriptor to 6 different hidden services directory servers and remove the old from the old hidden service directories that were serving the hidden service's descriptor [10].

2.3.2.3 Overview

To summarise, Tor hidden services have some relevant aspects, important to mention again, that differs from what we are used to in the conventional web services we know:

- Have specific domains ending in .onion that can only be accessed using Tor.
- Tor domains are intentionally hard to be memorised by humans as they are the result of hashing the server's public key to have a form of authenticating the server.
- Tor hidden services do not use DNS as there is no central authority to go to asking whom an address belongs to.
- We can still use the Tor browser to search websites that do not use hidden services but still having our anonymity assured as Tor always applies the onion routing on the client-side.
- Tor's onion addresses are usually hard to find as there is not a specific search engine to query for those.

As a curiosity, Tor's project team developed a tool named ExoneraTor¹⁴ that by inserting an IP address, it returns a message saying if it belongs to a Tor relay. This was developed because law enforcement agencies were doing needless raids after the computers running these IPs, thinking they found users accessing illegal content, and it was just a Tor exit node communicating with web servers on the user's behalf.

¹³Current directory authorities servers can be seen in real-time at <https://metrics.torproject.org/rs.html#search/flag:authority>

¹⁴Tool developed by Tor that helps identify if a certain IP belongs to a Tor relay or not. Tool available at <https://metrics.torproject.org/exonerator.html>

Some approaches to break down the ability of hiding users can be explored, such as time correlation analysis, traffic analysis strategies, and Sybil attacks¹⁵. Most of these techniques rely on something known as traffic confirmation attacks, which can be conducted when an attacker can observe an end-to-end communication [11]. In Tor, this means that an attacker must have control of the guard node and the exit node of a communication to be able to use traffic confirmation attacks that can be explored after for instances with time analysis correlation to match an input on a guard node from a client with the output of an exit node, or with traffic analysis to find patterns in the packets allowing an attacker to possibly find who is using Tor and what are they doing. An example of applying a protocol-level attack can be seen at [26] whereby manipulating Tor cells it's possible to easily correlate traffic navigating the Tor network, and again in combination with a traffic confirmation attack, it can be powerful to detect users activity in the Dark Web. The Sybil attack is an enhanced version of traffic confirmation attacks in the sense that an attacker owns multiple nodes of the network, allowing an attacker to have more knowledge of the network and more control [11].

These techniques can be used simultaneously, for instance, using time analysis correlations based on getting traffic that is entering the Tor network (entry node) and leaving the Tor network (exit node) and correlates the requests from a client with the requests made to web servers on an exit node while owning the entry node and the exit node at the same time could lead to a powerful mechanism to deanonymise users easily on the Tor network. However, Tor has knowledge of all of these techniques, and therefore they patch strategies to “fight back” these tentatives to disrupt the network. For instance, on a single IP, Tor doesn't allow more than 2 nodes to be running [11].

A study from 2013 shows there are some weaknesses related to Tor's hidden services descriptor distribution procedure [11]. This study shows how it is possible to reveal these hidden services IP addresses by collecting service descriptors of hidden services. However, for this technique to work, it is stated that the attackers must control the hidden service directories for the hidden service we are trying to unmask. The trick behind this approach is related to how a hidden service generates its service descriptors and how it picks the 6 hidden service directories to which the server has to send its descriptors. The idea is to, with Tor's algorithm, calculate which are going to be the possible future hidden service directories to which a hidden service will upload their descriptor because, as we've seen, these directories change on a 24 hours basis. This algorithm is based on arranging the fingerprints from the available hidden services directory servers and picks the ones with the closest fingerprint to the fingerprint of the hidden service [11]. Having found which fingerprints are the ones to be possible picked by the hidden service in the future to upload its descriptor, the attacker must run Tor relays with the computed public/private key that matches the fingerprint and wait possibly 25 hours to become

¹⁵The Sybil attack was named after a study of a woman diagnosed with dissociate identity disorder and it computer security it is related with having multiple entities on a system in a form where the owner of this multiple entities can have a privileged view on the network [23]

one of the hidden services directories, which as we've seen is not that linear because they will only be assigned as hidden services directories when the nodes from the directory authority activate that flag. Suppose the attacker's nodes go lucky and complete all these tasks without any problem. In that case, this could lead to a big flaw in terms of allowing anonymity because through understanding how Tor works, it was possible to target hidden services, be their hidden service directory and with a traffic correlation attack get their real IP [11].

Apart from the superstitious dark side of the Tor Project, this is a network that is used by banks, diplomatic, officials, law enforcement agencies, activists and much more, but in the end, they all seek for the same, do their job without being worried of being tracked and monitored [5]. There is a study from 2014 that shows through statistical analysis from the Tor network that the content devoted to the criminal side and content devoted to the real objective of the Tor network (i.e. allow people to have the freedom and have their right to privacy) is roughly the same [10]. This study also concluded that the most popular hidden services are used by botnets¹⁶ and hidden services serving adult content.

There is so much that could be said about the Tor network as it is an extremely active system that is evolving day after day. There are a lot of researches around, as we saw, that puts the system under pressure when it comes to their principal goal, anonymity. However, it is interesting that all of these works also explore countermeasures to these possible flaws, which shows that the existence of all of these researches is a tentative to explore an open-source protocol such as Tor and sensitise its developers to some approaches that could disrupt the network.

These works are of huge importance as they raise awareness for the Tor network's fragile aspects and the Dark Web. This ease of use from the user perspective of the Tor network makes this network the most popular [42], with more than 2 million users and more than 175 000 hidden services up and running at this moment [58],[56]. This makes the Tor network the centre of attention when someone speaks about the Dark Web, and it is definitely an excellent case of study to understand: how the Dark Web works? How is it structured? And how is it organised?. Therefore, it is vital to study and understand how we can apply models to answer those questions.

2.4 Dark Web Models Approximations

In this section, we will approach some techniques used to describe and model the Dark Web. It is relevant to mention that as described before, most of the following techniques we will explore are going to be centred on the Tor network as it is the most popular darknet and for all of the other reasons we explored in the previous section.

¹⁶A botnet is composed of a big number of computers running a bot with the same end. It can be used, for instance, to deploy a distributed denial-of-service attack, send spam, mine cryptocurrencies, etc. These networks can incorporate computers from users that are not aware of this and that are part of the botnet through malware.

Network analysis is an important asset to provide relevant information that can help us understand how information is being diffused and shared among network structures [39]. There are different forms of analysing a network, and each of them gives a unique perspective that depends on what we are trying to analyse and what properties from a network we're focusing on. In this document, we are trying to understand relations between entities, diffusion of information and influential actors within a network. Therefore we will focus our research on a social perspective.

It is curious how these criminal activities and groups living in the Dark Web are similar to "offline" criminal networks, which is an area from criminology that has a lot to offer in terms of mechanisms, knowledge and approaches to disrupt these networks [39]. As an "offline" criminal group, there have to be ties that link members inside a group and eventually form a network where data and goods are exchanged. There is a study published in 2014 by [24] that traces how important it is to study topics such as social network analysis that, through the years, have shown to be a useful tool to combat organised crime. This study tries to perceive why it is important to intervene on these criminal networks at earlier stages as these networks show a high capability to re-organise after being disrupted. It also shows that there are two main approaches to disrupt a criminal network: the social capital approach and the human capital approach. The first one targets strategic positions that specific nodes occupy on a network, and the second approach focuses more on how a specific node can bring value to the network. More interesting is the results they obtained, as applying different strategies to disrupt an illegal cannabis network led to the network become more efficient than before applying those techniques.

This study reinforces the idea that network analysis is a broad field that can be fully explored in the Dark Web context. Actually, the ability to modelling data that can be retrieved from the Dark Web makes the network analysis even more useful as we're surrounded by multiple technologies in our hands that offers libraries and packaged giving us the capability of retrieving data easily from this environment and analyse the information we get [37]. Therefore, we will now study a field of network science: social network analysis, which gives us the ability to have a complete vision of the network and eventually create graphs that will point out relevant characteristics from the network.

2.4.1 Social Network Analysis

As pointed out, social network analysis is a field of network science that gives the ability to analyse a graph showing the relationships within a network. On such graphs, links between two entities are designated as edges, and those entities are referred to as nodes or vertices. A node "o" with an edge to a node "d" is said to be adjacent to "d", and the inverse is also true. These nodes can be anything (persons, animals, diseases, food webs, computers, etc.). However, in this document, we will approach it as network infrastructures or as entities that model data relevant to this project, such as hyperlinks

from the Dark Web. For instance, in the Tor Project, this could be an example of how hyperlinks are found [39]. We saw Tor's hidden services hyperlink end in .onion and are auto-generated, giving it a random aspect and the most straightforward form to find .onion hidden services are by learning it from other websites. Through social network analysis, we can identify which websites helps the spread of hidden services by looking at how many outgoing hyperlinks it shows, revealing which websites are more popular and essential within the Tor network [39].

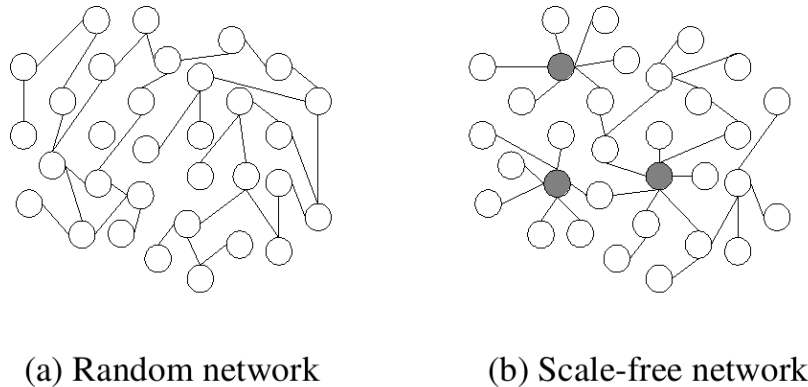


Figure 2.11: Comparison between a random network (A), where every node is dispersed in space, and a scale-free network (B) where some nodes have more connections to the rest of the network. Retrieved from [16]

Networks are usually structured in a random form or in a scale-free form. Random networks result from an organisation of nodes according to some probability resulting in a graph with low patterns on the connections between nodes.

If we look into the figure above, we can understand that under the situation depicted in A, a random attack can be more effective as there is a higher chance of targeting a central node or an important node in comparison with situation B, where we have a scale-free network that can be characterised with a power-law distribution as there are a few central nodes compared with the number of nodes of the network [24]. This makes these networks more resilient to random attacks. However, a targeted attack can cause more damage to those networks as there are nodes with a higher number of connections, such as the central nodes and eliminating those could disrupt some relevant links to other parts of the network [24]. Scale-free networks are extremely relevant in the World Wide Web context as they characterise the regular Internet and therefore can give an interesting insight into Tor's network [39]. As these networks have central nodes that own a large number of connections from the network, they have an important role in how information travels through the network, and there are models such as core-periphery structures that allow to identify and visualise these characteristics on the network, but we will get to that in a while [39].

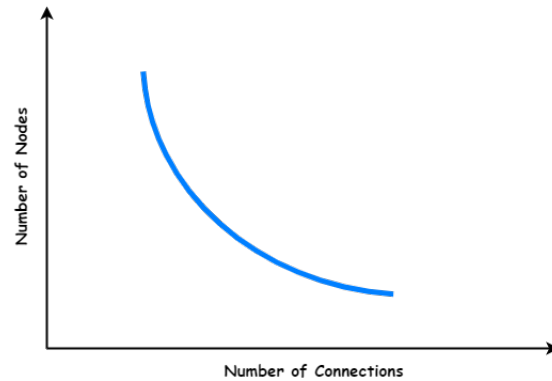


Figure 2.12: Power Law Distribution Representation

In the figure above, we represent one of the most prominent characteristics of the scale-free networks, a power-law distribution. This figure is just representative, and it does not show real data. A scale-free network has a curve similar to the one on the figure that translates what has been said before about these networks. Little nodes have a higher number of connections, while the majority of the nodes lives in the periphery.

To evaluate these attacks, it is crucial to understand mathematically what happens with the network before and after. Therefore, some measures can be calculated to understand how centralised a network is according to its degree¹⁷.

Centrality There are a few measures with different meaning that gives relevant information about a node on a network. To understand how are those measures calculated and what information we can get from them, we will consider the following example:

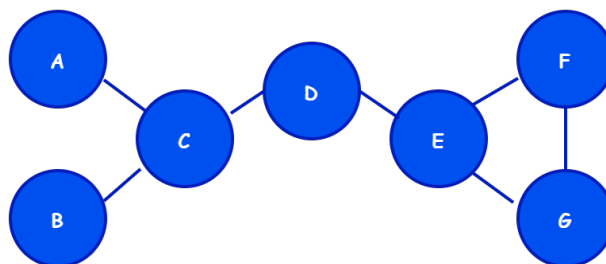


Figure 2.13: Example of a network graph with 7 nodes

Using the example above, we will explore some characteristics we can retrieve from this network that can be important to make some conclusions about the network and the role of its nodes. We will consider N as the number of nodes on the network and that every link has a cost of a unit.

¹⁷A degree of a node reveals how many edges are connected to that node.

- **Degree Centrality** - it is essentially the number of edges a node has. A high value means that the node has a higher role as a hub. A network with a zone full of nodes with a high value of degree centrality is called the core of the network. On the other hand, a zone where nodes have low centrality may be denominated as the periphery of the network [27]. Nevertheless, this measure does not give much information about how central a node is in the whole network or how important is it in interconnecting others [27].

To calculate the degree centrality score of a node, let's say for the node D, we must sum the distance between the node D and its direct neighbours and divide the result by N-1:

$$\frac{\sum_{n=C}^E Dist(D, n)}{N - 1} \quad (2.1)$$

In our example, the degree centrality score for node D is 2/6, which empirically means that for 6 links on the network 2 of them are linked with the node we are evaluating.

- **Closeness Centrality** - This measure tells with an average number how close is a node to all of the other nodes in the network, and the lower the value, the more central is a node on the network [27].

To calculate the closeness centrality score of a node, keeping with node D for comparison, we must sum the distance between the node D and all other nodes and divide the result by N-1:

$$\frac{\sum_{n=A}^F Dist(D, n)}{N - 1} \quad (2.2)$$

In our example, this would result in a value of 10/6 where 10 represents the sum of all the distances between D, and all of the other nodes and the denominator represents the number of the other nodes on the network.

- **Betweenness Centrality** - The betweenness centrality score of a node shows how important a node is to the shortest paths of the network [27]. This is the measure that is more complicated to calculate, but it is simple as calculating a fraction of all the shortest paths between a pair of nodes that passes through the node we are evaluating by the number of all the shortest paths between the pair of nodes [27]. If we repeat the process for every single pair of the network, we just need to sum the result, and we will get the score for the betweenness centrality of the node we're measuring.

So, keeping again the node D of our example to show how can we measure the betweenness centrality, we can proceed as follows:

1. In our example, we must consider 15 pairs: AB, AC, AE, AF, AG, BC, BE, BF, BG, CE, CF, CG, EF, EG, FG (assuming that the network is undirected¹⁸).
2. Calculate for each pair the number of shortest paths that goes through node D and divide them by all the shortest paths of the pair. Ex: for pair AF we would have 1/1 because there is only a single short path from node A that achieves node F and this path has to go through node D.
3. After calculating for each pair we would have somethings like (following the order of the pairs described in first step):

$$\frac{0}{1} + \frac{0}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{0}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} = 9$$

The fractions of 0/1 are associated with pairs where there is only one shortest path between their location, and it does not pass through node D. On the other hand, the fractions 1/1 are related to pairs of nodes such as AE, AF, AG, BE, BF, BG, CE, CF, CG that have only one shortest path between their nodes and that path includes node D.

This measure is very useful as it helps to understand which nodes are important in the flow of data from one part of the network to another [27]. If we want to describe this process with a mathematical formula we would achieve the following expression:

$$\sum_{o \neq D, d \neq D} \frac{Sp_{o,d}(D)}{Sp_{o,d}} \quad (2.3)$$

where $Sp_{o,d}(D)$ is the number of shortest paths between the pair o and d that goes through node D and $Sp_{o,d}$ is the number of all of the shortest paths from o to d.

- **Eigenvector Centrality** - The eigenvector centrality measures how popular is a node in the network according to its neighbours [27]. This measure is very used as it offers a unique perspective of a node's position and relevance on a network in a single number. However, its calculations are too exhaustive as we need to make iterative calculations using matrices. Nevertheless, this is one of the most important measures that we can get from a network as it gives a perspective on how a node is popular, taking into account its neighbours. There are network analysis software programs that calculate this value easily. Therefore it is not an extreme concern if we do not dive in-depth into how it is calculated [27]. An example that can help understand this measure is if we imagine a social network and compare a user that has 300 unpopular friends with a user that has 300 where some of them are popular persons, we would expect that the user with popular friends will get a higher eigenvector centrality score [27]. To show the power of this measure and why it is relevant, Google's search engine algorithm uses it to rank web pages. However,

¹⁸A undirected network is a network where a path between two nodes is bi-directional in contrast with directed networks where a path have a direction

it uses a variant of this measure named PageRank which on its core is based on the eigenvector centrality [27]. It gives good feedback on how important is a node to a network in general and not just to their neighbours as the degree centrality.

There are other measures such as the Katz centrality, PageRank centrality, Percolation centrality, among others. Yet, most of these other measures are variants of the ones we just observed.

With all of these measures, we can get confused about each one should we focus on or if we should calculate them all to have the most transparent view of the network, but actually, it will depend on what are we asking or in what are we searching for. For instance, if we want to find the most popular person in the network, we would choose the degree centrality as the most important metric. However, if we want to understand which node on a network can obtain information more efficiently, we could go with the closeness degree as it gives the node that is closest to the centre of the network.

Now we have a background on why social network analysis is important and which measures are relevant to analysing these networks, but a few aspects are important to understand. It is relevant to comprehend that the World Wide Web can be represented with a directed network graph of hyperlinks that link files all over the network. However, the example shown above was approached assuming that the network was undirected for the sake of simplicity in the calculations.

If we understand the measures explored, we can conclude that they give a perception of a single node on a network, eventually computing those measures to a group of nodes, we can build a better view of the network as a whole [27]. There are other measures that take into account all the nodes in the network, such as degree distribution, density, connectivity and centralisation, but all of these measures use the same concepts of the measures we studied above, so we will not look into detail at this moment for them [27]. Just for showing an example on how to use a measure to have a perspective of the whole network, if we would like to calculate the density of a network, we would have to perform the following calculation:

$$\frac{\text{No.of edges}}{N * (N - 1)} \quad (2.4)$$

This equation is basically a ratio between the number of edges existing in the network *No.of edges* with the number of possible edges that could exist in the network $N * (N - 1)$.

Now that we have the basic knowledge of which properties we can compute to help us have an analytical insight of the network, there are a few concepts that are important to refer to in terms of network analysis, such as homophily and web link analysis.

Homophily The homophily in network science studies the relation between actors enforcing that similar actors have a higher tendency of creating bonds comparing with nodes within the network with more differences [39]. This could be an important aspect to study within the Dark Web as the presence of homophily would indicate that after

finding a certain website serving a certain type of content, it would be easier to find other websites with similar purposes and therefore be a negative aspect for this network in the eyes of the criminals [39].

Web Link Analysis The link analysis studies the structure and the importance of the existence of hyperlinks within web pages. It states that having a hyperlink on a web page to another is a sort of recommendation from that web page to the user visit the other one [30]. The existence of this hyperlink within a web page can also be a signal that the pointed hyperlink might be a similar web page [30]. This approach can be used together with some special metrics such as PageRank or Hyperlink-Induced Topic Search (HITS) that are extremely important to rate web pages according to their importance in terms of content [30]. Therefore, this type of analysis can be important to some applications such as correlated web pages, predicting link usage and categorise web pages [30].

Another relevant property that can be discussed within the social network analysis is something designated as a small world network. A network is characterised as a small world network if every node can achieve any other node with 6 hops or less. This maximum number of hops is not exactly that accurate, but it supports the model as it shows that these networks have a short average shortest path length and high clustering [27].

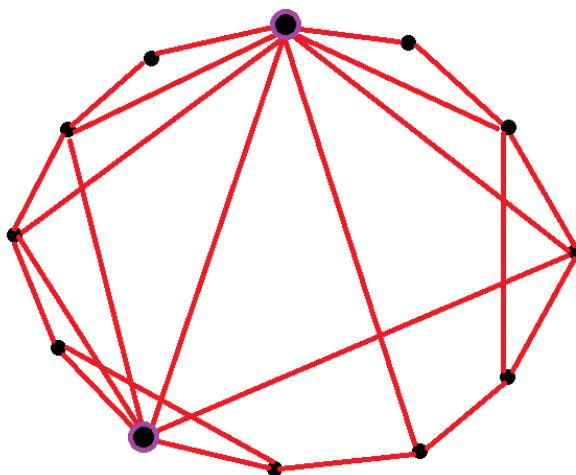


Figure 2.14: Example of a small world network. Retrieved from [51]

This is important as Darknets such as Tor have a topology that shows compatibilities with scale-free networks, and small words networks [22]. Nevertheless, it is worthwhile to understand how the network structures are characterised between domains, and for that, we will explore the concept of core-periphery structures [39].

2.4.2 Core-Periphery Structures

The core-periphery structures are a notion that resides within the social network analysis that offers a view of the core and periphery nodes of a network where we will find a strong and cohesive core against a sparse periphery [25]. This type of analysis can be effective in networks such as the Tor network that is a covert¹⁹ network because it gives a better perspective than the one we get from computing scores of the network or its nodes [39]. This analysis allows us to discover or identify nodes on a network that belongs to the core without recurring to the centrality measures because not all nodes with high centrality have to belong to the core. There are two possible models of the core-periphery structure that offers two different perspectives. The discrete model breaks the nodes into two partitions, whereas one is characterised by high interconnections (core), and the other partition is composed of periphery nodes. The continuous model gives the possibility of having more than two partitions and therefore having multiple cores [12]. It is fundamental to understand that this technique allows different implementations and perspectives of the model we choose to use. However, we will just explore its basics as the final objective is to find a core-periphery structure and apply the knowledge we have about those structures in real data, independently from the process we took to get there. To explore the discrete and continuous models, we will follow the example below that hopefully will explain how we can take advantage of each model.

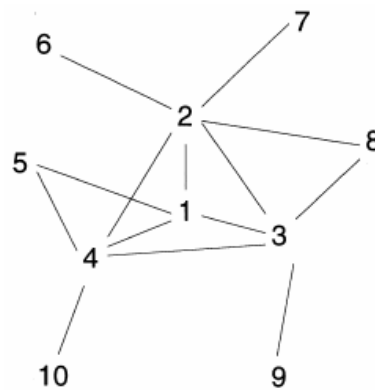


Figure 2.15: Example of a network with a core-periphery structure. Retrieved from [12].

Discrete Model This model looks into the network as a graph with two classes of nodes, the core, with nodes highly connected, and the periphery, with nodes that have fewer connections. In this model, a node can be a core or a periphery, and with that characteristic, it is possible to build a table showing the relation between the nodes in terms of core-periphery structure. To exemplify this, we can take a look at the figure below.

On this table, we have each node represented horizontally and vertically, showing the relation between them. If we look for the table above as a matrix A with index i and j

¹⁹A covert network has some characteristics such as security, secrecy and anonymity that are extremely important for criminal activities.

	1	2	3	4	5	6	7	8	9	10
1		1	1	1	1	0	0	0	0	0
2	1		1	1	0	1	1	1	0	0
3	1	1		1	0	0	0	1	1	0
4	1	1	1		1	0	0	0	0	1
5	1	0	0	1		0	0	0	0	0
6	0	1	0	0	0		0	0	0	0
7	0	1	0	0	0	0		0	0	0
8	0	1	1	0	0	0	0		0	0
9	0	0	1	0	0	0	0	0		0
10	0	0	0	1	0	0	0	0	0	

Table 2.1: Adjacency matrix resulting from the example in the figure 1.15 [12].

we can observe that $A_{ij} = 1$ if the nodes with the index i and j are adjacent to each other, otherwise they are classified as 0. Looking for example 1.15, we can easily build this adjacency matrix that will look just as table 1.1.

On this table, we can observe a pattern where core nodes are adjacent to other core nodes, and as we move further to the periphery, we observe that we start to have fewer and fewer connections. The ideal adjacency matrix is the one that supports the model and can be used to evaluate how well an adjacency matrix from a network relates to the idealised version that is essentially an adjacency matrix of a star network topology.

	1	2	3	4	5	6	7	8	9	10
1		1	1	1	1	1	1	1	1	1
2	1		1	1	1	1	1	1	1	1
3	1	1		1	1	1	1	1	1	1
4	1	1	1		1	1	1	1	1	1
5	1	1	1	1		0	0	0	0	0
6	1	1	1	1	0		0	0	0	0
7	1	1	1	1	0	0		0	0	0
8	1	1	1	1	0	0	0		0	0
9	1	1	1	1	0	0	0	0		0
10	1	1	1	1	0	0	0	0	0	

Table 2.2: Ideal adjacency matrix showing core-periphery structures [12].

On the table above, we can see a table that is split into 4 rectangles. The first one shows the core-core links, and here we have a cohesive core just like the model supports. This is represented in the table on the indexes between 1 and 4. Then, we have two core-periphery groups, one between 5 and 10 horizontally and 1 and 4 vertically, and the other between 5 and 10 vertically and 1 and 4 horizontally. On an ideal core-periphery structure, this proves that the periphery is fully connected to the core besides being on a sparse zone of the network as the periphery-periphery zone, forth rectangular, shows. On this last rectangle, we can see that the adjacency matrix is full of zeros showing that

there are no links within the periphery-periphery structures.

It is extremely relevant for the core-periphery analysis to have these idealised forms of a core-periphery structure to be able to compare and compute a quality function to understand how does a real network from the real world fits on these theoretically models in order for us to be able to identify these structures in noisy environments such as the one in table 1.1 [12]. Therefore it is crucial to define our algorithm to identify a core-periphery structure on a network either if we're using the discrete model or the continuous model which we're going to visit next.

Continuous Model The other model for the core-periphery structures is the continuous model, where we will not have just two partitions with the core and periphery nodes, but instead, we will have multiple partitions building multiple classes that will differ on its coreness property. The coreness property is a measure that represents the Euclidean distance from a central point.

This model is important as the discrete model is too simple. However, there is a problem associated with the continuous model that is related to its complexity, as increasing the number of partitions the systems starts to be difficult to handle [12]. To calculate the coreness of each node on a network we can use a software that is specialised for social network analysis named UCINET²⁰ using algorithms such as MINRES to find core-periphery structures [13].

Later on, in this document, we will explore more further this approach and understand how and why it is valuable for discovering important cores within a network and subgraphs. An interesting aspect we can retain from this approach is that the coreness score might sound like just another centrality measure, and that is true. However, the contrary is not necessarily true [12]. For instance, we can have some nodes within the core of the network that does not form the core of the network as there can be nodes highly connected with other nodes, but with few ties with nodes from the core, [12].

These models, either if we use the discrete model or the continuous model, are subject to our implementation and in how we decide to evaluate the coreness score of the nodes within the network if we're using the continuous model, how we identify the core-periphery structures on the network and how we correlate them with the base model. This is crucial as in empirical networks we often do not have what is known as ground truth²¹ [25]. An example of this can be seen in the research [20], where the authors apply a core-periphery analysis by using a continuous model which follows the behaviour of a random walk model. In a brief explanation of how they built the model, they first created a core-periphery profile for each node that provided a coreness value to each of them through a random walk model. After, they grouped all nodes that had a coreness below a

²⁰This is a software package with visualisation tools for social network analysis. More information can be obtained at <https://sites.google.com/site/ucinetsoftware/home>

²¹Capability to remove pieces of evidence by looking into a problem in contrast to finding it through studying it

certain threshold. With that computed information, they were able to understand which nodes are part of the periphery and which are closer to the core. However, this is much more complex than what is described here, but we just want to show how creative we can be in applying algorithms and models to approach the core-periphery structures.

2.4.2.1 Multiple Core-Periphery Pairs

Another interesting approach for the core-periphery structures is looking for multiple pairs of these on a single network. The study [34] developed an algorithm to identify these structures as the authors believe that empirical networks are closer to this approach than by just a single core-periphery structure, and it is not hard to understand. So the idea is that instead of looking for two groups of nodes that compose the core and periphery, we could look for multiple pairs of such groups. Now our adjacency matrix will have its core-periphery pattern multiple times, as we can see in the figure below (b). In this figure, the black colour means the same thing as the number “1” in table 1.1 and table 1.2, the existence of an edge.

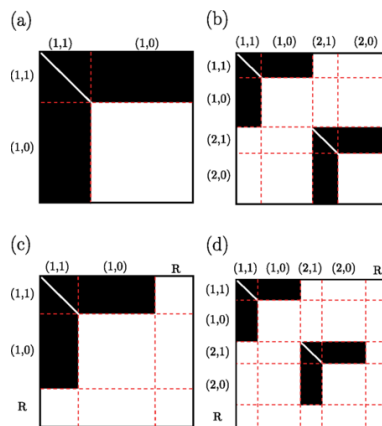


Figure 2.16: Comparison between a single core-periphery structure (a) and (c) with a multiple core-periphery structures (b) and (d), respectively. Retrieved from [34].

The work developed by the authors of [34] shows their algorithm to identify these multiple core-periphery structures in comparison with an algorithm developed by Borgatti-Everett in [12] and with a two-step algorithm which is based on the Borgatti-Everett solution but using a Louvain algorithm²². To achieve this, they used empirical networks as proof of case to test and compare the results. To briefly show their results and help clarify why is this useful, we will describe their results obtained over a karate club network.

The modelled network is based on a karate club from a university where adjacent members (members that share a link on the network) are members that have interacted socially outside the club activities. The network was created with 34 members and to study the relation between members after a conflict between an instructor represented by

²²The Louvain algorithm is a method to detect communities from a bigger network [47]

node 1 and the president represented by node 34. It is known *a priori* that 15 members are on the instructor's side, 16 on the president's side and 3 are neutral.

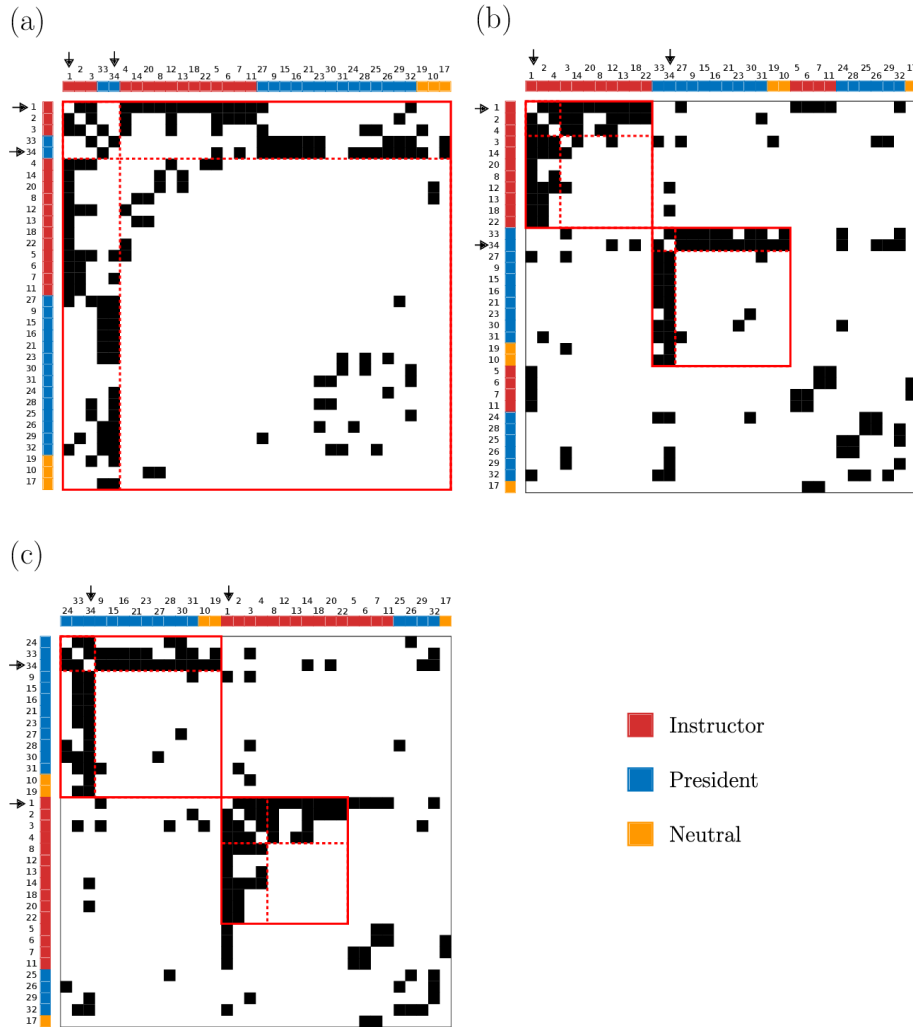


Figure 2.17: Core-periphery structure of the karate club network. On (a) it was used the Borgatti-Everett algorithm, (b) the two-step algorithm and in (c) the authors of [34] algorithm.

In the figure above (a), we have the core-periphery structure resulted using the Borgatti-Everett algorithm, where we can observe that it only detects a single core-periphery pair where the core is made of the instructor and president supporters. The structure shown in (b) was built using the two-steps algorithm, and this solution found two core-periphery pairs where the first pair core is made of instructor nodes and the core of the second pair is built of president nodes. Something interesting happens on this solution. If we look at nodes 10 and 19 they have both an edge with the president, and they are both in the president's core-periphery pair. However, they are both neutral. At last, we have the solution provided by the authors (c), where we can see two core-periphery structures showing a result similar to the one from the two-steps algorithm.

This multiple core-periphery approach is definitely worthy of further exploration as

a network is more likely to have a better representation with multiple core-periphery pairs, and by combining this with the right algorithm for core-periphery detection, we can build a strong tool for network analysis [34].

We have seen a classic type of analysis used in network science (social network analysis) and a model that is powerful by revealing a different perspective of networks (core-periphery model). This analysis focuses both on studying the structure of a network while revealing details about its topology and on nodes relevance. Nevertheless, there are other types of analysis relevant for studying the Dark Web that will be explored in this project. We are talking about technologies in the area of machine learning, such as neural networks or natural language processing. These technologies are important as they can parse data and help us find interesting patterns and characteristics from it.

2.4.3 Machine Learning

A research from 2018, [63], explains well the essence of machine learning as it is an area in constant evolution. The idea of the algorithms used in the area of machine learning is to somehow replicate the same process we, humans, do when we work to learn something. For that, we explore new information in time, and we try to find patterns in conclusions we observe from reading or living experiences. The difference is that computers can handle and do a huge amount of computations in a minute, and we cannot. Therefore, these algorithms have incredible results and make computers look really “smart” or “intelligent”. However, they are just able to do simple computations in fractions of seconds which makes them look as so.

Machine learning can be broken into four parts according to the authors of [63].

- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforced Learning

In **Supervised Learning**, we are able to learn from examples where we have an input and an output. This means that we have algorithms that use full examples (i.e. which input led to which output) to find patterns and learn. In this part of machine learning, we can be able to classify the data from a given number of possible categories (**classification**) or predict new categories (**regression**). Regression is used to predict a value, for example, the temperature in the next day.

For **Unsupervised Learning** we have a similar scenario with supervised learning. However, we don't have examples to learn from. This is useful when we want to learn without having any old knowledge available. It can be done by grouping data according to their similarity (**clustering**) or by taking into account the connections between the data

(association). **Semi-supervised Learning** can be seen as a particular case of supervised learning where we have a mix of examples to learn from which some have the output and others do not.

The last type of learning is the **Reinforcement Learning** where basically we learn by test and failure techniques. It is like having to go from A to B, and for that, we try all the possible paths till we learn which is the correct one.

In machine learning, we have a set of possible algorithms that can be used depending on the type of learning we're using from the 4 we just observed. We have Artificial Neural Networks, Decision Trees, Naive Bayes, Nearest Neighbours and so on. Nevertheless, from these algorithms, there is one that can be used in almost every type of learning, neural networks.

Neural Networks The neural networks are based on the biological neural network we have in our brains. As the name suggests, it is made of nodes called neurons that are a really simple element from these networks. Basically, we will have a large number of neurons forming a network that will learn something from a given dataset, designated training dataset, by finding patterns on the data. The "knowledge" is saved on weights, which is a value attached to every connection between neurons that is updated many times during the training process according to an optimisation algorithm that minimises the loss.

A neuron has 5 important properties such as the **input** of the neuron, the **weights** of the inputs, a **transfer function**, an **activation function** and finally the **output**.

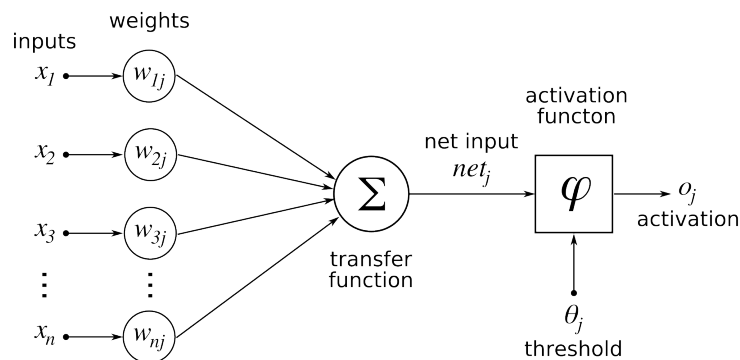


Figure 2.18: Neural Networks Neuron Model. Retrieved from [59].

A neuron can have multiple inputs, each one with a different weight that will be updated during the training process using an algorithm such as backward propagation of errors. Basically, when the data passes by a neuron, it will enter by the input, be multiplied by the weight of the input, then pass through a transfer function that will take all the inputs multiplied by the corresponding weight and do a computation according to the transfer function that usually just adds up all the values. After that, these added values will go through an activation function that can have different characterisations,

and its result will flow to the output, which can be the input of another neuron or the output of the neural network.

The idea here is that we can aggregate multiple neurons in parallel to create a layer and have multiple layers of neurons.

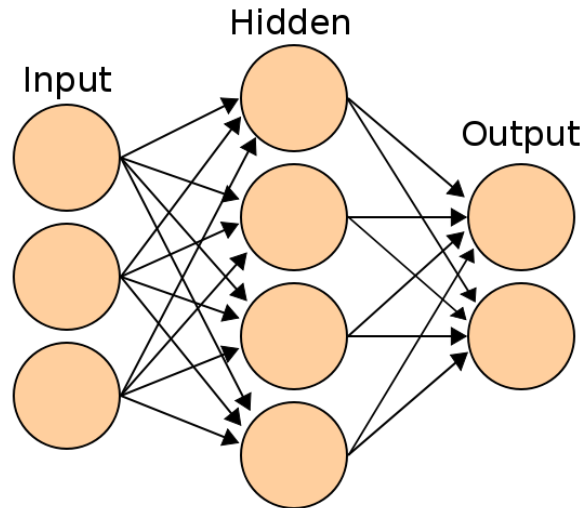


Figure 2.19: Neural Network Diagram with 3 layers. Retrieved from [15].

These are the basics of neural networks. By combining neurons in different orders, with different connections and different algorithms for the learning process, we can have distinct neural networks architectures that will be more appropriate than others depending on the problem. A neural network architecture with multiple hidden layers is described as a deep neural network. We can have different types of architectures, such as Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN), which are some of the most common architectures. They all work on the same idea we described before but use different algorithms and dimensions according to their application.

To evaluate the quality of a neural network model, we have some metrics that are important to understand. The most used ones are the accuracy and the F1 score, however, we also have the precision, loss, recall and others. They are critical because there are some concerns we have to take into account when training a neural network. One of them is overfitting, which means that the neural network model works well for the given training dataset, and when we present new data, it will start to fail. This problem happens because the neural network could not generalise the problem well, and it was only capable of learning the giving examples. It is like when we memorise something instead of truly understand it. To avoid this, it is important to analyse some metrics.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total predictions}} \quad (2.5)$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (2.6)$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (2.7)$$

$$\text{F1 Score} = 2 * \frac{1}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} \quad (2.8)$$

All of these metrics have their purpose, and they must all be considered because they alone are not enough to understand the quality of a neural network model. The accuracy, equation 2.5, is not enough if we have an unbalanced dataset. If we have a training dataset with two categories, true or false, and one of these has 98% of the examples while the other one has only 2% if the neural network model output is always the category with more examples, it will be right 98% of the times. Therefore we have metrics like the F1 score that avoid this effect by counting not with the total of the predictions but with the false positives and false negatives.

Machine learning opens a world of different analyses to solve a different set of problems. It allows the detection of patterns in data, classifying data, predicting outcomes, finding solutions, and so on. In reality, it has almost no limitations in terms of applications, and it is all due to its unique capability of finding patterns in data. Natural Language Processing is an example of applications of neural networks where we are able to make computers understand spoken languages and text almost like humans do and extract relevant information at a pace that humans can not do.

This leads us to the end of a long introduction chapter. We started by understanding why the Dark Web is such a sensitive topic, followed by a journey within the different software tools used to access it and the existing methods and research to disrupt them. It is more than the superstitious place we always hear on social media and the Internet. However, it is a place where dubious things happen, and therefore, it was essential to understand what is it made of and on what tools can we count to increase our knowledge about these covert networks. We are ending this chapter with a problem statement where we will present what we will build, how, and what we expect.

STATE OF THE ART

In this chapter, we will explore and travel through the most recent and relevant researches that were conducted regarding the exploration of data obtained from the Dark Web to increase the knowledge around this big system. Just like we saw in the second chapter, we are talking about a secret part of the World Wide Web that is protected with multiple technologies to hide information and anonymise users and web servers, and it is a big challenge to find a foolproof way to get a deeper insight from this environment [52]. The secrecy and difficulty of finding web pages on the Dark Web are completely different from the Surface Web. A study from 2017 concluded that 87% of the web pages in the Dark Web does not have links to other sites [28].

In the study, [52], regarding the challenges with gathering and analysing data, they describe a tool designated as BlackWidow that allows representing extracted data into a relevant graph providing tools such as machine learning to explore some patterns and unique characteristics of the network. Later on in this study, it is stated that web forums have been a big target from the research community as they can reveal unique information regarding user's preferences, user's relationships, and user's origin within the network. Therefore, we will start by exploring a little more on how we can retrieve relevant data from a web forum on the Dark Web.

3.1 Using Unsupervised Learning Algorithms on the Dark Web Forums

The following research from 2015, [45], focused on examining Dark Web forums revealing the importance of social network analysis techniques as tools to dig into hidden information that can be obtained through untreated data extracted from the Dark Web. The authors of this research aimed to use those techniques to identify crucial members within

Islamic Networks that can be found on Dark Web Forums.

Using metrics that the social network analysis offers, such as the ones we demonstrated in chapter 2 section 2.4.1 enabled them to better understand the actors of the forums they explored. After, they applied unsupervised learning techniques such as clustering to classify users within those forums.

3.1.1 Process

The process which the authors of [45] followed to achieve their final results are depicted by order below:

1. **Data retrieval**
2. **Build the social network model**
3. **Compute social metrics**
4. **Apply the unsupervised learning algorithm**

Data retrieval The first task was to extract data from the forums the authors chose to explore. In this research, the authors used a few Dark Web forums that were investigated by them previously.

Build the social network model After possessing the raw data from the Dark Web forums, they started to create the social network. To do that, they used an algorithm that they designate as “Algorithm 1”, where it essentially converts a forum into a social network. The algorithm can be summarised in:

1. Navigate through each thread in the forum
2. Within the thread, navigate through each message in the thread
3. Now inside the message, visit every previous sender within the thread and accumulate the weighted score of the message that takes into account the author of the message, the sender, and the message itself, also designated on the algorithm as $w(a, s, m)$ where m is the message which is being sent from sender a to the sender and s .

The sum of every weighted score of the message between the author and previous senders in the thread will accumulate on a variable designated as the overall weight of the directed edge between the author and every other previous sender, also represented as *SNW* (Social Network Weight).

Below there is a copy of the pseudo-code presented in the research to describe the Algorithm 1:

```

for each thread t in forum f do
  for each message m in t do
     $a = author(m)$ 
    for each previous_sender in t do
       $SNW(a, s) = SNW(a, s) + w(a, s, m)$ 
    end for each
  end for each
end for each

```

Compute social metrics This study used 12 metrics that were later split into five categories to help improve the characterisation of the nodes in the network. The list of metrics used can be seen below:

- **Sent Messages (SM)** The numbers of messages sent by a node
- **Received Messages (RM)** The numbers of messages received by a node
- **Degree Centrality Score (DCS)** The numbers of direct edges with a node
- **Betweenness Centrality Score (BCS)** Score to evaluate how important is a node in connecting the network (see chapter 2, section 2.4.1 for more details)
- **PageRank Score (PRS)** This score reveals how important is a node in the network according to its edges
- **Markov Ranking (MR)**
- **HITS Authority Score (HAS)** Authority score of a user that is higher if he receives messages from central nodes (hubs) with high weights
- **HITS Hub Score (HHS)** This score will be higher if the node has sent messages to nodes with high HAS
- **Clique Score (CS)** The clique score is defined as the number of cliques, based on maximal subgraphs. A clique is composed of n vertices of a network all connected through direct edges, creating, therefore, a subgraph
- **Weighted Clique Score (WCS)** The weight clique score is calculated based on the number of nodes within each clique (subgraph)
- **Average Distance Score (ADS)** This score is a similar version from the closeness centrality seen in the chapter 7, section 2.4.1, where it reveals how far is a node from all of the others on average
- **Clustering Coefficient (CC)** It reveals which nodes in a graph do a node tend to cluster with

With the metrics above, the authors created 5 categories that are highlighted on the following figure removed from the study:

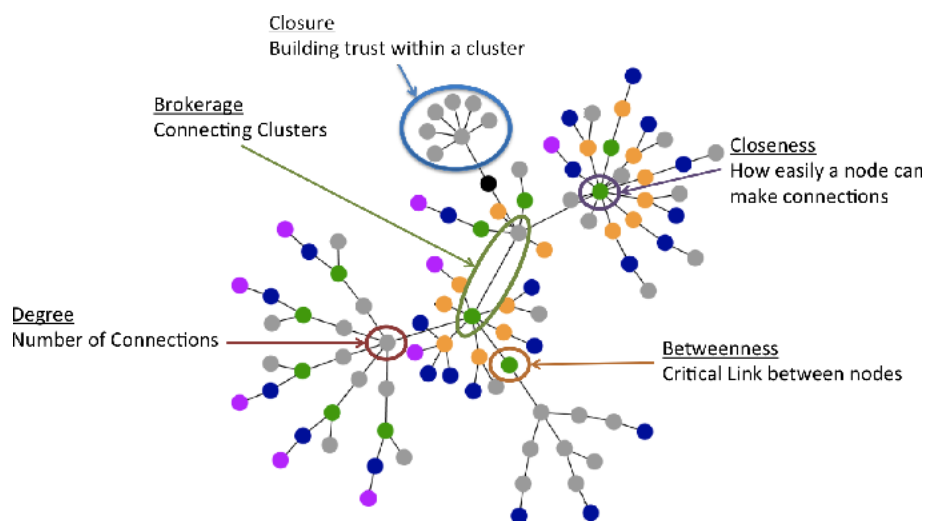


Figure 3.1: Network displaying the five categories created by the twelve metrics used in the study [45]. Retrieved from [45]

Apply the unsupervised learning algorithm In this part of the process, the authors used their dataset that had forums from radical Dark Web websites. With this dataset, they created four social networks using a combination of the following metrics:

- **Uniform weighting and unique senders**
- **Uniform weighting and all senders**
- **Inverse-Proportionality weighting and unique senders**
- **Inverse-Proportionality and all senders**

For instance, let us take as an example the forum “Ansar AlJihad Network”, which had 29492 messages, 11244 threads, and 382 members. The social network created by the authors for this forum using inverse proportionality weighting and all senders can be seen below.

With these social networks created and their info about each node in terms of metrics, the authors decided to apply the unsupervised learning algorithm based on expectation-maximisation to cluster information within the network. This algorithm, in a brief form, allows, given a dataset without labels, let’s say, for instance, without identifying the critical nodes on a network, to cluster seamless nodes by estimating and guessing values of the missing labels and update the algorithm parameters with those guesses and check if the system is converging into a cohesive outcome.

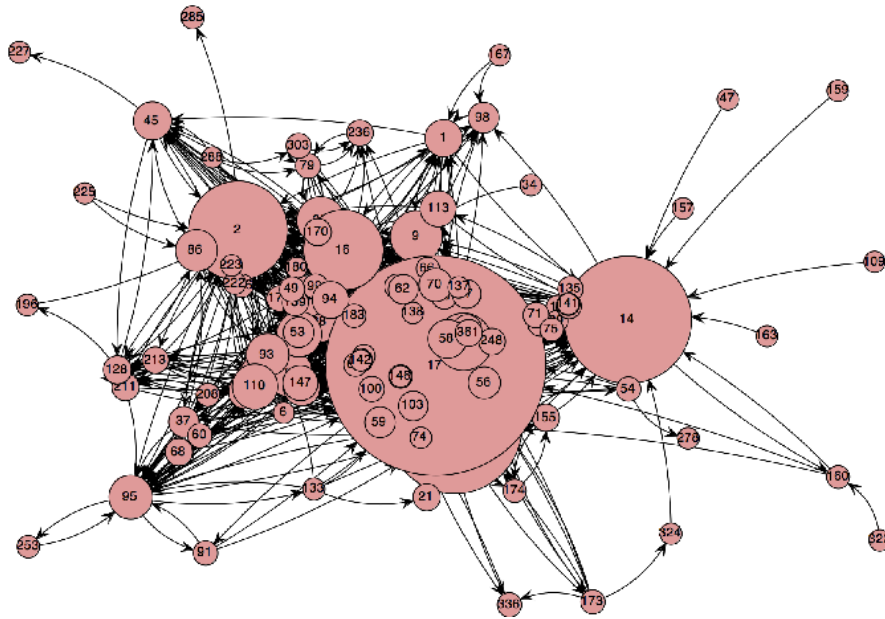


Figure 3.2: Network created from forum Ansar AlJihad Network where each node size shows its HITS hub score. Retrieved from [45].

3.1.2 Final Results

For the network shown in figure 3.2, the authors could identify five clusters from the Ansar AlJihad Network forum.

1. Cluster0 with 26 members
2. Cluster1 with 37 members
3. Cluster2 with 1 member
4. Cluster3 with 5 members
5. Cluster4 with 42 members

The metrics from all the clusters can be seen in the following table.

These results showed some relevant information, such as if we look into cluster2, we can observe that it has higher scores in almost every metric, making this unique member that is part of the cluster an important node on this network.

This study has other details that are not our focus here as we just want to understand what technologies they used, from where they started, and where they reached. From the research, we can observe that this technique is extremely interesting as most of the time, we can't label every member on a web forum making the use of an unsupervised machine learning algorithm the perfect fit to solve this problem as using important metrics such as the ones used to be able to understand which users are the most relevant on the network.

Attribute	Cluster				
	0	1	2	3	4
degree					
mean	0.0953	0.0095	0.8559	0.3568	0.0293
std.dev	0.0419	0.0021	0.1104	0.0944	0.0124
betweenness					
mean	30.5428	0.0535	5549.8267	551.7925	0.0002
std.dev	42.8349	0.3254	541.1433	337.2082	0.013
PageRank					
mean	0.0105	0.0031	0.1488	0.0464	0.0053
std.dev	0.005	0.0007	0.017	0.0223	0.002
Markov					
mean	0.0113	0.002	0.1513	0.0538	0.0048
std.dev	0.0064	0.0009	0.0185	0.0258	0.0027
HTTS_authority					
mean	0.0977	0.0168	0.4142	0.2655	0.0477
std.dev	0.0451	0.0118	0.0715	0.0601	0.0273
HTTS_hub					
mean	0.1085	0.0149	0.4118	0.2502	0.0445
std.dev	0.0423	0.0126	0.0712	0.0506	0.0291
weighted_cliques					
mean	538.4462	2.107	5108	3875.3263	17.6769
std.dev	666.9511	0.458	997.1299	669.2559	28.3311
cliques					
mean	5.968	1.0532	110	42.0026	1
std.dev	4.5005	0.2258	13.7192	14.3096	13.7192
avgDistance					
mean	1.5016	1.4613	1.8868	1.5895	1.474
std.dev	0.027	0.0414	0.0586	0.0342	0.0306
clusteringCoeff					
mean	0.6634	0	0.0589	0.257	1
std.dev	0.1631	0.0029	0.4387	0.0935	0.4387

Table 3.1: Table showing the metrics from the clusters identified on the forum Ansar AlJihad Network [45].

As the web forums are indeed an important case of study within the web and specifically within the Dark Web, we will now see another research made to, instead of identifying the most important nodes on a network, identify and predict possible future crimes by analysing web forums content using a supervised learning algorithm, in contrast with the unsupervised learning algorithm used on the study we just explored.

3.2 Predicting cyberattacks within Dark Web forums

A study from 2018, [50], reveals that using supervised learning algorithms within web forums, it is possible to predict cyberattacks. The authors used information from Dark Web forums and built an artificial neural network on top of a social network to understand whether a specific day could be a target for a potential cyberattack or not. The data used was from 53 forums with content over one year.

3.2.1 Process

The process which the authors of [50] followed to achieve their final result are depicted by order below:

1. Datasets
2. Build the social network model

3. Highlight and generate dynamics from the network

4. Use a supervised learning algorithm

Datasets As the objective of this research was to use an artificial neural network with a supervised learning algorithm to predict cyberattacks, the authors wanted to have data that would work as a ground of truth to allow them later on, evaluate their model. For that, they used a dataset retrieved from the Intelligence Advanced Research Projects Activity Cyberattack Automated Unconventional Sensor Environment program¹ where it had examples of cyberattacks labelled with attributes such as “event type” and “event occurred date”. It is essential to mention that the cyberattacks used as a reference in this study are the ones related to malicious emails and endpoint malware. After having this attributed data to use later on, for the authors empirically correlate results, they used an application programming interface (API) from Cyber Reconnaissance² that had information about multiple forums within the Dark Web. From this API they were able to get 179 forums, however, some of them were short, so they filtered the forums for the ones with more than 5000 posts leaving them with a fair number of 53 forums to create their model.

Build the social network model To successfully use an artificial neural network, one of the first steps is to define which inputs are relevant to integrate within the network. Therefore, the authors had to explore the real data obtained and find relevant information within that data to use as inputs for the neural network. For that, the authors created temporal networks from which they described features to use on the predicting algorithm.

To create the temporal network, they connected the users that have directly communicated between each other (i.e. that have, for instance, a reply between them) while having some temporal constraints to compensate for the absence of some information regarding whom someone replied on a specific forum. Then the authors also describe an operation of “merge” that allows using two networks to create a time series description for a specific feature. This operation is extremely relevant as it can show the network at different points in time. In the following figure, we can observe a merge operation with a historical network G_{H_T} of the network at certain times t_1, t_2, \dots

Highlight and generate dynamics from the network In this step, the authors relied on the networks created to extract features that could be relevant for the prediction algorithm. The features used on the predicting model can be split into the following three groups:

1. **Expert-centric Features** The term *experts* is used for a set of users who have been mentioned on their posts about security vulnerabilities and that have some relevant

¹Organization focused on investing in innovative research programs. More information at <https://www.iarpa.gov/index.php/research-programs/cause>

²More information at www.cyr3con.ai

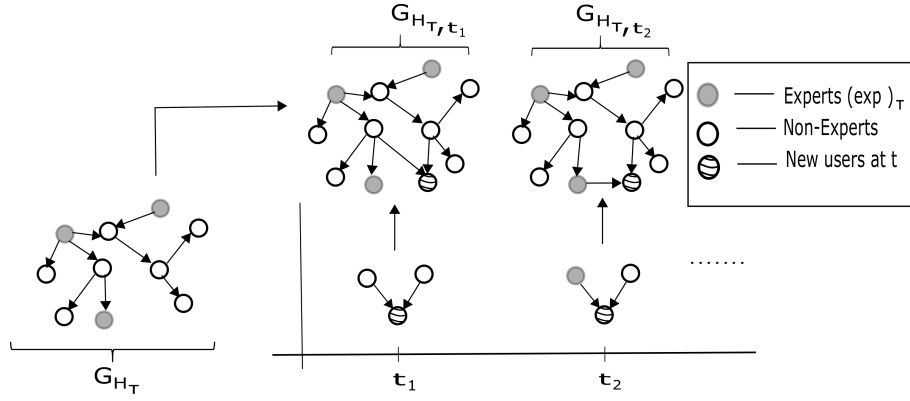


Figure 3.3: Merge operation of the network with important nodes highlighted in grey. Retrieved from [50].

quantity of reply messages. These features highlight users with more discussions about security vulnerabilities and words related to cyberattacks.

2. **User/forum statistics** The features related to users or forums statistics might give an insight into future cyberattacks making their exploration relevant.
3. **Network Centralities** Network centralities are related to the use of metrics such as the ones seen in chapter 2, section 2.4.1. These metrics can give an increased knowledge of the network itself and its nodes.

In the table below, we can observe in detail each of the features of the groups listed above.

Group	Features	Description
Expert centric	Graph Conductance	$\tau_x[t] = \frac{\sum_{x \in \text{exp}_T} \sum_{y \in V_t \setminus \text{exp}_T} \pi(\text{exp}_T) P_{xy}}{\pi(\text{exp}_T)}$ where $\pi(\cdot)$ is the stationary distribution of the network $G_{H_T, t}$, P_{xy} denotes the probability of random walk from vertices x to y . The conductance represents the probability of taking a random walk from any of the <i>experts</i> to one of the users in $V_t \setminus \text{exp}_T$, normalized by the probability weight of being on an expert.
	Shortest Path	$\tau_x[t] = \frac{1}{ \text{exp}_T } \sum_{e \in \text{exp}_T} \min_{u \in V_t \setminus \text{exp}_T} s_{e,u}$ where $s_{e,u}$ denotes the shortest path from an expert e to user u following the direction of edges.
	Expert replies	$\tau_x[t] = \frac{1}{ \text{exp}_T } \sum_{e \in \text{exp}_T} \text{OutNeighbors}(e) $ where $\text{OutNeighbors}(\cdot)$ denotes the out neighbors of user in the network $G_{H_T, t}$.
	Common Communities	$\tau_x[t] = \{ \mathcal{N}(c(u)) \mid c(u) \in C_{\text{experts}} \wedge u \in V_t \setminus \text{exp}_T \}$ where $c(u)$ denotes the community index of user u , C_{experts} that of the experts and $\mathcal{N}(\cdot)$ denotes a counting function. It counts the number of users who share communities with experts.
Forum/User Statistics	Number of threads	$\tau_x[t] = \{h \mid \text{thread } h \text{ was posted on } t\} $
	Number of users	$\tau_x[t] = \{u \mid \text{user } u \text{ posted on } t\} $
	Number of expert threads	$\tau_x[t] = \{h \mid \text{thread } h \text{ was posted on } t \text{ by users } u \in \text{experts}\} $
	Number of CVE mentions	$\tau_x[t] = \{ \text{CVE} \mid \text{CVE was mentioned in some post on } t \} $
Network Centralities	Outdegree_k	$\tau_x[t] = \text{Average value of top } k \text{ users, by outdegree on } t$
	$\text{Outdegree}_k \text{ CVE}$	$\tau_x[t] = \text{Average value of top } k \text{ users with more than } 1 \text{ CVE mention in their posts, by outdegree on } t$
	Pagerank_k	$\tau_x[t] = \text{Average value of top } k \text{ users, by Pagerank on } t$
	$\text{Pagerank}_k \text{ CVE}$	$\tau_x[t] = \text{Average value of top } k \text{ users with more than } 1 \text{ CVE mention in their posts, by pagerank on } t$
	Betweenness_k	$\tau_x[t] = \text{Average value of top } k \text{ users, by Betweenness on } t$
	$\text{Betweenness}_k \text{ CVE}$	$\tau_x[t] = \text{Average value of top } k \text{ users with more than } 1 \text{ CVE mention in their posts, by betweenness on } t$

Table 3.2: Table showing the groups of features and their description [50].

Use a supervised learning algorithm The objective of the prediction algorithm is to tell if at a point in time t there will be a possible attack. This was achieved by having a binary output from the prediction algorithm where 1 means an attack and 0 is the

inverse. For that, the authors decided to use the logit model, a statistical model that uses regression analysis, over Markov models and recurrent neural networks. It allowed them to adjust the data transparently as there were some concerns over the absence of data and its sparsity.

So before training and testing the performance of the model, they had to prepare the data they were using. For instance, they calculated the feature values for every day within their dataset and incrementally considered a period of 1 month to compute the time series values. To evaluate the algorithm and the model obtained, they cut every time frame into 80 per cent for the training dataset and the rest for the test dataset without shuffling the data to allow it to have some chronological sense. By analysing the data retrieved from the ground truth, they observed that the level of the cyberattack was different for each point in time, so they decided to use for the malicious email cyberattack and for the endpoint malware cyberattack the periods that showed to have more crimes to use as train and test datasets for their learning algorithm.

3.2.2 Final Results

The study's authors revealed interesting results from exploring the described features and by especially analysing the data on crucial time frames. In the image below, we can observe the results obtained from the model created by the authors. It reveals on the left side the results for the malicious email attack for the different feature groups and the same on the right of the figure for the endpoint malware attack.

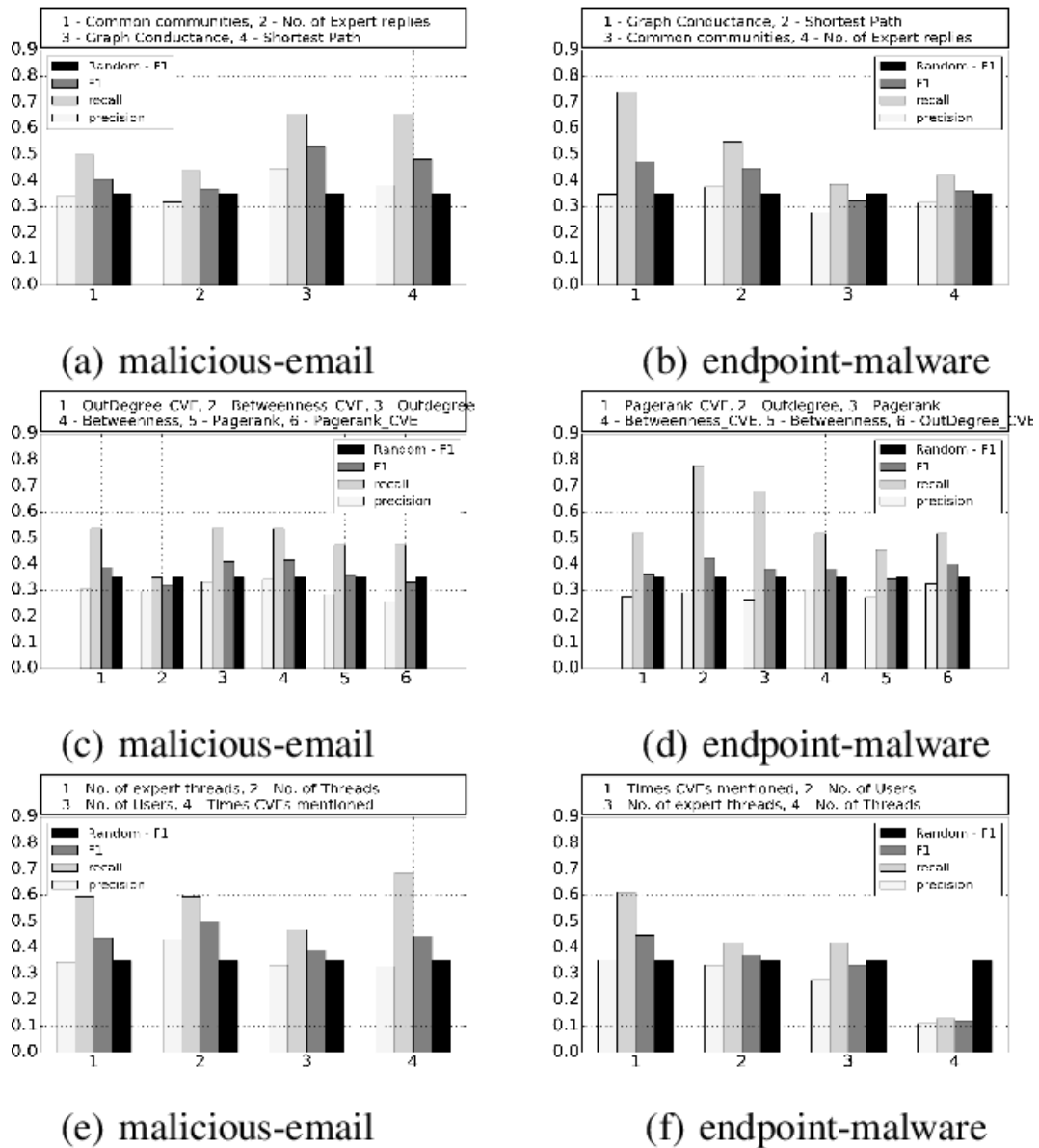


Figure 3.4: Results obtained from the learning algorithm in 1 week time window. Retrieved from [50].

On the figure above, each graph have a legend with machine learning scores such as F1 (a score that reveals the accuracy), recall (a score that indicated how much can we trust on the model in terms of true positives), and precision (reveals how much correct scores the model achieved). This study allowed us to understand which features are more relevant within the different types of attacks and the different feature groups.

The authors observed which weeks had more activity in terms of cyberattacks on their datasets and decided to evaluate their model on those specific weeks for the malicious email attacks. The authors decided to try something that revealed interesting results that were not unexpected, but that helped prove that using machine learning within the Dark Web can get good results. The result can be seen below on a figure that displays the same

information as figure 3.4 with an added legend for comparing the accuracy obtained with the accuracy from the last example designated as “Prior - F1”.

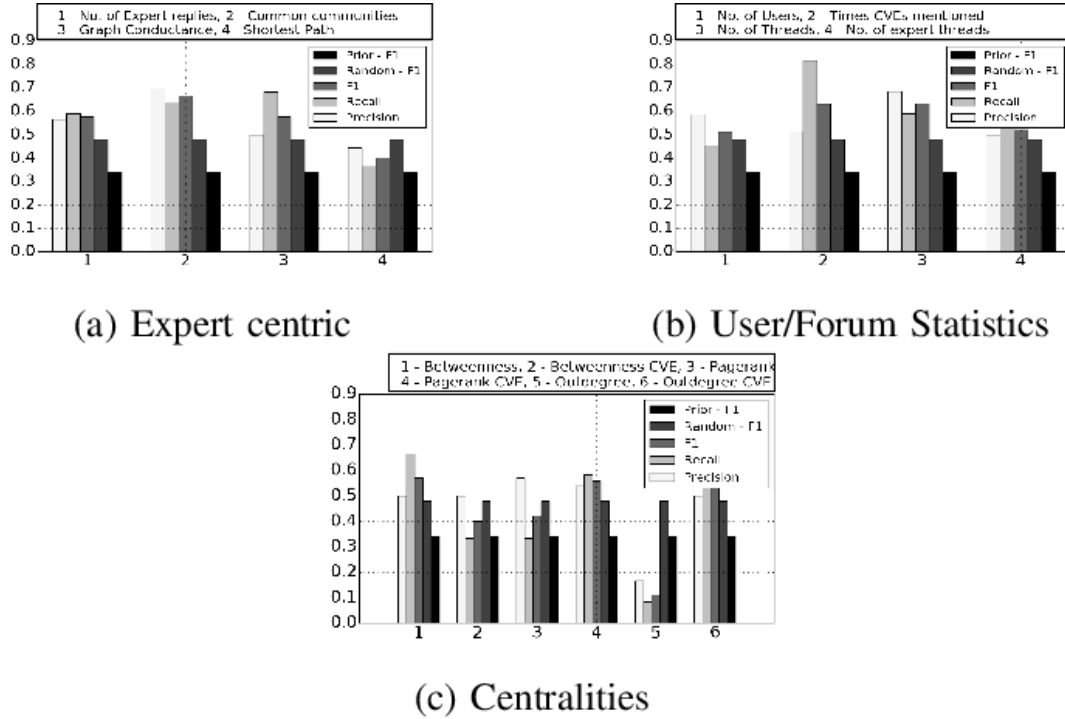


Figure 3.5: Results obtained from predicting malicious email attacks within the data range with more cyberattacks. Retrieved from [50].

It is interesting that on these last results, we can observe that the new accuracy is higher than the accuracy obtained in 3.4 (a), (c), (e), which is not a surprise as this time window was selected taking into account the number of cyberattacks, however, it reveals that the algorithm allowed to create a model sensible to detect this type of threats.

With these two researches, we have seen a first approach in section 3.1 where the authors of [45] used an unsupervised learning algorithm to detect important and relevant nodes within data extracted from forums residing in the Dark Web. We have seen a second approach where the authors of [50] decided to go with a supervised learning algorithm to predict two types of cyberattacks (malicious emails and endpoint malware) within criminal Dark Web forums. These two state-of-the-art works show that we have access to a lot of technology, models and algorithms, that only require us to be creative with the raw data we have to create valuable data and valuable information. We could keep briefly describing more researches around Dark Web forums because it is an active point of research within the Dark Web and there are much more interesting works such as [44] and [49]. However, they have something in similarity, they use algorithms to cluster or predict some raw data and with the enhanced data extract relevant information using social network analysis.

In the next section, we will take a look at a different work in the same context that looks not to Dark Web forums but web pages and hyperlinks extracted from the Dark Web.

3.3 Building and Analyzing Tor web graph

This research from 2019, [9], tries to study the topology of Tor hidden services and understand the relationship between content and actors of the network. They reveal how they collected data, which metrics they used for evaluating the data, and then created a graph to analyse the topology, and together with semantics techniques, they could get some valuable insight from the network.

Before describing their process, the authors defined some notes of what they are expecting and what they will focus on while assessing the research.

- They expect the Tor network to be oriented by topics, with most of the hidden services focused on a specific topic and only a few forming hubs that would represent marketplaces, forums, and wikis.
- The authors also expect to find many pages with just a few hyperlinks as they state that hidden services are mostly managed by single entities.
- Nodes on the network that are characterised as hubs for spreading other web pages within the network are more likely to be found on the surface Web, and therefore, they expect these hubs to be important to be used on crawlers to index other hidden services.
- They also expect the topology to be close to a forest graph resembling a spanning tree where it starts from the seeds of the crawler (we will explain later on what they define as the seeds of the crawler).
- The last note from the authors is that the hidden services serving dubious activities will be the hardest ones to find and access.

In terms of measures on this research, they used the following metrics:

- **Degree distribution** This distribution gives a statistical overview of the network and is one of the most used characteristics to analyse these networks [9]. Some measures within this distribution are the in-degree and out-degree, where the first reveals how many edges point to a node and the last one reveals how many edges are leaving a node [9].
- **Centrality** The centrality measures are the usual ones for doing social network analysis, such as the one that can be seen in chapter 2 in section 2.4.1.

- **Connected components** This metric allows studying subgraphs that are composed of vertices connected between each other. It can be classified as weakly connected components and strongly connected components. If a node is connected to another using an edge that has only one direction, it will be classified as weakly connected components. On the other hand, if the edge is bi-directional, we will have strongly connected components.

Apart from the metrics used, they also relied their study on an automated semantic engine that basically can be fed with unformatted text to extract meaningful information and map it into predetermined categories.

The authors also mention that they just want to focus their semantic analysis on two forms. The first will be on web pages that reveal to have highest in-degree, highest out-degree, or the highest betweenness centrality score. The second form will be to analyse whether the relation between contents and connections reveals the web pages to be similar in topic, a form of homophily analysis. The engine to perform the semantic analysis chosen by the authors was the Cogito³ semantic engine that is a specialised software that offers some intelligence features in terms of text mining, fact mining, categorisation, etc. The only disadvantage the authors depict is that they will have to filter out all non-English words as they used 17 predetermined categories to apply on the Cogito software that only considers English.

3.3.1 Process

Preparing the Datasets The data used in this study was crawled using BUBiNG crawler as, according to the author, it performs very well while being scalable and open-source. The data gathered from the authors were focused on onion links, leaving the possibility to also study links from the Surface Web to find references to onion links. In a summarised form, the authors describe the operation of the crawlers as follows:

1. They fed a predetermined set of hidden services that will act as the seeds of the crawler.
2. Then they extract the first onion link from the list and analyse whether its content reveals other hyperlinks to visit.
3. The newly found hyperlinks are verified to check if they were not visited already before.
4. Then it is also checked if those links are available or not by giving three attempts of establishing a connection with them

³Specialised software offering big data solutions to retrieve useful information from raw data. More information at <https://www.expert.ai/>

The process above shows how from a small number of hyperlinks, it was possible to gather much more links within the Dark Web. We, as readers, could ask if the results would depend on the seeds of the crawler, in other words, we could doubt if having different predetermined hyperlinks to start the process would return a different outcome, however, the authors state that if the web graph is strongly connected that would not be true.

The crawler, in the end, returned 1119048 records (onion addresses), but only 824324 were active as the rest of them returned with HTTP status codes such as 3xx, 4xx and 5xx.

Interestingly, the authors of this study tried to create a random algorithm to crawl onion addresses by generating random hashes (see chapter 2 section 2.4.2 for more information about how onion addresses are generated). However, even though they let it run for roughly one month, they found 0 active onion addresses, which left them unsatisfied. Nevertheless, they could conclude that a brute-force search for onion addresses is unfeasible. Therefore tools such as the BUbiNG crawler are important as they can return a better outcome. An important aspect is that, as described before, it needs a seed, a starting point of onion addresses, so it can learn more and more by navigating through those hyperlinks. In this study, the authors reveal that they fed the BUbiNG root set hyperlinks with hidden services found “by hand” that were mostly wikis web sites and link directories that are like a Google search engine but for the Dark Web (however not so powerful as Google).

Building network graphs With the data obtained, the authors of this study created three different graphs:

- **Page Graph (PG)** A page graph is a network that shows the relations between the visited web pages. A vertex is a web page, and an edge is a hyperlink to other web pages.
- **Host Graph (HG)** This graph will be constructed using the relation between Tor hosts. This is, vertices are domains and sub domains, and edges will be added between hosts if they have a hyperlink from their web page to a web page in another host.
- **Service Graph (SG)** The last graph will be constructed using hidden services as vertices and edges between hidden services if their pages have a hyperlink pointing to the other.

So to help clarify the different graphs, the author described an example.

The page graph will take every single page the crawler visited, create a node for them and add an edge to another page if there is a hyperlink for it.

The host graph, on the other hand, will now only look at different hosts. This is, if we have a domain *blackmambamarket.syndiccjacy*****.onion/login* we will only have edges from *blackmambamarket.syndiccjacy*****.onion* domain to other domains that have a hyperlink from a page belonging to their host that points to a page on this host.

Last but not least, we have the service graph that summarises all of the information on the network and creates graphs that simply link Tor's hidden services and not their single web pages. So we are observing nodes that represent a hidden service that offers many web pages as just a single node within the graph.

3.3.2 Final Results

The authors show their results according to the metrics they used and the semantic analysis that allowed them to build those network graphs we have just seen.

Degree Distribution In the following figure, we have two graphs that show the results obtained using the three types of graphs for the degree distribution metric. The one on the left shows the in-degree score for a fraction of nodes, and the one on the right does the same for the out-degree score.

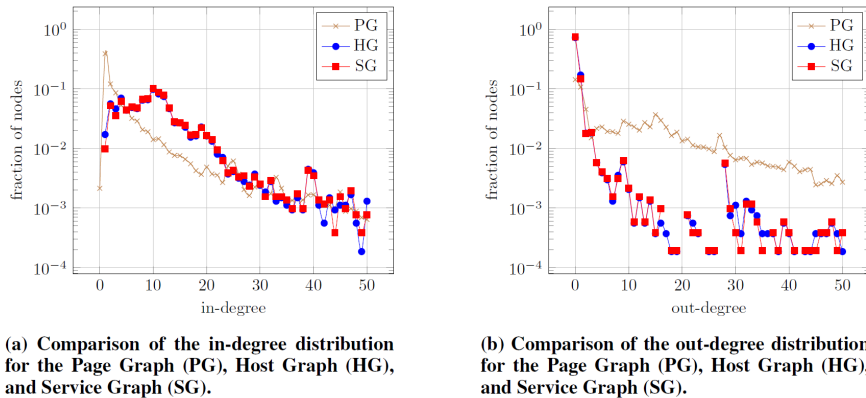


Figure 3.6: Graphs showing degree distribution scores for each type of network graph. Retrieved from [9].

The figures above prove some of the concepts that the authors were expected, such as Tor pages are difficult to find. If we look into figure 3.6 (b), we can observe that only a small fraction of nodes have a high out-degree score. It is also interesting that the HG and SG are almost identical in comparison with PG.

Centrality Then, following the introduction of what metrics the authors used, they created some figures to show how betweenness centrality (see chapter 2, section 2.4.1 for more details) could give a different insight on the centrality of a node. In the figure below, we will have a comparison on the right for the undirected graph with the directed graph where edges have a direction in the left.

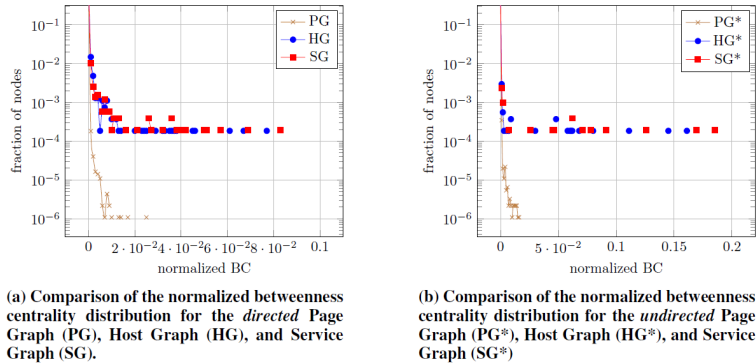


Figure 3.7: Graphs showing betweenness centrality scores for each type of network graph. Retrieved from [9].

Both of the graphs show that there are a few central nodes (hubs) with a bigger periphery. An interesting correlation the authors did was to check the top 10 nodes in terms of betweenness centrality and degree score, and they found that the nodes with higher betweenness centrality are not the ones with the bigger degree, which can be understood as the betweenness centrality measure how important is a node connecting parts of the network that are separated. Another interesting aspect is that the nodes with higher degree scores were some of the ones used on the root set for the crawler that was mainly wikis and link directories, which also supports one of the expectations in terms of results for the authors.

Connected Components Now, the last metric missing is the connected components, and for that, the authors focused on the size and the number of connected components for the three different types of graphs under study. For that, the authors did something interesting, as they removed the nodes incrementally with larger betweenness centrality and larger degree distribution score from the Page Graph with an undirected network. The consequence of this incremental removal can be seen in the following figure.

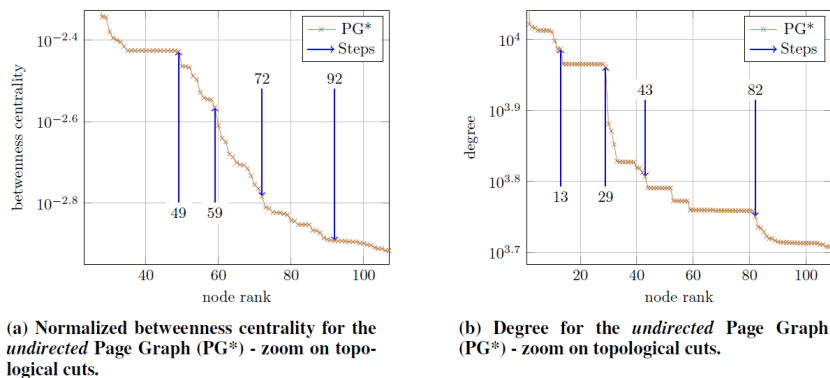


Figure 3.8: Graphs showing the betweenness centrality (a) and degree (b) score upon removal of important nodes on Page Graph undirected network. Retrieved from [9].

In figure 3.8, we can observe on the left graph some drastic cuts at node rank 49, 59, 72, and 92, and on the right side some cuts on the nodes 13, 29, 43, and 82. With this information, they built a table showing the connected components after those cuts observed. This table shows that the number of connected components rises rapidly, but the size decreases as expected because we are removing important nodes within the network, breaking them into smaller subgraphs.

Cut (BC)	# CC	# SRCC	Max Size	Cut (Degree)	# CC	# SRCC	Max Size
49	462	32	876348	13	13433	5	903680
59	4792	34	872008	29	14464	6	902323
72	4874	41	860346	43	16055	9	895265
92	9672	73	851268	82	16902	10	894294
500	41966	444	685351	500	43023	53	847006
1000	58646	861	550912	1000	48287	66	835470
5000	137732	1395	322329	5000	86071	343	727876
10000	171310	1553	261276	10000	115122	536	652547
50000	307000	1785	70993	50000	265580	3696	261239
100000	390179	2137	12127	100000	396926	5866	103085

Table 3.3: Table revealing the connected components after the cuts in figure 3.8 [9].

The SRCC column gives the number of *Semantically Relevant Connected Components*, in other words, the number of components that have more than 2 web pages.

Semantic Analysis To make the semantic analysis, the authors used the Cogito engine, as explained before, to quantifies how content from a page can be related to a predetermined category. To compare information between web pages, the authors used the cosine similarity, which is a measure of similarity, that compares two vectors, and the angle between them will translate into a similarity measure. If the result is 0 the two vectors are orthogonal, and therefore the pages are not similar in any aspect. On the other hand, if the result is 1 then the two vectors coincide, and the two pages can be classified as completely similar.

To apply this analysis, they only used the undirected graph PG, and with that, they created a statistical distribution of the cosine similarity between two vectors of the Page Graph.

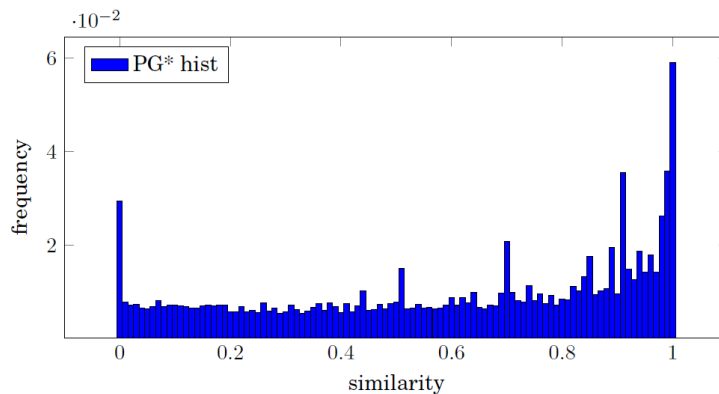


Figure 3.9: Statistical distribution of the cosine similarity in Page Grap. Retrieved from [9].

In figure 3.9, we can observe that the data is unbalanced and either the similarity is high or either it is really low with more frequency, which also supports one of the expectations about the network in terms of being more topic-oriented and that only a small part is related with wiki services, link directories, forums, and marketplaces. Therefore the next obvious step from the authors was to instead of mixing everything into a statistical distribution to watch it separately into the different 17 categories used on the semantic engine. The result was the following long figure.

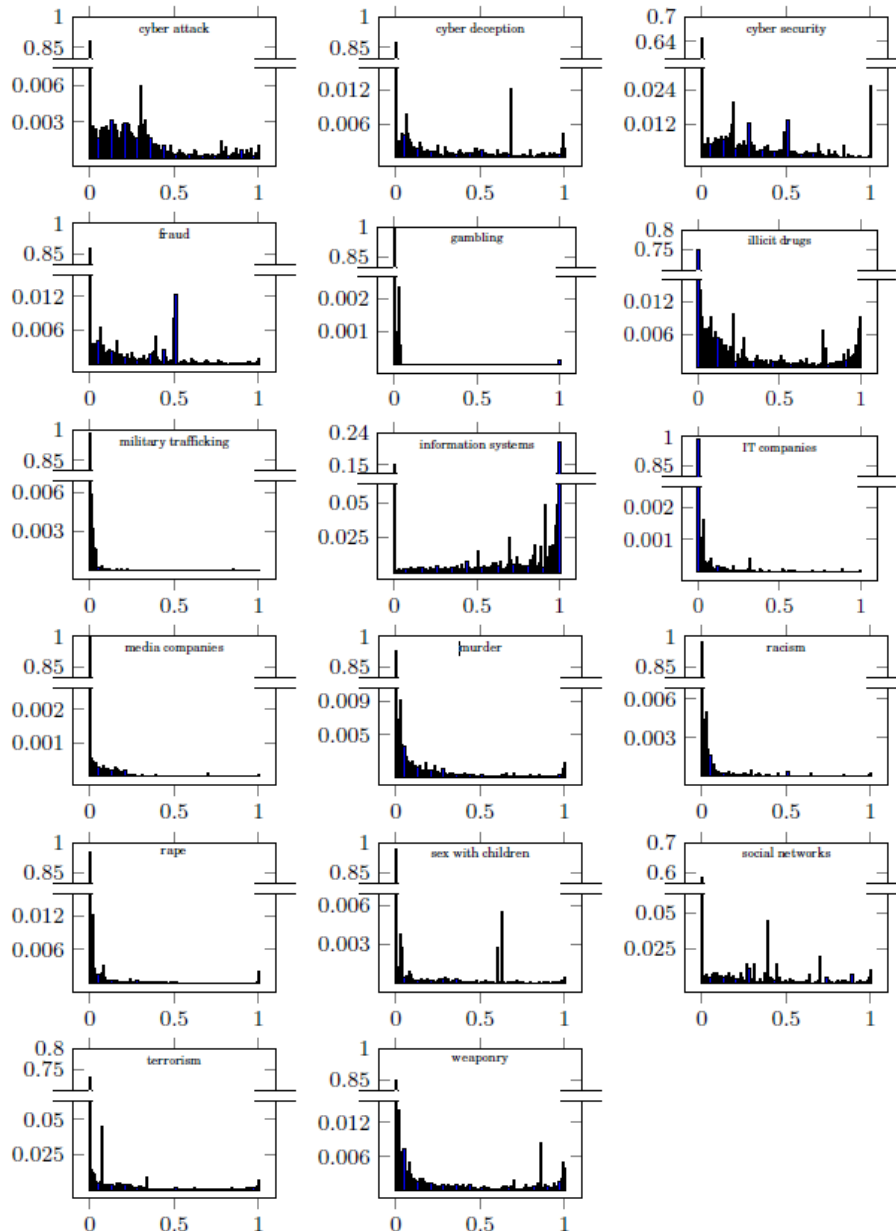


Figure 3.10: Statistical distribution of the cosine similarity for every single category studied. Retrieved from [9].

What highlights from all of these charts is that most of the categories do not have a spike at cosine similarity of one, which might suggest that the categories were too

broad and that most of the pages within these charts could be associated with different categories.

This study reveals once again the power of combining different technologies and using software such as Cogito to address these problems and to improve the knowledge within the Dark Web. The results led to a positive outcome where most of the expected observations from the authors were proved with the analysis of the data they extracted by using different metrics and techniques.

3.4 Overview

At this point, we have studied 3 works with some details that basically focused on using tools from machine learning algorithms to different types of metrics regarding the graph theory and social networks that reveal how large and how many techniques exist to extract relevant information from the Dark Web.

<i>Research</i>	<i>Data</i>	<i>Processing</i>	<i>Analysis</i>	<i>Results</i>
Unsupervised Learning Algorithms to apply on Dark Web forums	Provided Dataset	Social Network Analysis Metrics	Unsupervised Learning Algorithm [ML]	Found principal actors on the network
Predicting cyberattacks within Dark Web forums	Provided Dataset	Social Network Analysis Metrics	Supervised Learning Algorithm [ML]	Predicted cyber attacks
Building and Analyzing a Tor web graph	Crawled	Social Network Metrics	Social Network Analysis / Semantic Analysis	Insight Knowledge from Dark Web structures

Table 3.4: Comparative table with the three research works explored

There other recent researches that explore these unique characteristics of the networks to find different and interesting conclusions, such as the study [62], from 2020, where the authors make use of a crawler to gather web pages information according to some keywords so they could create a network graph with degree distribution and centrality scores to reveal the most popular websites. Another interesting study from 2021, [2], that studies the topology of the Tor network and examines the presence of bow-tie structure⁴. In this study, the authors used a web crawler built in python and built a graph with the found hidden services to analyse and compute metrics from the network, revealing some characteristics about these domains, such as the difficulty to find them as they are not spread that much within the Dark Web.

Apart from using different technologies and techniques, these studies have something in common, the process. A study from 2020, [40], not only reveals this assumption

⁴The Bow Tie model in graph theory gives a unique visual perspective over the structure of a network [61].

as they used a Systematic Literature Review method around what are the rising threats within the Dark web crimes and around what techniques are used to find criminals within the Dark Web. They did extensive research about studies that could help them clarify those questions. One relevant aspect concluded for this document is that there is an architectural framework analysis used that can be split into 4 steps: Data Collection, Data Preprocessing, Data Processing, and Results. However, how is each step of the framework implemented can differ depending on the data we're working with and on what are we seeking.

In terms of techniques used for crime detection, the authors of the study highlighted diverse techniques, however, we will just show the ones that had more researches according to the authors of this study and that are more relevant to this document.

- **Network Analysis Methodologies** This is one of the most used techniques as it can reveal the structure of criminal groups [40]. There are as well multiple works using network analysis to detect crime threats, classify network traffic and detect users within the Tor network [40].
- **Marketplace Scraping** Marketplaces within the Dark Web are where often criminals sell their illegal services and content. By evaluating data extracted from these marketplaces, it is possible to give some extra information about suspected criminals. This was already studied, and multiple types of research use this technique [40].

These methods used for crime detection also rely on techniques to process and “work” the data. This was also evaluated from the work in [40] where they took 10 research works selected from a list of 65 works selected carefully using the systematic literature review method and that had the bigger citation number. The 10 distinguished, identified by an ID, used the following process described in the following table.

With this, we believe we have all the tools to try our own approach and build an implementation based on these previous studies, but that tries to innovate and build a useful resource for future works.

ID	Data Collection	Data Pre-processing	Data Processing	Result
13	Crawling	Parser for text filtering	Classification with machine learning	Collect cyber threat warning
24	Available Database	Probabilistic disambiguation for affect ambiguity	Parser for analysis	Racism, violence, and hate affects from extremist groups
28	Crawling	Grouping web pages for blank pages	Classification with fingerprinting	Detect hidden services and communications
32	Available Database	Stylometric , character level n-gram, and time based features	Classification with machine learning	User identification in TOR network
41	Available Database	Topic modeling for network filtering	Network modeling for analysis	discover potential threats based on topics
35	Available Database	Regular expression for time-independent time-dependent variable selection	Classification with machine learning	Detect deception by non-verbal behavior monitoring
17	Crawling	Apple script scrapping	Analysis with Maltego tool	Get vendor account information and vendor's origin

Table 3.5: Comparison of the different techniques used from 10 different works [9].

DATASET 1

In this chapter, we will focus on the first dataset studied. To contextualise the dataset 1 and the approaches we took to study it, we will start by explaining where the data came from and how was it organised, followed by the approaches we took to increase the knowledge about the dataset and extract information ready to be looked at and analysed. These conclusions will take place in the last chapter, where we compacted all the results got from every process on an interactive web application.

4.1 Data Collection

The first dataset provided will be designated through the study as “dataset 1”, and it is simply the result of a Dark Web crawler that is a software that navigates to some given domains, here we will designate them as “seeds”, and follows every hyperlinks found in those while registering where it’s been. It allows understanding the relations between a domain (seed) and other domains (domains identified while visiting the seed). It is important to note that we might call those domains found by the “seeds” as either by domains analysed or simply by hyperlinks as in reality, they were discovered by hyperlinks while crawling the seeds.

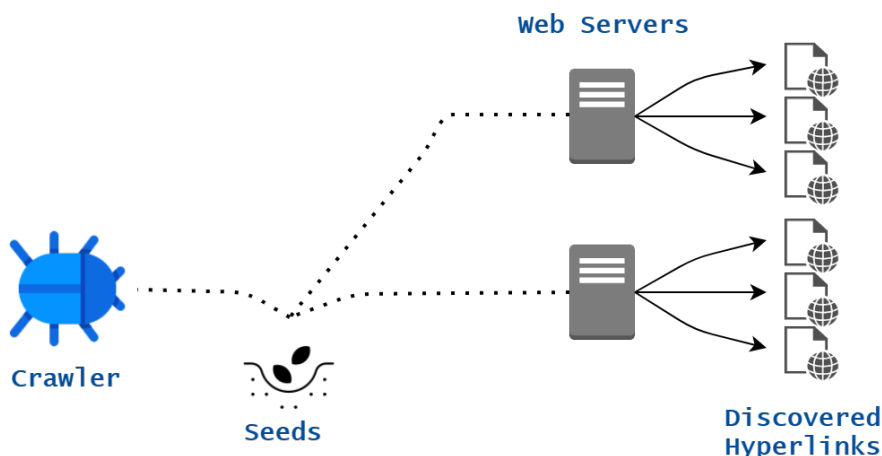


Figure 4.1: Web Crawler Diagram where the crawler represents the software, the seeds represent the starting domains used to access the web servers which the “bug” will crawl to find hyperlinks and discover new domains.

With this CFLW Cyber Strategies could identify 10 000 relations between 114 unique seeds and 7864 unique domains from the TOR and I2P Dark Nets. This means that from 114 domains, a crawler could find 7864 unique domains. Some seeds found repeated domains, therefore, from the 10 000 entries, we have 7864 that are unique and 2136 repeated. It is also important to mention that some domains discovered (hyperlinks) are at the same time seeds. This means that some of the 7864 unique domains are also in the 114 unique seeds. To be more precise, 104 of those are repeated between the domains discovered and the seeds. It can be confusing now, but it will get clearer by the end of this chapter.

The dataset was provided in an excel format with two columns, where the first one has the seeds and the second one the discovered domains. After understanding with which

	from	to
1	http://kpvz7ki2v5agwt35.onion	http://deepro33jjvfb3n7.onion
2	http://kpvz7ki2v5agwt35.onion	http://oniot2zvfczp4lpc.onion
3	http://kpvz7ki2v5agwt35.onion	http://abodxeycuklpva2v.onion
4	http://kpvz7ki2v5agwt35.onion	http://childhsifechod5u.onion
5	http://kpvz7ki2v5agwt35.onion	http://julyjaprir2uurlh.onion
6	http://kpvz7ki2v5agwt35.onion	http://zooscaqbvd5wfjul.onion
7	http://kpvz7ki2v5agwt35.onion	http://lolitmhfkpif7sky.onion
8	http://kpvz7ki2v5agwt35.onion	http://lolitmhfkpif7sky.onion

Figure 4.2: Excel file provided with the dataset 1.

data are we going to start this project, we’re now going to drill down on every task and process done on that data to create valuable information and take interesting conclusions.

4.2 Data Analysis

This dataset looked to be interesting in order to understand the relationship between domains within the Dark Web, however, it looked also limited as there was nothing but

a connection between a domain (the seed) and another domain (the domain discovered). So, the idea was to add more data to be possible to filter these relations and have different perspectives from the network. This led to the first task on this dataset, developing a scraper.

However, before that, we needed to have the data structured in order to it be easily accessible by any tool, so the first thing was to deploy a MySQL Web Server online with all the data.

4.2.1 MySQL Data Structure

The first step was to create three tables:

- **urls** - every unique URL from the dataset - 7874 rows
- **seeds** - every unique URL from the dataset excel file column “from” (i.e. the seeds) - 114 rows
- **hyperlinks** - every unique URL from the dataset column “to” (i.e. the domains discovered by the seeds) - 7864 rows
- **edges** - relations from the dataset 1 - 10 000 rows

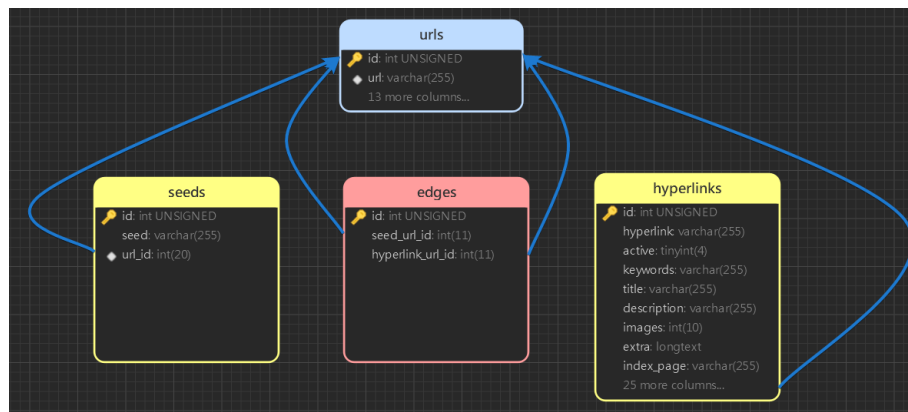


Figure 4.3: Entities relation diagram of the first four tables created on the MySQL database.

With these four entities created, we related the URL table upon a one-to-many relation with the seeds table, the hyperlinks table and the edges tables, as an URL can be a seed and a hyperlink simultaneously, and edges are made of a seed that discovered a hyperlink (two URLs). In the figure above, you can observe that the hyperlinks table has already a lot of fields, but we will get to them in a moment.

After modelling the data, we started the next step that was gathering more information about each hyperlink, which is our object of study here.

4.2.2 Scraping Domains

Objective The goal was to add more information to every domain found by the seeds in this dataset. This would be very important to afterwards allow us to use the results with a set of different data analysis technologies involving natural language processing and others.

Technology The technology used to create this scraper was based on the programming language Python on its 3.7 version, with the help of Python's packages such as Selenium WebDriver¹ we were able to configure a web driver using Firefox to connect either to the I2P network or to the TOR network depending on the hyperlink we were analysing because, as we know, our dataset holds hyperlinks from both darknets.

Preparation The preparation needed to run the script for scraping these websites was basically to have I2P and TOR installed on our machine. Both of these technologies work through a proxy that is running in your machine when we start them. This means that we can easily be navigating on TOR or I2P in any web browser driver by properly configuring the proxy setting.

Process The process was basically to run each of the hyperlinks from the database, one at a time, and access their index web page (i.e. the first web page shown when you access a root domain) and extract all the relevant information from it. We can organise the process as follows:

1. Get an URL from the hyperlinks table
2. Check if it belongs to the I2P network or the TOR network and prepare the Firefox web driver proxy accordingly
3. Check if the website is available by checking the HTTP status code answer.
4. If the website is available, access it and retrieve the following:
 - title from the title tag
 - description from the meta tag with description name
 - keywords from the meta tag with keywords name
 - source code from the web page (later we would cut the HTML tags and save only the text without HTML)
 - count the number of images by counting the number of .jpeg, .jpg, .gif and .png files formats in the source code

¹Python package that provides the possibility of using browsers web drivers such as Firefox, Internet Explorer and Chrome. More information at <https://selenium-python.readthedocs.io/>

- Save the information gathered on the respective table of the MySQL database (hyperlinks table)

Results In terms of results, they were successful as we turned a simply dataset containing just URLs into a dataset containing a lot more information about these URLs. All of the information was saved on their respective fields added to the MySQL hyperlinks table. An important remark here is that due to the nature of the networks we were accessing, I2P and TOR, the script took some time to scrape the information from every domain.

id	hyperlink	active	keywords	title	description	images	extra	index_page
2	http://oniot2zvfzczp4lpc.onion	1	None	OnionDir	None	0	OnionDir OnionDir oni	http://oniot2zvfzczp4lpc.onion
3	http://abodxeyculjpvva2v.onion	1	None	Abode Desires	None	1	Abode Desires Login U	http://abodxeyculjpvva2v.onion
13	http://yb24hizyti5oudw.onion	1	None	10x Your Bitcoi	None	5	m 10x Your Bitcoins in	http://yb24hizyti5oudw.onion
22	http://torbox3uiof6wchz.onion	1	None	This domain ha	None	1	This domain has been	http://torbox3uiof6wchz.onion
24	http://area23t23azzonor.onion	1	None	This domain ha	None	0	This domain has been	http://area23t23azzonor.onion
25	http://tordesk6z3bg3sb.onion	1	None	Tor Desk :: CRM	None	13	Tor Desk :: CRM Outso	http://tordesk6z3bg3sb.onion
26	http://kpvz7ki2v5agwt35.onion	1	None	Hello!	None	0	Hello! Looking for the	http://kpvz7ki2v5agwt35.onion
31	http://xmh57jrzrnw6insl.onion	1	None	This domain ha	None	1	This domain has been	http://xmh57jrzrnw6insl.onion

Figure 4.4: Portion of the hyperlinks table with the results from the scraping process.

An important aspect we noticed was that some domains had little text, and the text was not relevant. As an example, we can observe the following hyperlink scraping result that curiously was the very first hyperlink from the dataset. So, we decided to check

id	1
hyperlink	http://deepro33jvfb3n7.onion
active	0
keywords	None
title	Teen Deepthroat
description	None
images	10
extra	Teen DeepthroatYOUR WEBSITES NAME!username and password!user or password error!invitation code!Sign Up!To have full access to the web, you must be invited by another member of the site or give a financial contribution. Copyright 2015-2021 - deepro33jvfb3n7.onion - Deepthroat onion
index_page	http://deepro33jvfb3n7.onion/

Figure 4.5: On this image, we can see the first hyperlink of the dataset with little and non-relevant text on the “extra” field, which is the one with the text removed from its index web page source code

manually if this was something recurrent, and we quickly noticed that some of the other web pages that did not have a lot of informative text about the website content on its HTML source code had it written in pictures. This encourages us to find a solution, and it led us to try an Optical Character Recognition algorithm to extract this “hidden” text from the images to enrich our data extracted from the websites.

Notes Some notes about this task:

- As we know from chapter 1 section 2.3 these darknets are slow due to their nature, and therefore, requests took some time to be fulfilled, and if a website was having connection issues or was down, it would just be pending for a while. This forced us to find a good timeout value to understand when a domain is down upon a request. The values needed to be different depending on the darknet, especially

because I2P and TOR work differently, and they are not similar in terms of speed. Assuming that our machine had a stable and fast internet connection after some testing and reading, as there are no specific thresholds for this, we found that 30 seconds would be acceptable for TOR and 120 seconds for I2P. This was the result of trying manually multiple accesses on many of these domains and checking how long was the worst successful tentative and adding a margin to it. After some small research, we found these values to be acceptable.

- As the objective was to just gather more information about each hyperlink and not to know and parse everything about it, we just limited our scraper to run just on the domains index web page. However, it could have been interesting if we did follow every hyperlink on this web page to other web pages within the same domain to enrich the dataset.

4.2.3 Optical Character Recognition (OCR)

Objective The objective of this approach was simply to solve a problem and possibly prove that this is something that the owners of websites take into account when they're building their websites, which is that they use images to display text instead of just writing them normally within the web page HTML code.

Technology About the technology used for this approach, it was subject to some thinking and reading because there are so many tools that are capable of doing OCR. We ended up using one named Tesseract OCR² that was developed by HP and maintained by Google, but more important than that, it gave the best results when compared with other libraries for image processing. Therefore, we stuck with Python and used a package named pyTesseract that uses the Tesseract OCR Engine.

Preparation The preparation here was not much as the packaged used just needed to have the Tesseract OCR Engine installed on our machine to be ready to start receiving images and outputting their text. However, we had to prepare a structure to save the results, so we created a MySQL table named `hyperlink_images` within our database with the following fields:

- `url_id` - to identify the URL id to which the text references to
- `image_url` - domain's path to the image
- `image_text` - the output from the Tesseract OCR Engine

This table has a one-to-many relation with the URLs table (figure 4.3) through the `url_id` field as one URL can have multiple images with text.

²More information about it at <https://github.com/tesseract-ocr/tesseract>

hyperlink_images	
id	int UNSIGNED
url_id	int(11)
image_url	varchar(255)
image_text	longtext

Figure 4.6: MySQL table created to save the texts extracted from the images.

Process The process used was incorporated within the scraping process. So basically, while scraping a website and after gathering the number of images within the same website, we would verify if that number is higher than one and if so, we would give those images to the Tesseract Engine. If the output had more than 3 words, we would save the output on the `hyperlink_images` table with the respective URL id to which it belongs. The reason we filtered out the outputs with less than 3 words was simply because the engine was returning a lot of texts with random spaces and UNICODE characters, and we found that 3 was a good threshold to avoid this “trash text” however you will still see some of it in the results.

Results The results were acceptable as for most of the websites, we were able to enrich the data scraped and, more important than that, for some websites with little text, the use of OCR was crucial because they had images with explicit text about the website purpose. To exemplify, let’s look into the first domain discovered in this dataset. If we take a look at the previously shown figure 4.5, we see that there is not much text on its “extra” field that could give an insight into what is this website content about, however, after running the OCR on the website images we achieved the following: If we look carefully for the

id	url_id	image_url	image_text
17	1	http://deepro33jivfb3n7.onion/images/f.jpg	molavlag ↕
18	1	http://deepro33jivfb3n7.onion/images/v.jpg	Vite (=Yoys ↕
19	1	http://deepro33jivfb3n7.onion/images/d.jpg	IDYoyiVialterskeks ↕
20	1	http://deepro33jivfb3n7.onion/images/s.jpg	↕
21	1	http://deepro33jivfb3n7.onion/images/2.jpg	Welcome to DeepthroatFeatures the largest collection of user submitted teen videos and pictures.
22	1	http://deepro33jivfb3n7.onion/images/4.jpg	News 22/11/2018 We havea english and a Polish forum. How about a German forum now? i'Now

Figure 4.7: Entries from the `hyperlink_images` table for the URL id = 1 with 6 images where the OCR was applied.

figure above, most of the images had trash, but one of them (row with id = 21) was crucial by giving explicit information about the website purpose.

Below, we will show a figure with the row id 21 in detail: Another important aspect

id	21
url_id	1
image_url	http://deeepro33jjvfb3n7.onion/images/2.jpg
image_text	Welcome to Deepthroat Features the largest collection of user submitted teen videos and pictures. Deepthroat is the first original teen porn site on the deep web. Deepthroat is updated every single day of the year with fresh new user-submitted porn content.

Figure 4.8: One of the entries (row id = 21) in detail where we can confirm that the text extracted from the image clearly passes the message of this website purpose

we noticed was that on these tests, and while building the scraper together with the OCR, some of the websites that we were trying to debug sometimes were online and sometimes offline. This is a characteristic that is not so usual in the clear web, and there are even measures and tools to analyse the performance of a website in terms of its availability because it is a really important aspect and a reliability point. In the Dark Web, this is not so true as we don't have those measures and statistics available for a given domain, so we decided to create a temporal analysis script to check the changes of this availability within the dataset.

Notes Some notes about this task:

- One important aspect in this analysis was that to apply the OCR, we need to have the image file, and it would be risky and time-consuming to be downloading every image from every website, so we decided to simply take a screenshot through Firefox web driver for each image we wanted to study, and we successfully solved that concern.
- Another important remark is that as images can have text in many languages, the Tesseract Engine allow us to give beforehand the language in which we are interested in finding text. This helps the detection to be more accurate, and therefore we detected the language of every website, using a library that we will present in the section 4.2.5, before using the OCR to increase our accuracy.

4.2.4 Temporal Analysis

Objective The objective was to confirm whether it is usual that websites within the Dark Web are unstable in terms of their availability. So, the idea is to check the availability of each hyperlink to afterwards make conclusions and have visualisations such as a web graph with the current status of the network where active nodes would be represented in green and inactive nodes in red.

Technology The technology used to perform this task was a script in Python that is continuously checking the availability of the dataset with the help of a python package named Requests³. Apart from that, we also used a Social Network Analysis tool named Gephi⁴ that basically would allow us to afterwards represent the results in a web graph.

Preparation The preparation needed was related with having a structure to save the data. In order to do that, we created a new MySQL table within our database named `hyperlinks_temporal_status` that had the following fields:

- `url_id` - URL id to which the measure references to
- `status` - 0 if it was down or 1 if it was reachable
- `status_code` - HTTP Status Code response
- `traceback` - error logging
- `created_at` - timestamp from when the measure was taken

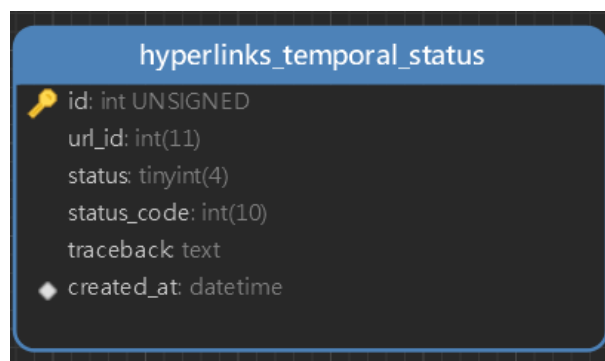


Figure 4.9: Table created to store the measures from the temporal analysis script.

The MySQL table represented in 4.9, has a one to many relation with the URLs table (figure 4.3) through the `url_id` field.

Process The process was simply to have a script running in a loop, meaning that when it reaches the end of the dataset, it would turn around and start over again. The logic used was the following:

1. Get a hyperlink one at the time
2. Check if it belongs to the TOR or I2P network
3. Prepare the request proxy settings accordingly to access the TOR or the I2P network
4. Make a request and check the status code

³More information at <https://docs.python-requests.org/en/master/>

⁴More information at <https://gephi.org/>

5. Save a new entry on the MySQL table with the URL id from which it references to, the measure and the timestamp

With the measures saved, it opened a set of possibilities to display and find interesting patterns from the data. One that was interesting to study was the visualisation of the network availability web graph, in which we took the last measures from each node and created a web graph where active nodes would be painted in green and inactive nodes painted in red. The nodes are unique domains from the dataset, and edges (with direction) represent if a destination domain has been seen in the origin domain. In other words, we will only have edges from seeds to the hyperlinks discovered on those seeds.

Results In terms of results, they were not so immediate as this was a long term approach. We kept the script running for some months (from March till June) with some stops in between, but in the first month, we could already observe that some websites had a great variability in their status.

id	url_id	status	status_code	created_at
2782	2420	1	200	2021-03-23 02:03:21
31556	2420	0	503	2021-04-18 21:05:50
44544	2420	0	503	2021-04-21 18:37:15
52408	2420	1	200	2021-04-23 17:44:13
60273	2420	0	503	2021-04-25 18:37:03
68137	2420	1	200	2021-04-27 18:26:03
76001	2420	0	504	2021-04-29 18:33:31

Figure 4.10: Measures between March 23rd and April 29th showing some variability in the availability of the website with ID 2420.

The figure above shows an example of measures for the website with the ID 2420 where its availability changes almost every time we came back to check it. However, we can also see that there is a big distance between the timestamps, and this happened because the script was taking long to run because not only does access the Dark Web darknets such as TOR and I2P take longer than accessing the usual clear web but also because that websites that were down made the script being there waiting for a timeout as these websites could be just having a slow connection. A later solution was to run it in parallel, so basically, we broke the dataset into 4 pieces, and the scripts were making 4 requests simultaneously, increasing the speed of analysis by 4.

In terms of the availability web graph, we obtained the following:

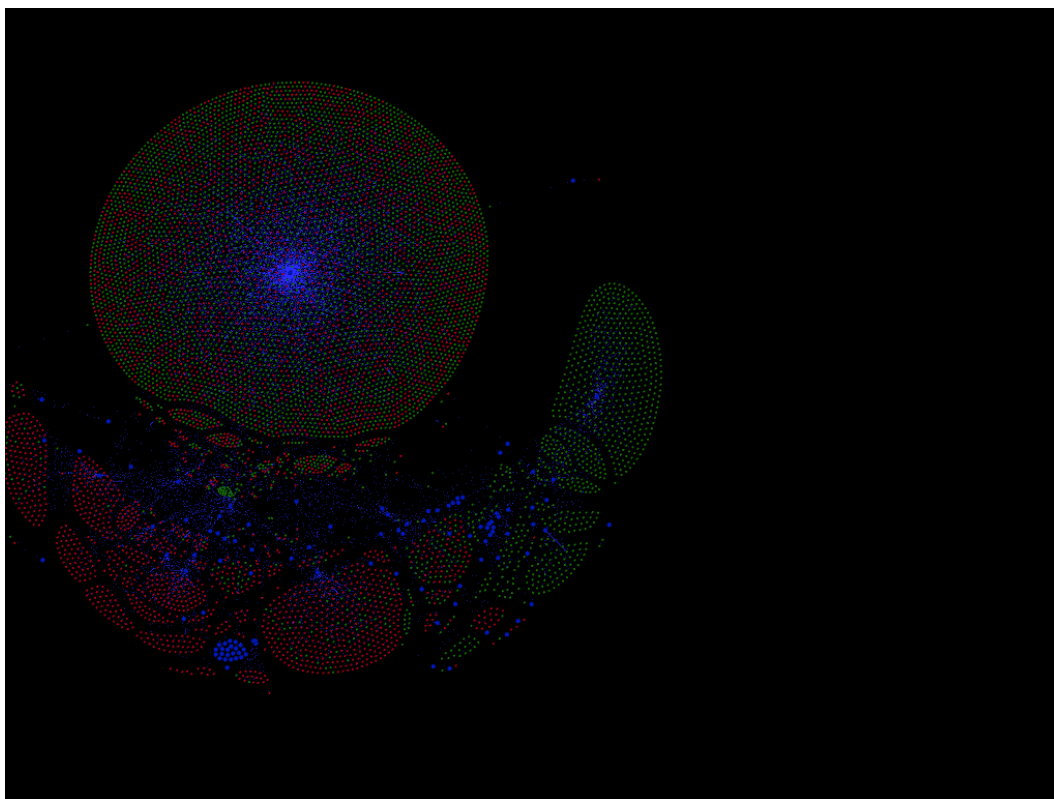


Figure 4.11: Availability web graph from the dataset 1 with measures taken from 21-03-2021 till 24-03-2021.

On the figure 4.11 we can observe the whole network where online domains are represented in green and offline domains in red. We can also observe a set of domains on the right that are almost 100% online and on the bottom left a part that is mostly offline. In the middle, we have a lot of nodes connected to one of the biggest cores of the network, which has more than 5000 edges, with a mix of online and offline domains.

From this figure, we cannot give exact conclusions about the network, but we can raise some possibilities. Is the left part of the network (zone with red nodes) discontinued? or are these nodes just offline and will turn online in the future?

These are just some questions we raised that we will take into account in the chapter 6 where we will combine the results from every approach.

Till now, we had just worked on forms of increasing the information about the dataset. From this point on, we started to study the data gathered and making it useful. With this data in our possession, the first idea was to create an automatic classifier that would work as a helper to give hints of possible classifications for web domains content.

4.2.5 Automatic Classifier

Objective The objective of this automatic classifier is to give possible classes on which we could classify a website according to its content. To do that, we decided to go with one

of the most used technologies for these problems, neural networks.

Technology The Technology used here was also based in Python, with the help of Google Colab⁵ which is a platform that offers a virtual environment of Python helpful to use with neural networks as they also provide GPU processing which is important to speed up the training process of neural networks. Google also developed a library named Tensorflow⁶ which is powerful for deep learning solutions. Tensorflow will be the library we will use to train, optimise and export the neural network model, which we will use to automatically classify the websites giving the data we extracted before.

Preparation The preparation here was more complex than before as we decided to create a neural network model from scratch. So, basically, we decided to go with a supervised learning algorithm from machine learning named neural network, which needs a training dataset for it to be able to learn from examples and then be capable of making its own classifications. As we did not have such a training dataset and it would be difficult to create it in the time frame of this project, we looked up online and found a research, [1], in which they provide a dataset named Darknet Usage Text Addresses 10K or simply DUTA 10K. This dataset is nothing more than an excel file with 10 000 entries where on each row there is a unique domain from the TOR network with the corresponding language, the main class and sub class classification given manually for each of the domains.

	Onion_Address	Main_Class	Sub_Class	lang
1	darkw4u3xkeb5pzn.onion	Art	Music	en
2	tune4xs6mj2evcr6.onion	Art	Music	en
3	a4yedjgciupu7zzt.onion	Art	Music	en
4	76qugq4pb42lpgwx.onion	Art	Music	en
5	kurdox4tfzujxddq.onion	Art	Music	en
6	zohaq6hhk2p52c7d.onion	Art	Music	en
7	5cqzpj5d6ljxqsj7.onion	Art	Music	en
8	76qugh5bey5gum7l.onion	Art	Music	en
9	kiy52zq46nqu426l.onion	Art	Music	en
10	weasylartw55noh2.onion	Art	Music	en

Figure 4.12: Small portion from the DUTA 10K dataset

Before proceeding with the study, we decide to take a look into the DUTA 10K and get more knowledge about it.

⁵More information at <https://colab.research.google.com/>.

⁶Deep Learning Library. More information at <https://www.tensorflow.org/>

In terms of classification, it has 26 unique main classes, and 24 unique sub classes. Their distribution is as follows:

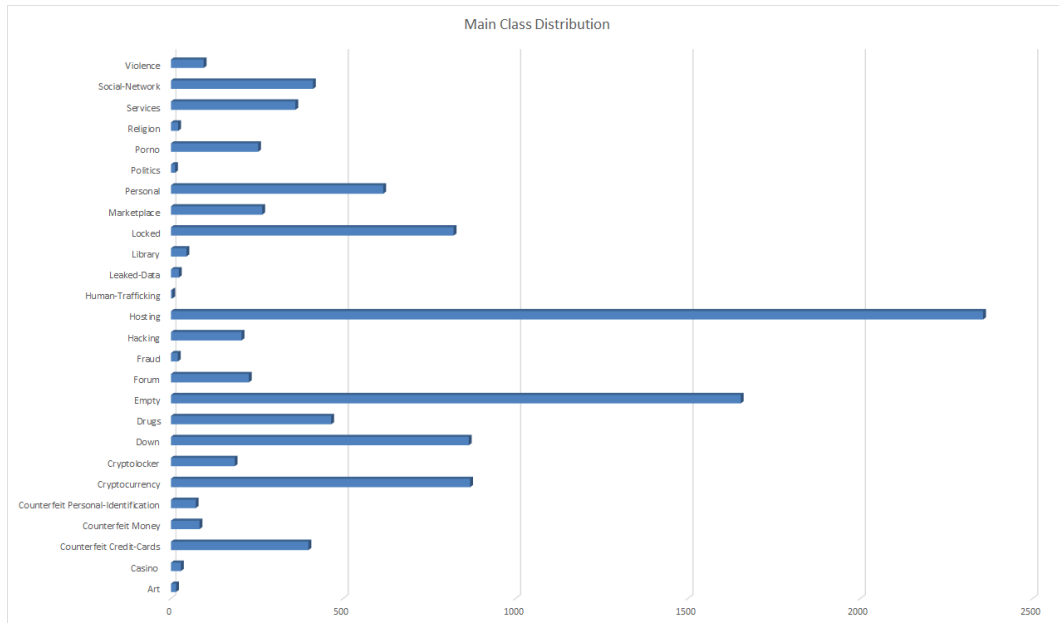


Figure 4.13: DUTA 10K Main Class Distribution

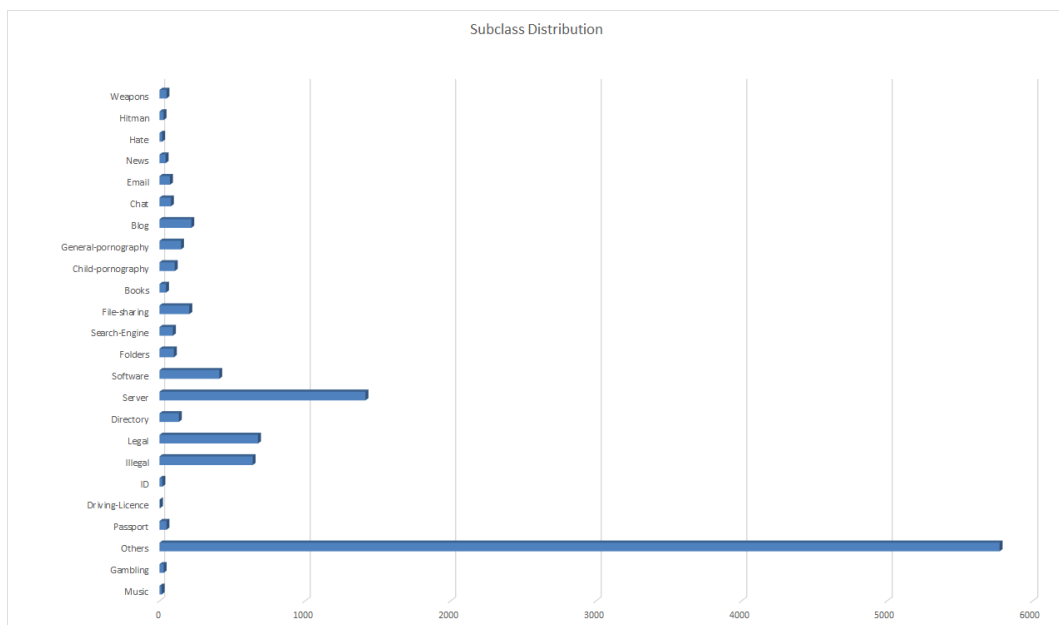


Figure 4.14: DUTA 10K Sub Class Distribution

From the figures above, we can observe that some classes in the main class distribution or in the sub class distribution are way more common than others. This can be a problem for the neural network training process because the training dataset is clearly unbalanced, however, there are solutions for that, and we will explore them shortly.

At this point, we have a dataset that can be used for the training process of the neural network, however, the objective is to input the neural network with text (i.e. with the text extracted from the hyperlinks) to get a classification. We have the domains from the DUTA 10K with the respective main class and sub class classification, but we don't have their text to use as examples for the neural network to understand which texts relates to which main class and sub class. Therefore, we would need the texts from every domain in the DUTA 10K to use as examples for the neural network training process. To do this, we had to scrape the DUTA 10K domains, just like we did with dataset 1 in section 4.2.2.

Before that, we created two tables, just like we did with dataset 1, a table for the hyperlinks from the DUTA 10K, which we called `hyperlinks_training` and another table called `hyperlink_training_images` for the OCR analysis of the images from the DUTA 10K. With this, we would have two datasets, the DUTA 10K and the dataset 1, which is our object of study, with the same exact structure, apart from the fact that the DUTA 10K dataset have already been classified manually by its creators.

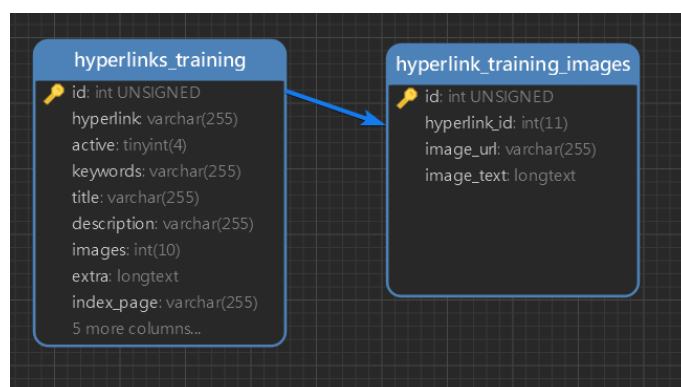


Figure 4.15: MySQL tables created to prepare the DUTA 10K dataset for the training

We hope that the explanation for the preparation of the neural network model training was clear, but if not, at the end of this section, everything will hopefully make sense. Nevertheless, to summarise the idea till now is that we found a dataset with 10 000 domains that were already classified (DUTA 10K), but this dataset did not have the same structure as dataset 1, so we had to make them equal in terms of structure so that the only difference between these two datasets will be that one has a classification (DUTA 10K) and the other does not (dataset 1), everything else will be the same, they both have a count of the number of images, the title of the domain, the description, keywords and the text from its index web page.

After running the same scraper we created for the dataset 1 in the DUTA 10K we obtained the same structure as the dataset 1.

id	hyperlink	active	keywords	title	description	images	extra	index_page
2	http://oniot2zvfzcp4lpc.onion	1	None	OnionDir	None	0	OnionDir	OnionDir oni http://oniot2zvfzcp4lpc.onion
3	http://abodxeyculjpa2v.onion	1	None	Abode Desires	None	1	Abode Desires	Login U http://abodxeyculjpa2v.onion
13	http://yb24hizyti5oudw.onion	1	None	10x Your Bitcoi	None	5	m 10x	Your Bitcoins in http://yb24hizyti5oudw.onion
22	http://torbox3uiot6wchz.onion	1	None	This domain ha	None	1	This domain has been	http://torbox3uiot6wchz.onion
24	http://area23t23azzonor.onion	1	None	This domain ha	None	0	This domain has been	http://area23t23azzonor.onion
25	http://tordesk6z3bg3sb.onion	1	None	Tor Desk :: CRM	None	13	Tor Desk :: CRM	Outso http://tordesk6z3bg3sb.onion
26	http://kpvz7ki2v5agwt35.onion	1	None	Hello!	None	0	Hello!	Looking for the http://kpvz7ki2v5agwt35.onion
31	http://xmh57jrzrnw6insl.onion	1	None	This domain ha	None	1	This domain has been	http://xmh57jrzrnw6insl.onion

Figure 4.16: Portion of the hyperlinks table after running the scraper

id	hyperlink	active	keywords	title	description	images	extra	index_page	class	sub_class	lang
1	darkw4u3kxeb5pzn.onion	1	None	Eva and Franco	None	0	Eva and Fr	http://darkw4u3kxeb5pzn.onion	Art	Music	en
3	a4yedjgciupu7zst.onion	1	None	GNUMP3d 192	None	133	GNUMP3c	http://a4yedjgciupu7zst.onion	Art	Music	en
7	5cqzpj5d6ljxqsj7.onion	1	None	Eva and Franco	None	0	Eva and Fr	http://5cqzpj5d6ljxqsj7.onion	Art	Music	en
8	76qugh5bey5gum7l.onion	1	None	*** Deep Web R	None	26	*** Deep V	http://76qugh5bey5gum7l.onion	Art	Music	en
9	kiy52zq46nqu426l.onion	1	None	No Tone	None	18	No Tone N	http://kiy52zq46nqu426l.onion	Art	Music	en
12	mnyz5mo5konjbnl.onion	1	None	Onion Dir - Adu	None	17	Onion Dir	http://oniondir5mo5konjbnl.onion	Art	Music	en

Figure 4.17: Portion of the hyperlinks_training table after running the scraper which shows the exact same structure as figure 4.16 apart from the language, main class and sub class fields

If we compare the figure 4.17, which is representative of the scraping result of the DUTA 10K, and the figure 4.16, which is representative of the scraping result of the dataset 1, we can conclude that now they are similar. The difference is on the DUTA 10K hyperlinks table that has three extra fields when compared with the table of the hyperlinks from dataset 1. These fields are the lang, class and sub class, which were both given on the DUTA 10K. Now we “just” need to train a neural network with the prepared structure of the DUTA 10K table and see if we can get the training process to generalise well the problem and find patterns in the text that would afterwards allow us to input that model with the dataset 1 and get those extra two fields, the main class and sub class. It is also relevant to mention that as the DUTA 10K provided the language to each domain, we also thought that it could be interesting to also have that for our dataset not only for statistical purposes but also because it is information relevant to enhance the results of some algorithms that can distinguish languages such as the OCR, so, we used a python package named TextBlob which is a natural language processing library that offers many functionalities like detecting the language of a given text with the help of Google Translate Engine.

One final remark for the preparation of the training dataset is that the scraping of 10 000 domains left us with just 2527 domains to use, which is a small number because the rest were offline or were inaccessible. This fact made us check some of the domains manually, and another problem surged, as the DUTA 10K was more than 2 years old, at the time of writing, some of the classes given were not correct with the website because meanwhile, it changed. This could be a problem for the final results of the classifier, and in fact it was, but we will get there shortly.

Process The process to create the automatic classifier was also complex, and it was one of the tasks that required more time developing. Now that we had the training data

prepared and on a MySQL table (easy to access), we created a script on Google Colab and developed a neural network. There are multiple types of neural networks, but as we saw in the chapter 1 section 2.4.3 the main architectures are the Recurrent Neural Networks, Convolutional Neural Networks and Artificial Neural Networks. Within these types, there are different variants, and as there are tons of parameters and forms of mounting a neural network, these structures are not so fixed.

After a lot of reading and testing, we went with a Convolutional Neural Network architecture by using one dimension convolutional layers on our model as this architecture is mostly used to be worked with images and video kernels that have higher dimensions. Nevertheless, these types of neural networks are powerful because they take into account the relation between values going through a layer and the older values that have already passed through there, taking therefore into account spatial and temporal characteristics of the data. The process used from preprocessing the text until saving the model can be succinctly described as follows:

1. Process the dataset texts (text scraped + images with text)
 - Remove Punctuation
 - Remove Stop Words (e.g. “a”, “the”, “is”, etc.) - required language detection
 - Lower all the text
2. Split the dataset in 67% for training and 33% for validation
3. Create an array for every domain with all the words scraped from them
4. Convert that words in numbers (as neural networks do not understand words)
5. Train the neural network
6. Save the neural network model

The process described above seems too direct and clear, but it was way tougher than it looks. We have already chosen an architecture, but there are many other factors such as the number of layers, number of neurons on each layer, the activation function of each layer, number of epochs, loss functions, etc. As there is no right answer for all of these parameters, and every scenario is different, we used some hyper parameterisation techniques to find the best parameters for the neural network. There are many techniques to do this, such as Grid Search, Random Search, Bayesian Models, etc. They all have different forms of finding the best parameters, but their goal is the same, return the model's characteristics with the highest performance metrics (usually referencing the accuracy or f1 score). With this, we decided to try some of the techniques, and we used Random Search, Hyperband (which is based on Early-Stopping) and Bayesian Search. They work very similarly: we give a set of options to try based on our architecture (i.e. we establish that we want to try having [100, 200, 300] neurons on the first layer, try having

as activation functions [“sigmoid”, “tanh”], and so on) and then the algorithm will try different combinations of the possible options we provided and return the best result. As you might think, this is a process that takes quite long, and therefore it took some time for us to find a good solution.

After finding good parameter settings for the problem, we also searched for other optimisations, and one that is really worth it is the use of an embedding layer. An embedding layer is nothing more than a layer that simply takes an embedding matrix, which represents the words that we’re going to train by a set of weights with a semantic meaning. In other words, if in my dataset I have the words “father”, “son”, “further” and “sun”, this matrix will transmit that the word “father” and “son” are semantically closer and therefore they will have similar weights when compared with the other two words. This is powerful because it facilitates the neural network in discovering patterns.

Another important setting we added to our model was balancing the dataset. As you have seen in the figure 4.13 and figure 4.14 the biggest part of the training dataset belongs to the main class “Hosting” and the sub class “Empty”, this means that if we did nothing, the classifier would be influenced by this biased vision of the dataset. So, a thing that can be done when we have an unbalanced dataset is adding weights to the classes and making them have the same chance of being the classifier’s result. This is done by simply finding the class with the higher number of examples and dividing that number by the number of examples of the class we are computing the weight. For each of the neural networks, we made some modifications within the training dataset to improve the results.

An important remark to highlight here is that till now we gave the idea that we would input the neural network with text and it would give us the main class prediction and the sub class prediction, however, we decided, as there would be too many outputs, to simplify the output from the neural network and brake the classifier in two different neural networks. So, we decided to create not one classifier but two, one for the main class and the other one for the sub class.

For the main class classifier, we erased some of the main classes.

- “Down”, “Fraud”, “Empty” and “Hosting” were deleted

If we look into the figure 4.13 we can observe that these main classes are either the ones with higher examples or lower examples, and the reason why we did this is simple, the main classes “Down”, “Empty” and “Hosting” were affecting the results as they were more than 50% of the training dataset and the balancing optimisation was not enough. The “Fraud” was removed because the domains with this main class were offline, and therefore we could not scrape any text from them, meaning that we would not be able to use it to train the neural network.

For the sub class classifier we used a different approach and we compacted some of the sub classes and erased others.

- “Driving License”, “Passport” and “ID” were compacted in a sub class named “Identity Documents”
- “Folders”, “Hate”, “Email”, “Blog”, “Chat”, “Server” sub class were deleted
- “Directory” and “Search-Engine” were compacted in a sub class named “Directory”
- “File-Sharing” and “Software” were compacted in a sub class named “Software Services”
- “Illegal” and “Legal” were compacted in a sub class named “Markets”
- “General-Pornography” and “Child-Pornography” were compacted in a sub class named “Pornography”

The reasons here are similar to the ones in the main class, some were deleted because the domains with those sub classes were offline or they were unbalancing the dataset too much or were not relevant. The ones we compacted was intending to help the quality of the neural network training process as we notice a significant improvement in the training by doing this. We also tried to mitigate the effect of the training dataset being outdated by changing each sub class of the DUTA 10K manually to “Pornography” when keywords related to this content were found regularly through a website.

Results The results obtained were not great as there were some problems identified along the process that was crucial to the outcome of the classifier. The fact that only a small percentage of the domains from the DUTA 10K were available to crawl and that a percentage of those domains were outdated in terms of classification, was decisive for the final results.

In terms of training we were able to train a model for classifying the main class with the following metrics:

- Accuracy - 38.79%
- Loss - 5.97
- F1 Score - 32.47%
- Precision - 48.99%
- Recall - 24.42%

and for the sub class with the following metrics:

- Accuracy - 97.03%
- Loss - 0.22
- F1 Score - 97.45%

- Precision - 98.07%
- Recall - 96.85%

We can observe that the second model (sub class classifier) was way better than the first one (main class classifier), and that has to do with the adaptations we made to the dataset. By compacting some classes and correcting some outdated classifications from the DUTA 10K, we were able to increase by a lot the quality of the model, however, we observed some overfitting that was hard to reduce.

Below we will see an example of the results obtained with the automatic classifier for the domain analysed with id 178:



Figure 4.18: Index web page of a hyperlink found by seed with ID 178.

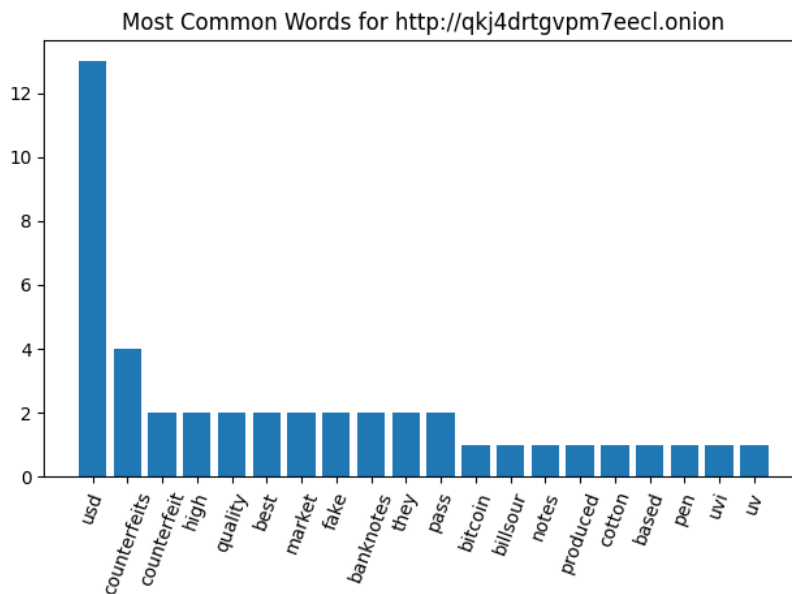


Figure 4.19: Top 20 words found on the text scraped from URL id 178

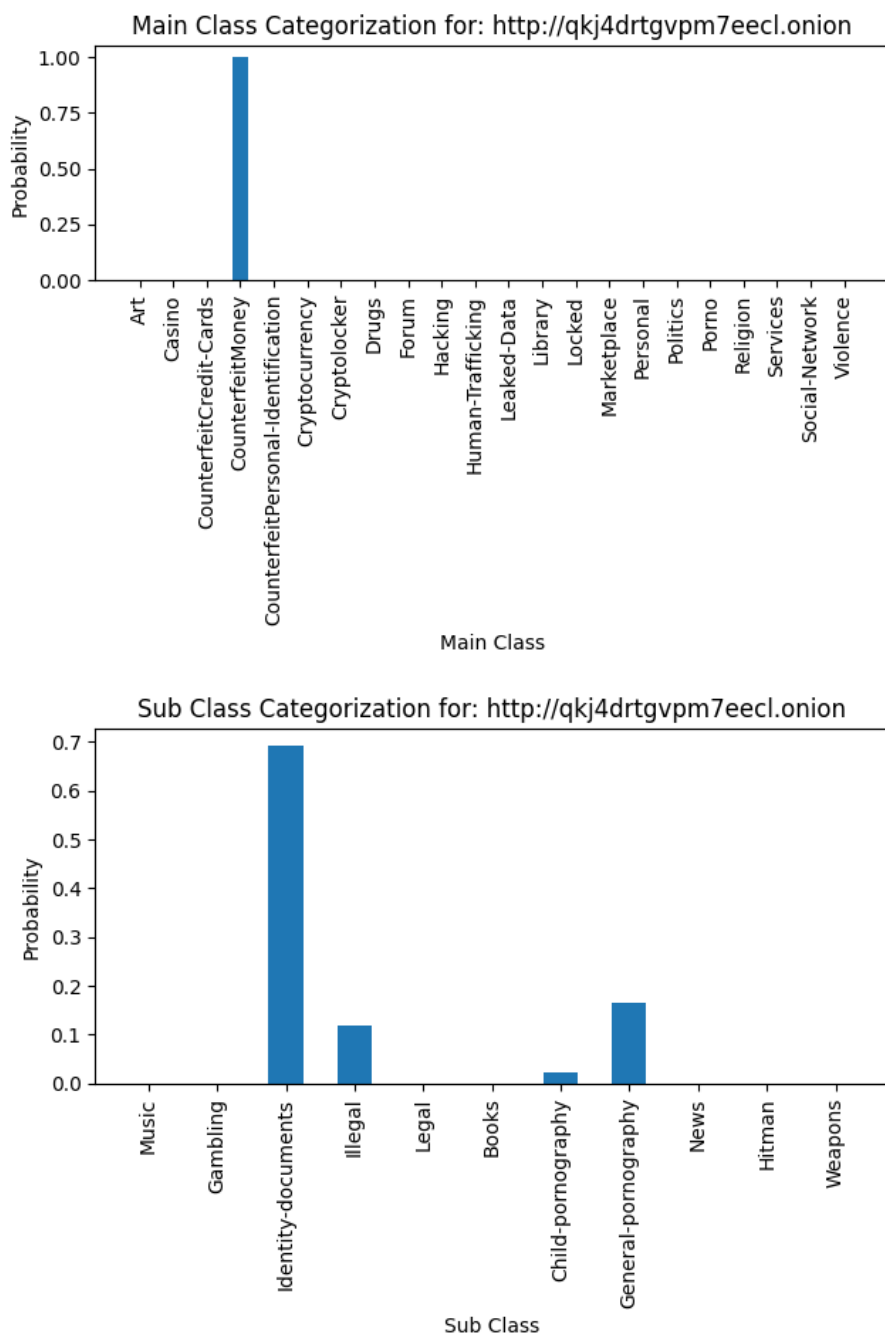


Figure 4.20: Automatic Classifier result on the left for the main class and on the right for the sub class

The results for this specific URL id were very accurate, however, we noticed a problem with domains where the main class result indicated a website content of “Cryptocurrency” or “Counterfeit Credit-Cards”. This problem is the result of the DUTA 10K dataset being outdated since it is more than 2 years old and some domains changed their content meanwhile.

Below, we have an example for the URL id 1:

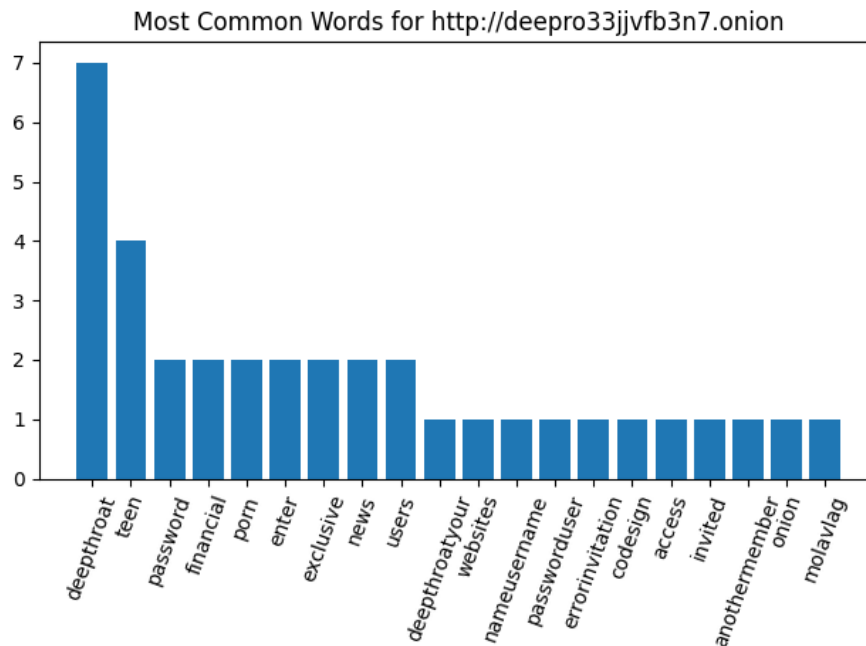


Figure 4.21: Top 20 words found on the text scraped from URL id 1

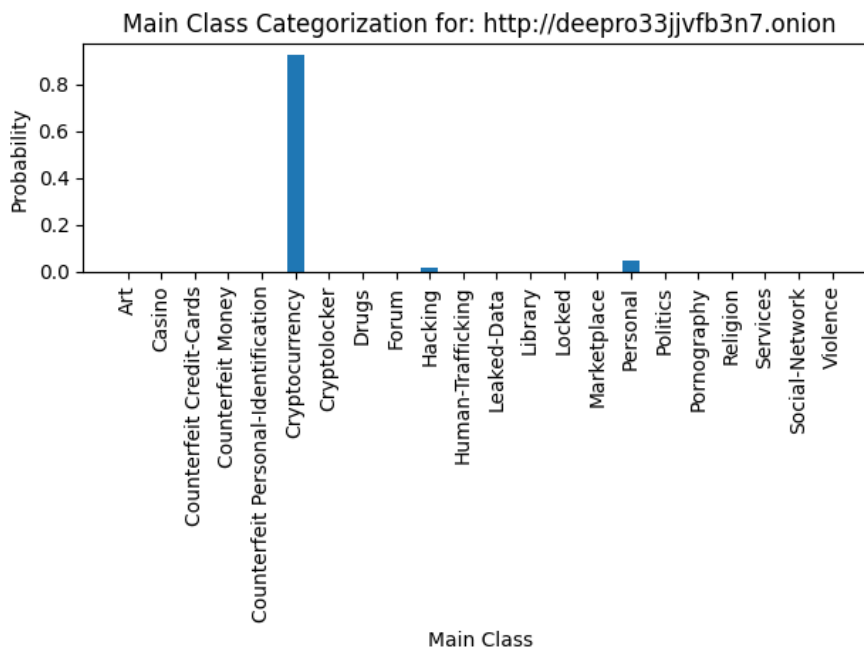


Figure 4.22: Automatic Classifier result for the main class

On the figure 4.22 we can observe that the classifier believes this is a website about “Cryptocurrency” and from the figure 4.21 we can observe that this website is clearly a pornography domain. We could think that it is an error from the neural network model as

it is not perfect, but in fact, the neural network is doing its job here. The problem is that most of the domains from the DUTA 10K that were about “Cryptocurrency” and “Counterfeit Credit-Cards” are now mostly websites of “Pornography”, so the neural network model learned that the text about pornography should be classified as “Cryptocurrency” or “Counterfeit Credit-Cards”. To prove this, we can simply look into the results of the sub class neural network model, as when we trained it, we already knew about this problem, and we updated the sub class for pornography domains. We basically searched on the text scraped from every domain in the DUTA 10K for pornography keywords, and if found, we would ourselves change the sub class to “Pornography”. The result was the following:

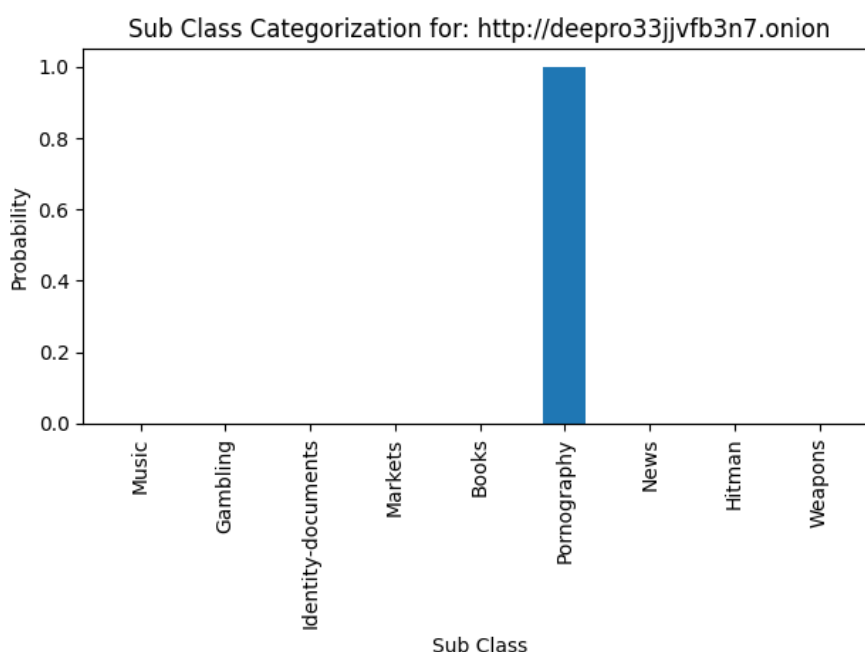


Figure 4.23: Automatic Classifier result for the sub class

Now, as you can observe in the figure above, the classifier gave a correct prediction. That was the only significant infection we could identify, but we believe there are more, however, as the training dataset was small, this one was the clear one. This was important to prove that even with a proper neural network training process, we can achieve wrong results if the training dataset is not coherent. This also proves that the training dataset is hugely important on these problems because “infections” such as the one we’ve just seen can mislead the results.

Notes Some notes about this task:

- Only 25% of the DUTA 10K dataset was online to be scraped.
- From the small percentage of 25% available domains from the DUTA 10K, a big part had already changed their content since the time they were originally classified.

- Due to the “infections” (i.e. the fact that the DUTA 10K classifications were outdated), we decided to manually find every domain of the training dataset and change the sub class manual label to “Pornography” if we found keywords such as “child-pornography”, “pornography”, “porn”, etc. Then, we used that version to train the model of the sub class classifier, which showed a significant improvement when classifying pornography domains, compared for instance, with the main class model, which we trained with the original version of the DUTA 10K.

4.2.6 Social Network Analysis

Objective The objective of using Social Network Analysis (SNA) on this dataset was to essentially have some interesting measures that quickly could give us an insight into the importance of nodes within the network and some hints on its topology.

Technology In terms of technology, we used a Python library named, NetworkX⁷, which is a package designed to study complex networks.

Preparation The preparation needed for the SNA study was basically to have a form of saving the results for each domain of the network. Therefore we added a field on the domains MySQL table for each of the metrics we wanted to measure.

Process The process was really simple as we just needed to upload all of the nodes of the network into a NetworkX structure in Python, and then run the desired algorithms to have the metrics we wanted. This package is very complete as it offers a lot of different measures with different algorithms, however, we wanted to keep it simple and meaningful, so we picked the following:

1. Centrality Measures
 - Betweenness Centrality
 - Closeness Centrality
 - Degree Centrality
 - In-Degree Centrality
 - Out-Degree Centrality
 - Eigenvector Centrality
2. Clustering Measures
 - Clustering Coefficient
3. Link Analysis

⁷More information at <https://networkx.org/>

- PageRank

All of these measures were explored in the chapter 1 section 2.4.1, but in a brief form, centrality measures are related to the importance of a node to others, clustering measures give an insight into how is organised the network in terms of groups, clusters, and finally, the link analysis, PageRank metric, is a specific case of centrality measures as it also tells the importance of a node to other but focusing in the importance of node neighbours and not only on its location and relations in the network.

Results The results were great as this package is well implemented, and the difficulty in applying it was low. We could obtain all of the measures we aimed for while leaving others out. Every metric gives its insight on the network, however, some of them give similar results and therefore, we wanted to keep it short and demonstrate that these metrics are enough to give an accurate look at the network topology, as we will see later in chapter 6.

As we know these measures are a classic form of understanding the topology and the relevance of important nodes within a network, however, it does not provide a look of the structure from the whole network. In other words, some of these measures would be the same if we changed the structure of the network. Therefore, there are some models that carry patterns to look into a network structure as a whole, and one of those models is the Core-Periphery model, in which we will focus next.

4.2.7 Core-Periphery Analysis

Objective The goal of the Core-Periphery model is to model a network into groups of nodes belonging to a core and nodes belonging to a periphery. We already studied it in chapter 1 section 2.4.2, where we learned that these structures are susceptible to different meanings depending on our characterisation of core and periphery. Here, we will apply this model and study the network at the eye of a single core-periphery structure and multiple core-periphery structures. So, after understanding if each node belongs to the core or the periphery, for the single-core periphery structure, and to which layer it belongs and its coreness, for the multiple-core-periphery structure, we will try to make a visual representation of the results.

Technology The technology used to make this study was essentially an algorithm for detecting core-periphery structures named “Bayesian Core-Periphery Stochastic Block Models” [48], which, as the name suggests, uses Bayesian inference to find and detect core-periphery structures within a giving network. This algorithm was also available as a Python library⁸ that was not available on Python’s package management system, and therefore we had to install it manually.

⁸More information at https://github.com/ryanjgallagher/core_periphery_sbm.

Preparation The preparation for this task was not much as we already had the data well structured in a MySQL database. We just had to add fields on the domains table to save the results from the algorithm for each domain.

Process The process was, just like with the social network analysis, to load the nodes into a NetworkX web graph structure and then use the Python library with the core-periphery algorithms. This library offered the possibility of changing parameters in order to find the best fit between the data and the model. We could change the following parameters:

- `n_gibbs` (number of Gibbs samples)
- `n_mcmc` (number of Markov Chain Monte Carlo steps)
- `n_layers` (number of layers - available only for the multiple core-periphery detection)

This library also had a function that allows us to test the results with a quality function and get a perception of how well the model fits the data.

So the steps to achieve the final results were the following:

1. Load all the nodes on a NetworkX web graph
2. Use random search to find the best parameters using a quality function from the library
3. Save the best parameters
4. Run the library and detect the core-periphery structure(s)
5. Save the results for each node

After that, we went to the visualisation part of the results, and we started with the single-core periphery, as it was the simplest case. So basically, we ordered the nodes by core and then periphery and also by their probability of belonging to the respective partition (metric also provided by the library). Then, we created the adjacency matrix for the network in that order and created a graph that represents the matrix by having black dots where the matrix is 1 and white dots where the matrix is 0. The results were not great, and we didn't replicate the same methodology for the multiple-core structures, yet, we found a different solution, as we will see in the next chapter in the results section.

Results The results, as already announced before, were not great for the visualisation part, however, they demonstrate an empirical difficulty which led us to find a better solution. In terms of the single core-periphery structure visualisation, the following happened:

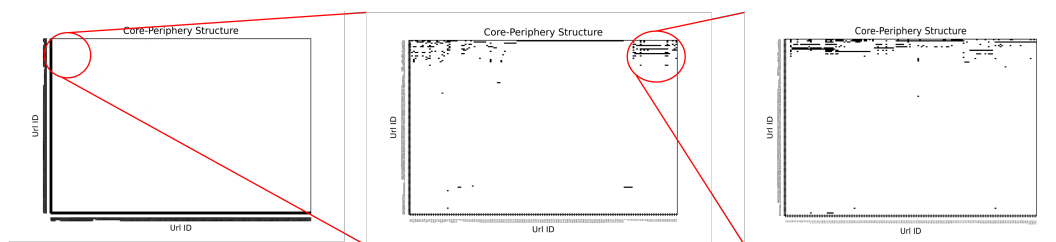


Figure 4.24: Result of the single-core periphery structure found for the dataset 1. The left graph shows the entire network, the middle one shows the first 150 nodes and the right graph is the middle graph with an offset of 150 nodes on the x axis

In the figure above, we can observe that on the left graph, where we have the entire network, it is basically a blank graph. In reality, it is not empty, but as we are trying to represent 7874 nodes by 7874, the matrix gets huge, and it is just impossible to display it in this format. Therefore, as the problem was a scaling issue, we thought that instead of visualising the entire network at once, we could just see snapshots of parts of it. So, in the middle graph, we have the part that is circled on the left graph, the first 150 nodes of the network. Here we can observe something more similar to a core-periphery structure, however, one of the advantages of the core-periphery model is that it gives a perspective from the structure of the whole network, and here we were sabotaging that in return for a way of visualising these results. If we wanted to see, for instance, what is going on at the red circle of the graph in the middle, we would need to travel there just like we did on the graph at the right of the figure, which shows the red circle of the middle graph. This is not viable, and we left this idea and decided instead to simply show a normal web graph with nodes coloured differently accordingly to their partition (core/periphery) or layer (for the multiple core-periphery structures). So that was what we did, and the results will be shown in the chapter 6 with the final results from all the approaches taken to study this dataset.

4.2.8 Domains Detection

Objective The goal of our final approach was to use the raw texts that we scraped from the available domains in our dataset and see what interesting information could we find with it. An interesting thought was that if initially the dataset provided was created by searching 114 domains (seeds) for hyperlinks, we could do the same for the hyperlinks found. We would be basically adding another column in the original excel file shared with us, figure 4.2, and have, therefore a third layer of relations between these domains.

Technology About the technology used, it was simply a Python script to find valid domains using specific regex rules.

Preparation The preparation needed before proceeding with this approach was to create a structure to save the results, and for that, we created a MySQL table named `extracted_hyperlinks` with the following fields:

- `url` - the domain discovered
- `url_id` - to identify the URL id from which the domain was discovered
- `valid_url` - to identify if the domain discovered is valid according to RFC 2396 (Uniform Resource Identifiers)⁹
- `updated_at` - last update of the entry (used for debugging)

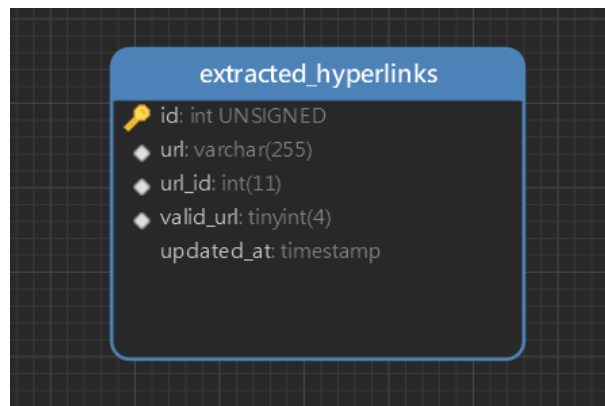


Figure 4.25: MySQL `extracted_hyperlinks` table created

Process The process used here was the following:

1. Get one URL from the hyperlinks table, one at a time.
2. Verify if that domain was scraped, and if so gather the text scraped (text + images text from OCR).
3. Parse the text scraped and identify, with some regex rules, domains.
4. For the found domains remove the path from the URL to get the domain's base URL
5. Save the result in the `extracted_hyperlinks` table

After that, we ran a script to check if the domains detected were valid according to RFC 2396 and saved the result in the `valid_url` field for the respective domain.

⁹More information at <http://www.faqs.org/rfcs/rfc2396.html>.

Results Results were interesting as this process resulted in 4537 entries in the created MySQL table, being however, just 1871 unique. This shows a lot of repeated domains within these web pages.

id	url	url_id	valid_url	updated_at
1	http://support.pornhubpremium.com	14	1	2021-07-06 14:30:09
2	http://n6doy7ndydmvdlts.onion	17	1	2021-07-06 14:30:09
3	http://bvten5svsltfpxrx172ukqxixwo2m5ek5svmcxgrmkta4tbmiemuibid.onion	56	1	2021-07-06 14:30:09
4	https://buggedplanet.info	84	1	2021-07-06 14:30:09
5	http://www.peoplesdrugstore.org	113	1	2021-07-06 14:30:10
6	http://www.nanaimogold.com	174	1	2021-07-06 14:30:10
7	http://localbitcoins.com	174	1	2021-07-06 14:30:10
8	https://bitcoinnordic.com	174	1	2021-07-06 14:30:10
9	https://en.bitcoin.it	174	1	2021-07-06 14:30:10
10	http://pastebin.com	192	1	2021-07-06 14:30:10

Figure 4.26: Portion of the extracted_hyperlinks table after searching for domains in the dataset

This analysis provided a powerful set of data that could reveal more interesting results about the domains we studied. All of these results, and others, will be shown in the chapter 6 where we compacted every study done with the dataset 1.

CHAPTER



DATASET 2

The following chapter will serve as an instrument to explain the second and last dataset studied in this project. This dataset has some particular characteristics that will give us the possibility to approach the dataset in a different form from which we approached the first dataset. Like with the first dataset, we will start by explaining how it was built and its structure, followed by the approaches we took to analyse it and extract relevant properties.

5.1 Data Collection

The second dataset given for the project will be labelled after “dataset 2”. This dataset goes a step further when compared with the process used to create the dataset 1. To explain, this dataset goes beyond the crawling process used for the first dataset as it was also used a scraper that is a software capable of extracting every content from an HTML web page. The idea here was to gather specific and detailed information about a set of domains.

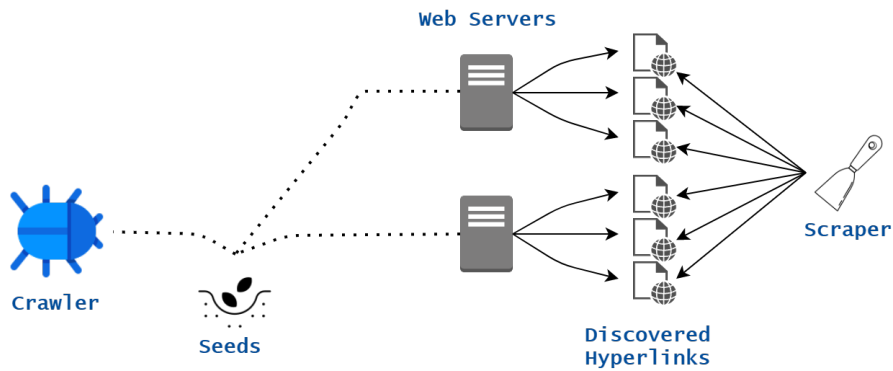


Figure 5.1: Figure 4.1 completed with the scraping process which was used for the dataset 2.

It was selected 11 domains from the dataset 1 in which they would run a scraper to extract all the information they could find. This resulted in the dataset 2, which is composed of the 11 domains, where each of them was scraped in order to retrieve all the .html pages found, all the cryptocurrencies wallets found, all the PGP¹ keys found and all the emails found.

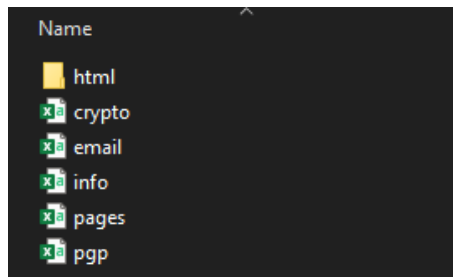


Figure 5.2: Data structure provided for each of the 11 domains scraped.

On this structure, we can observe a folder named HTML, which contains all the web pages found in the domain, a crypto excel file which contains all the cryptocurrencies wallets addresses found, an email excel file with all the emails discovered, an info excel file with some specific information about the domain itself, a pages excel file with an index for every HTML file in the HTML folder, and finally, a pgp excel file with all the PGP information discovered.

Below there are 3 figures showing a small part of the data inside the crypto, the email and the pgp excel file, respectively:

¹The Pretty Good Privacy is an encryption system that allows two users to exchange information safely by using symmetric cryptography and asymmetric cryptography techniques.

	A	B	C	D	E
1	id	address	type	discovered_at	last_discovered_at
2	19892	1BWFPuUTy8nFj6nQxU3HMaJC4pXgKjkmrX	BTC	5/14/2019 10:31	5/14/2019 10:31
3	19893	LsyorBQJchdPKBQkA7Wj6AAoJEF99z8	LTC	5/14/2019 10:37	5/15/2019 7:54
4	19894	MdR8sum4q8Sib8AHv6jfA3ASFkZnj1W8s	LTC	5/14/2019 10:38	5/14/2019 10:38
5	19896	LwkAeuGTALiZQgR6xjegG8PoJexX7	LTC	5/14/2019 11:01	5/14/2019 11:01

Figure 5.3: Crypto excel file example from the dataset 2.

	A	B	C	D
1	id	email	discovered_at	last_discovered_at
2	12367	josem29@yahoo.com	5/14/2019 10:31	5/14/2019 10:31
3	12368	yourfactory@tutanota.com	5/14/2019 10:31	5/15/2019 3:02
4	12369	NovaSupplies@unseen.is	8/15/2017 12:51	5/15/2019 9:29
5	12370	bambishop@protonmail.com	5/14/2019 10:33	5/15/2019 14:34

Figure 5.4: Email excel file example from the dataset 2.

	A	B	C	D	E	F	I	J
1	id	key	name	email	fingerprint	key_id	discovered_at	last_discovered_at
2	195	-----	admin		4CB517D67411517B1AA9C4083BE4033866023EC9	3BE4033866023EC9	5/15/2019 14:30	3/25/2020 8:50
3	194	-----	O0oo O0o		3B6101FD627836579BAD68089CC82ACCf625FCA4	9CC82ACCf625FCA4	5/15/2019 14:30	5/15/2019 14:30
4	3338	-----	admin.cm	admin.cm@protonmail.com	7750EBAACA696BE3D9778FBF5292CDC107B81647	5292CDC107B81647	6/15/2020 1:55	7/8/2020 12:15
5	3338	-----	admin.cm	admin.cm@protonmail.com	7750EBAACA696BE3D9778FBF5292CDC107B81647	5292CDC107B81647	6/15/2020 1:55	7/8/2020 12:15
6	3919	-----	Colin Cogle	colin@colincogle.name	07E884C70EA50F815CCC7CD754D78340D86C364C	54D78340D86C364C	7/9/2020 13:13	7/9/2020 13:13
7	3921	-----					7/9/2020 13:20	7/9/2020 13:20
8	3922	-----					7/9/2020 13:20	7/9/2020 13:20
9	2435	-----					5/4/2020 12:11	7/10/2020 2:51
10	2434	-----	YosemiteGhostWrite		788D73CA29B56D65676D468FF40F570ADCE20B18	F40F570ADCE20B18	5/4/2020 12:11	7/10/2020 2:51

Figure 5.5: PGP excel file example from the dataset 2.

To get familiarised with the dataset and as it was short in terms of domains, we looked in detail at every domain provided within the dataset, and we organised the domains as follows:

- Tochka Market (selling illicit goods) with more than 17 000 .html files
- 3 Directory (advertising illicit web sites) with 863 .html files
- 3 Apple Shop Market with 495 .html files
- 2 Cloned Cards Shop with a forum including 116 .html files
- 2 Others (protected by login and a blog) with just 102 .html files

5.2 Data Analysis

This was the second dataset studied in this project. As explained before, it had a particular characteristic compared with the dataset 1, as it had just a universe of 11 domains from the Dark Web. This made our approach be towards the following questions: what if someone needs to study a small set of domains? What interesting facts can they be searching for? and what should be our focus?

As you might think, some of the approaches taken for the dataset 1 could also be used here, but that is not quite correct because, for instance, why would I create an automatic classifier to study 5-20 domains when I could simply enter them and in less than a few

minutes do it manually with almost 100% accuracy. So, the idea here was to explore different natural language processing techniques and solutions from data analysis that make sense to be used on a dataset like dataset 2.

Initially, this dataset looked to be the more promising, but in the end, it turned to be not so wealthy in terms of data, and we believe this is due to the nature of it, as only one of these domains had thousands of web pages to be explored.

The first thought was, if we are studying a small set of domains and they have been probably picked with some criteria, why not make a cross-matching web graph with the specific information gathered about it, and see how these websites correlate with each other.

5.2.1 Cross-Matching Web Graph

Objective The goal of this approach was to intersect data provided in the dataset as it could give many insights about the domains relation, as we will see next.

Technology In terms of technology used to create this web graph, it was basically a web graph database named Neo4J² which we will also introduce in the next chapter when we talk about the web application we built to compact all the results from both the datasets. This software is more than a database as it also allows us to visualise, do queries on the data, integrate it with other applications, and a lot of other features. Nevertheless, for this task, we will focus on its visualisation.

Preparation The preparation needed to achieve the objective, as Neo4j allows to import data through excel files, was simply to create an excel file for each specific type of data (emails, cryptocurrency wallets and PGPs) with their unique values, as some of them were found repeatedly in multiple domains. Then, we created an auxiliary excel file that gives the relation between them.

id	host
15992	http://tochka3evlj3sxdv.onion
294314	http://4afoasf4xoxspbpf.onion
294317	http://y4dwte6vfoarzhbv.onion
294318	http://zgpuav2ywy7um6fx.onion
294319	http://cwszufwmvdwymrzo.onion
294320	http://bvn7fd3i76mm4iao.onion
294328	http://cr7s5vesqqozt7zt.onion

Figure 5.6: Portion of the structure of the domains

In the figure 5.6 we have a portion of the 11 domains studied in this dataset with their respective ID that will afterwards be used to relate a respective domain with the data found in terms of emails, cryptocurrency wallets and PGP keys.

²More information at <https://neo4j.com/>

Below we have an example of how we structured it for the emails:

id	email
1	mail@mail.com
2	HACKZUES@PROTONMAIL.
3	hackservice@secmail.pro
4	cc100best@protonmail.com
5	cc100bestwood@sigaint.org
6	cardshopff@protonmail.cor
7	linkdir@secmail.pro
8	mq4nm3obd62@torbox3uic

Figure 5.7: Portion of the structure of the emails

In the figure 5.7, just like with the structure of the domains, we have all the unique emails found in every domain of the dataset with the respective unique ID.

domain_id	email_id
15992	1
294314	2
294317	2
294318	2
294314	3
294317	3
294318	3
294350	4
294351	4
294350	5

Figure 5.8: Portion of the domain emails relation structure

Above, we have the structure that allows identifying where was a respective email found in our domains dataset. Now, we repeated the same for the cryptocurrency wallets and PGP keys in order to have this information on excel files ready to be imported to the Neo4j Database.

Process The method passed through importing the excel files we created to the Neo4j database and then process the data there to achieve our goal. In Neo4j there are no static structures such as tables in MySQL, but we can create labels that basically represent a group of nodes. Therefore, as the dataset had 11 domains with specific data about the

PGP, cryptocurrency wallets and emails, we created 4 groups of nodes with the following labels:

- Domain - 11 nodes - represent the domains in study
- PGP - 8 nodes - represent the unique Pretty Good Privacy Keys found
- Crypto - 39 nodes - represent the unique cryptocurrencies wallets addresses found
- Email - 134 nodes - represent the unique emails found

For each of these labels, we could add every information we wanted, so for instance, for the email group, we also added the respective email value and so on for the other type of nodes. In terms of relations, we can also define different types of relations for any node within the database, however, for this dataset, we just need one type of relationship that we designated as “KNOWS_DOMAIN” that will basically connect each node of PGP, Crypto and Email to the respective domain where it was found.

Results The results were great as after importing all the data, the Neo4j tool did the rest and gave us the following web graph:

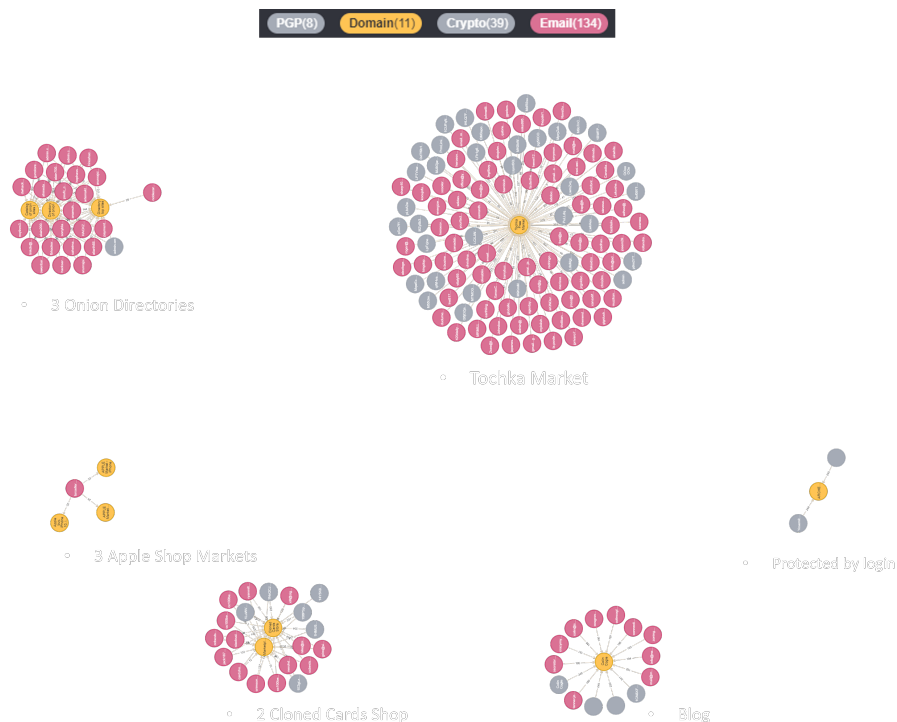


Figure 5.9: Cross-Matching Web Graph with the respective labels for each type of node. In yellow we have the 11 domains, in grey 8 PGP keys and 39 cryptocurrency wallets and in rose the 134 emails discovered.

The figure 5.9 was a success as it allowed us to easily understand how the different types of data (emails, cryptocurrency wallets and PGPs) are related to each domain. We

also visited every domain and discovered what was their content about, as mentioned before, and that information was completely coherent with these clusters of data. The Tochka Market has the biggest community and a lot of information, so we could automatically, correctly, say that this domain is the one with the biggest cluster in the figure 5.9. At the top left, we have three domains grouped that are the three directory domains. Below them, we also have three domains but with just one email, and these domains are the ones related to the Apple Shop Markets. Below the Apple Shop Markets, on the bottom left, we have two domains that are the ones about Cloned Card Shops, and on the bottom right, we have the remaining domains, the blog domain and the domain protected by login. By looking into every domain initially, we were able to identify every domain within the web graph just by looking at it, as we already knew the number of clusters and their sizes, the domains with more information and the domains with less information.

5.2.2 Name Entity Recognition (NER)

Objective The goal of this approach was to try to identify a set of entities within the domains of this dataset. The idea is that if we have a set of domains in which we want to understand more in detail what they contain, we could build a form of finding specific keywords and associate them with entities. This would afterwards allow us to have a searching system where we could look up for specific data way faster than simply search them through all the text of these domains, especially because some of these domains have thousands of web pages, each one with a lot of text.

Technology In terms of technology used here, it was also based on Python programming language with the auxiliary of an algorithm provided by a natural language processing library named spaCy³. We also tested other algorithms such as the Stanford Name Entity Recognition Algorithm⁴, however, we didn't get the same results as with the spaCy library, and we ended but using it. The spaCy NER algorithm is faster than Stanford NER, but that was not the reason to choose it over others, because in reality, the algorithm we decide to use will depend on the data we have, and spaCy results were more accurate than Standfords.

The spaCy name entity recognition algorithm works with machine learning technologies, and therefore they provided different models to run the algorithm. These models differ by language and the size of the dataset used to train the models. As the domains from the dataset were all in English, we decided to use the model named "en_core_web_lg" which is the best to run the NER with English text. This model allows the detection of the following entities:

- ORG - Companies, agencies, institutions

³Natural Language Processing library in Python. More information at <https://spacy.io/>

⁴Stanford (NER) is a software developed in Java that allows identifying specific entities from a text. More Information at <https://nlp.stanford.edu/software/CRF-NER.html>

- FAC - Buildings, airports, highways, bridges, etc
- NORP - Nationalities or religious or political groups
- GPE - Geopolitical entity, i.e. countries, cities, states
- LOC - Non-GPE locations, mountain ranges, bodies of water
- LAW - Named documents made into laws
- PERSON - People, including fictional
- PRODUCT - Objects, vehicles, foods, etc. (not services)
- MONEY - Monetary values, including unit
- EVENT - Named hurricanes, battles, wars, sports events, etc
- WORK_OF_ART - Titles of books, songs, etc
- LANGUAGE - Any named language
- ORDINAL - “first”, “second”, etc
- CARDINAL - Numerals that do not fall under another type
- QUANTITY - Measurements, as of weight or distance
- PERCENT - Percentage, including “%”
- DATE - Absolute or relative dates or periods
- TIME - Times smaller than a day

Preparation The preparation required to apply the NER algorithm passed through creating some MySQL tables to allow us to access the web pages provided by the dataset, scrape them and save their result. Then, we also needed a structure to save the results from the NER. Therefore we created the following tables:

- domains - 11 rows - identifies every domain from the dataset
- domain_web_pages - 19563 rows - identifies every web page provided in the dataset for every domain
- domain_ner - to save the results from the NER

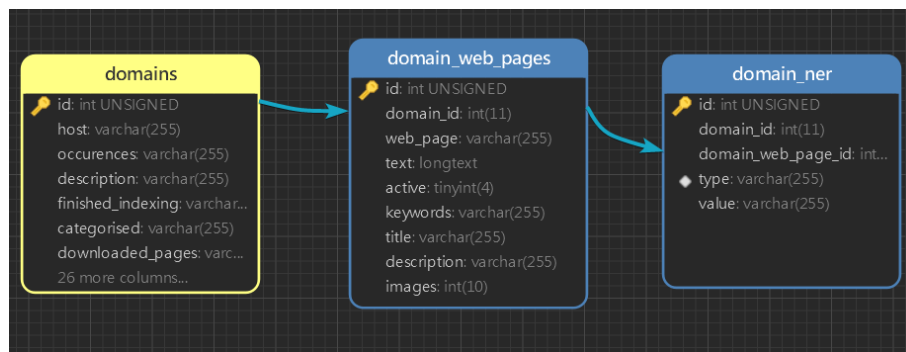


Figure 5.10: On the left, we have the domains table that will contain the 11 domains from the dataset. In the middle, we have a domain_web_pages table which contains the information provided in the pages excel file. Finally, at the right, we have the table domain_ner, which will serve as an object to store the results from the NER for the respective domain.

In the figure above, we can observe the three tables created (domains, domain_web_pages and domain_ner) that basically will serve as structures to achieve the objective. The domains table will have the domains from the dataset in study, the domain_web_pages table will have the information about each downloaded HTML web page for the respective domain (information provided in the excel file named pages as you might remember from figure 5.2 in section 5.1). The last table created, domain_ner, is the one in which we will save the NER results.

All of these tables are connected to each other by a one-to-many relation where a domain can have multiple web pages and a web page can have multiple name entity recognition results.

Process The process passed through in a simple form, scrape every HTML page available for each of the 11 domains and then run the NER for all of those web pages scraped. We could briefly describe the process in the following steps:

1. Gather a domain from the domains table
2. Gather all the web pages available for that domain from the domain_web_pages table one at the time
3. For each web page, scrape the text and save the result on a specific field of the table domain_web_pages for that web page
4. For each web page scraped, run the NER algorithm and save the result in the domain_ner table while identifying to which web page and domain that entity discovered belongs.

Results The results from the algorithm were not great as there was a lot of meaningless words classified, however, we could identify 513 654 entities in 19563 web pages from the 11 domains.

domain_id	domain_web_page_id	type	value
15992	297994	ORG	Tochka Refer Frier
15992	297994	CARDINAL	7485.14
15992	297994	PERSON	Pulver
15992	297994	TIME	11 hours ago
15992	297994	DATE	2 weeks ago
15992	297994	LAW	the MIT License P
15992	297994	CARDINAL	119.89
15992	297994	CARDINAL	0
15992	297994	CARDINAL	222
15992	297994	ORG	Tochka News Ven

Figure 5.11: Portion of the results saved on the domain_ner table with the identification of the domain and web page to which it belongs to, the type of entity and the value.

With this, we created a system that allows filtering the domains by type of entities found and search values on the domain_ner table with some efficiency. This system will be shown in the next chapter, where we will display all the refined results from the approaches taken to the first and second datasets.

WEB APPLICATION

In this chapter, we will merge all the work done during the project on a web application format. We created this web application to publicly and easily release the study online while showing how relevant this form of displaying results can be. It will also have some interactivity and an appearance of a kind of Dark Web Monitor where we can see all the information in detail about every domain, navigate through the web graphs and take our conclusions about it. We will explain the technology behind the web application and all the results obtained from its development.

The web application is publicly available at <https://darkor.org>.

6.1 Technology

The technology used to develop the web application was the following:

- Backend and Frontend built in Laravel, which is a PHP programming language framework
- JavaScript to most of the website interactivity and visualisations
- MySQL for the database
- Neo4J for the web graph database

The development of this web application was a challenge mainly for two reasons, integrate a lot of different technologies in a production environment (i.e. host Neo4J database online, host MySQL database, and host a web server), and the need of displaying a web graph with 10 000 edges and 7874 nodes. For the integration challenge, we ended up with having the Neo4J database and the Domain Name Server being hosted on Google

Cloud Platform while the web server and the MySQL database being hosted on a control panel named Plesk. For the web graph finding the best library to build it was crucial, as there are many solutions for that, and we actually tried some of the most known libraries for web graphs, such as D3 JS or Cytoscape JS. Nevertheless, both of these libraries were not as fast as we wanted, and we kept searching until we found a library named Vivagraph JS, based on a library named nGraph JS, a JavaScript package focused on drawing web graphs on a browser using Web Graphics Library. Despite being an unknown library when compared with D3 JS or Cytoscape JS, it was the one that performed better and was the one chosen for the web application.

The Neo4j database was used to simplify the process of having web graphs on the website as Neo4j offers APIs to access the data and query it remotely. The rest is quite simple to understand, a MySQL database online to store all data studied in the chapters 4 and 5 and a web server to deploy the website itself. After going through the challenging process of deploying all these services online, we were ready to start developing the application with the help of tools such as control version software like Git¹, it was easy to go through the development of this web application and have it easily updated with the web server.

6.2 Development

We structured the web application accordingly with this document, having a section about the dataset 1 and a section about the dataset 2. Here we will display and describe briefly the idea of each web page created. The structure was the following:

- <https://darkor.org/> - describes the purpose of the project and gives an intriguing overview of the study.
- <https://darkor.org/about> - describes the entities that helped the progress of this project and the people involved in it
- <https://darkor.org/login> - login web page as it was required to have some data protected
- Dataset 1
 - <https://darkor.org/dataset1/overview> - Overview of all the approaches taken to study this dataset
 - <https://darkor.org/dataset1/webgraph> - Interactive web graph of the dataset with access to every node detailed information
- Dataset 2

¹Control version software that allows to have a historical record of all the different versions from a project and also other features such as deploying a version easily into a web server. More information at <https://git-scm.com/>

- <https://darkor.org/dataset2/overview> - Overview of all the approaches taken to study this dataset
- <https://darkor.org/dataset2/webgraph> - Interactive web graph of the dataset with access to a Name Entity Recognition tool

6.3 Results

The results can be easily seen online at the project's website, however, we will describe briefly the results in which the web application was built.

In terms of the dataset 1, where results can be observed at <https://darkor.org/dataset1/overview> and at <https://darkor.org/dataset1/webgraph>, we started with a scraping process, chapter 4 section 4.2.2, where we could scrape 4985 domains as the others were offline or inaccessible. From these domains we were able to extract 40796943 words which give an average of 8183.94 words extracted per domain.

One of the things we noticed was that some domains happened to have not so much text written in the HTML code, but to have images instead of displaying some text, therefore we used an Optical Character Recognition, chapter 4 section 4.2.3, to extract those texts. About 85% of the domains had images, giving an average of 16 images per domain, and 15% of those had significant text to be recognised. This approach led to 2835 texts extracted which were valuable as we added this text to the text extracted from the scraping to increase the amount of information about each domain.

About the temporal analysis, in chapter 4 section 4.2.4, we could extract interesting statistics. The objective was to check whether these domains oscillated in terms of their availability, and after analysing them for a long period, we had 3896 domains which alternated their availability status, 1824 domains that were available every time we checked them, and 2144 that were completely inaccessible. On average, 56% of the domains from the dataset 1 were active, which lead us to conclude that the domains on this dataset are, in fact, not so stable and have indeed something that makes them oscillate in terms of availability. One interesting result obtained from the temporal script is represented in the following figure where we offer a view of the network filtered by the average availability of each node.

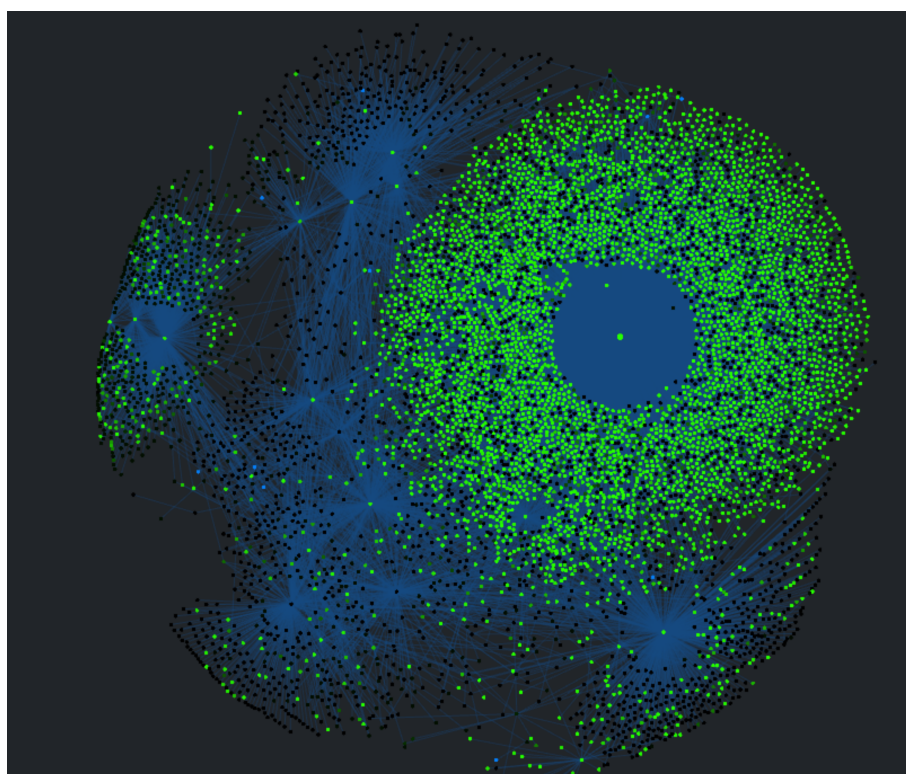


Figure 6.1: Temporal Analysis result represented by the average availability of each node. Brighter green represents nodes that have an higher average availability and darker green the ones with lower average availability.

One of the most interesting approaches we took with this dataset was the development of an automatic classifier, chapter 4 section 4.2.5. By using the text scraped either by the text available on the web page, either by the text extracted through the OCR from the images, we were able to have enough information that hopefully could be enough to classify these domains. After creating a neural network model, we were able to attribute a possible main class value and a sub class for every domain that was scraped. As we used a dataset to train our neural networks that was some years old, some problems did happen on the results. Some main classes such as “Counterfeit Credit-Cards” and “Cryptocurrency” were classified as so despite the domain being about “Pornography”. This happened because some of the “Counterfeit Credit-Cards” and “Cryptocurrency” domains from the training dataset were manually classified some years ago, and now these domains changed their content mostly to “Pornography”, which mislead the neural network models in error. Below we can observe the main class distribution for the dataset 1, which highlights the “Counterfeit Credit-Cards” as the top main class, however, we already know that these domains are mostly “Pornography” ones.

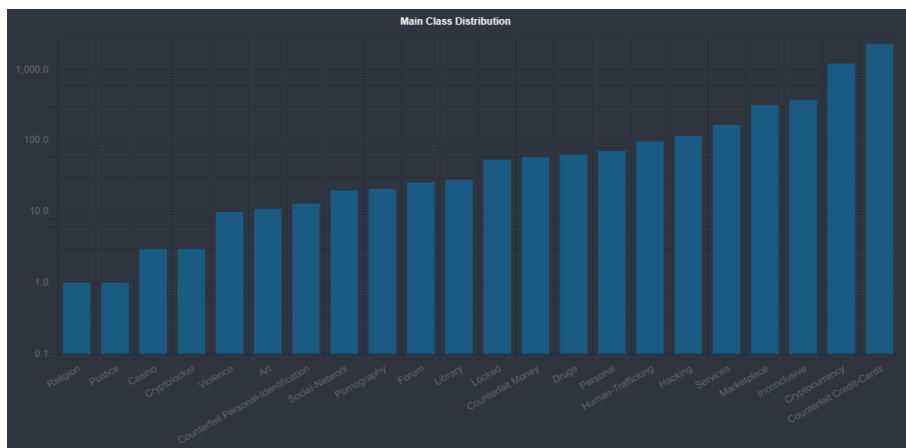


Figure 6.2: Main Class distribution from the automatic classifier applied on the dataset 1.

For the sub class, which tries to go more in detail about what a domain is about, we tried to solve some of the problems and here we tried to correct the outdated domains from the training dataset and the results were better. As we can observe in the figure below, the “Markets” sub class is the top one followed by the “Inconclusive” tag, which, like with the main class, are representing the domains that had a classification output from the neural network below the usual 0.5 on these multi-class classification problems.



Figure 6.3: Sub Class distribution from the automatic classifier applied on the dataset 1.

One of the later approaches we took at this stage was to understand the languages used in these domains. Below we have the language distribution, which shows a predominance of the English language.

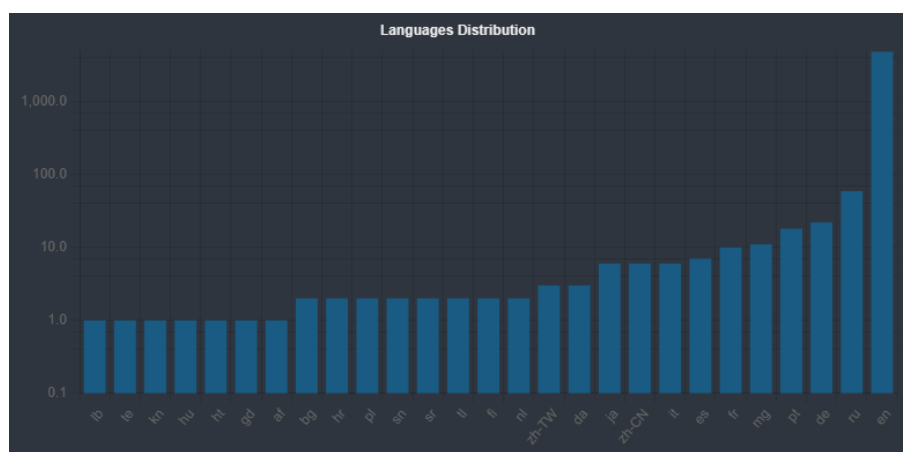


Figure 6.4: Languages distribution showing a predominance of the English language.

With an accuracy of 39% for the main class and 97% for the sub class we learned some interesting aspects with this small study on classifying domains. First, we should be careful with the training dataset we used in terms of being updated as things can change from the time the training dataset was created and the time it is used. Second, with some corrections without manipulating the data, we can suppress some of the effects of an outdated training dataset, and as we can see, for the main class we got a 39% accuracy, and for the sub class this value raised significantly to 97%, however with some overfit. The objective here was to prove that an automatic classifier can be an interesting tool to have on these problems. Nevertheless, we should never forget that they will always be susceptible to errors, and therefore we should look at them as an auxiliary helper and not as the final answer to a classification.

One study that is very common on networks is the social network analysis which we studied in chapter 4 section 4.2.6. From this study, we discovered some characteristics of the network like the link density (**0.000161**), the average clustering (**0.0544**) or the average degree (**2.54**). Apart from these global metrics of the network, we also studied for every node of the network some centrality measures, clustering measures and link analysis metrics. The results revealed that we have a structure with a low number of edges where on average, every node has 2.54 connections. The result from the average degree is misleading as it looks to the network as a whole because, in reality, we have one single node (a seed) that alone owns **5590** edges which is more than half of the relations from this dataset. Looking into the out-degree metric, we can observe that this node, with 5590 edges, has a 0.71 value for the out-degree centrality. For the in-degree metric, the maximum value seen was a node with 0.00356 in-degree centrality. Below, we can observe this node highlighted in white on the web graph we built to help us interpret the result from the social network analysis. In blue, we have the seeds, and in grey, all the other domains from the dataset.



Figure 6.5: Dataset 1 web graph with the node that owns the biggest part of the edges highlighted in white.

Another important node in this dataset is the one highlighted in white on the next figure, as it wins almost every one of the others measures from the social network analysis.



Figure 6.6: Dataset 1 web graph with the node that has the top measure for Closeness Centrality, Eigenvector Centrality, Page Rank and in-degree Centrality.

The node highlighted in figure 6.6 is definitely one of the most interesting ones in this dataset as it is also a seed that, compared with the seed depicted in figure 6.5, only have 24 edges.

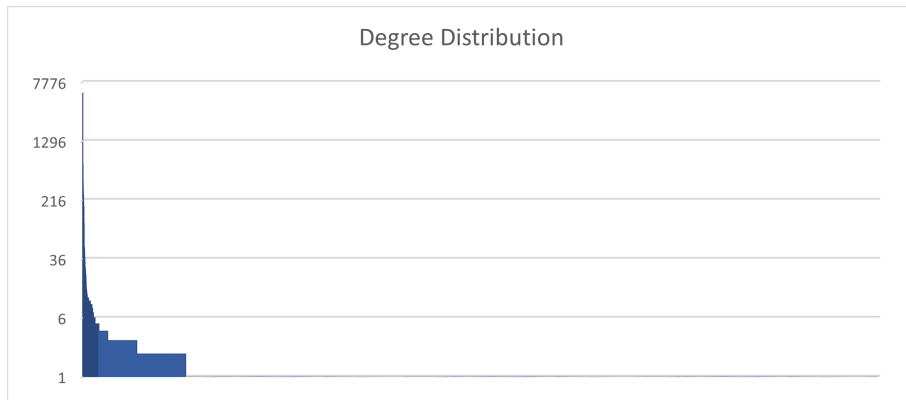


Figure 6.7: Degree distribution from the dataset 1 network showing similarities with a power law distribution.

The social network analysis is really important as it offers an excellent background from a network in terms of its topology. Nevertheless, there are different forms of understanding a network structure, such as network models like the Core-Periphery analysis. In the chapter 4 section 4.2.7, we explained how we made this study in the dataset 1 and why is it important. The algorithm used gave some interesting results, but for that, we had to build a form of visualising it as the size of the network is considerable.

In the next figure, we can observe the results for the core-periphery analysis, on its discrete variant, displayed in a web graph with cores coloured in red and periphery nodes coloured in a faded red.



Figure 6.8: Core-periphery result of the single layer variant (discrete model). Red represent nodes belonging to the core while faded red nodes represent the periphery.

The results make much sense if considering the web graph, because almost any central node creating a cluster, was positioned by the core-periphery model as a node belonging to the core. We got 12 cores and 7862 periphery nodes that show how small is the core. As

mentioned before, this dataset was created by looking into 114 domains and discovering more domains from those, so, a first thought could be that those 114 nodes could be cores of the network, but the core-periphery only marks 12 cores that, not surprisingly, are all domains from that 114 list.

In terms of the continuous model from the core-periphery study, according to the analysis done on the chapter and section mentioned previously, we obtained the following representation:

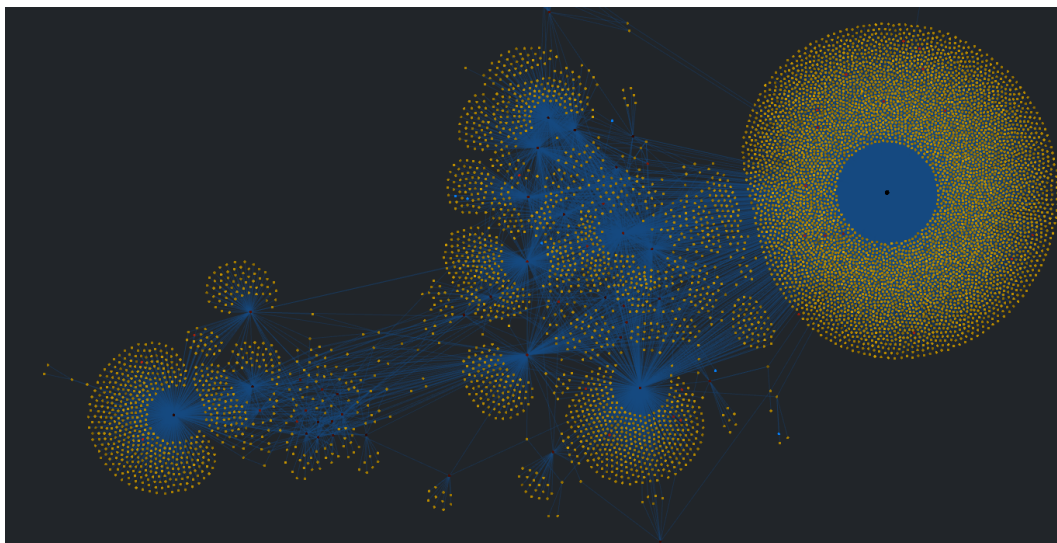


Figure 6.9: Core-periphery result of the multiple layer variant (continuous model). Red represent the nodes belonging to the first core-periphery pair and yellow represent the nodes belonging to the second core-periphery pair.

This representation has two main colours: red and yellow. In red, we have different fades of colours that represent its coreness. So darker red has a higher coreness value than a lighter red, and the same for the yellow, which represents the second layer. The multiple core-periphery model gives a lot more information because we can distinguish which nodes are more important to the core than others. Interestingly, the second layer is way bigger than the first one and only owns nodes that do not form clusters on the network. On the other hand, the first layer has the most important nodes of the network when it comes to the social network measures, and therefore it owns almost every seed within the network. On the first layer (red nodes), we have **72** nodes, and on the second layer, the remaining **7802** nodes.

A final interesting result we studied from this dataset was discovering new domains within the text scraped from the domains analysed. We tried to create a new layer of relations on this dataset, and we could conclude that the domains found by the seeds had **532** domains pointing towards the clear web and **3987** domains towards the Dark Web.

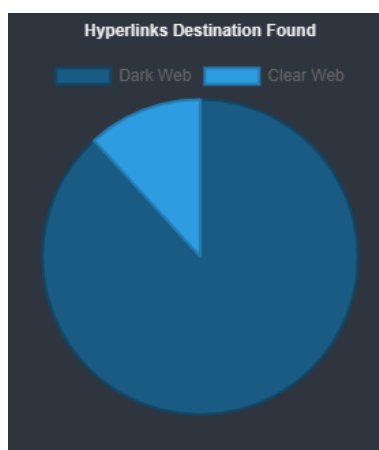


Figure 6.10: Hyperlinks destination found on the texts scraped from the domains analysed in the dataset 1.

Just 11.77% of the domains, discovered on the texts scraped, pointed to the clear web, which reveals that on this dataset, the majority of the domains have more connections and relations with other Dark Web domains, nevertheless we cannot ignore the small percentage that gives users a way out from the Dark Web in direction to the surface web.

Now changing to the dataset 2, where results can be observed at <https://darkor.org/dataset2/overview> and at <https://darkor.org/dataset2/webgraph>, we created different visualisations and tools to dig into the data analysed. Below we have the cross-matching information web graph that allowed us to conclude a lot about these domains. This study can be seen in more detail in chapter 5 section 5.2.1.

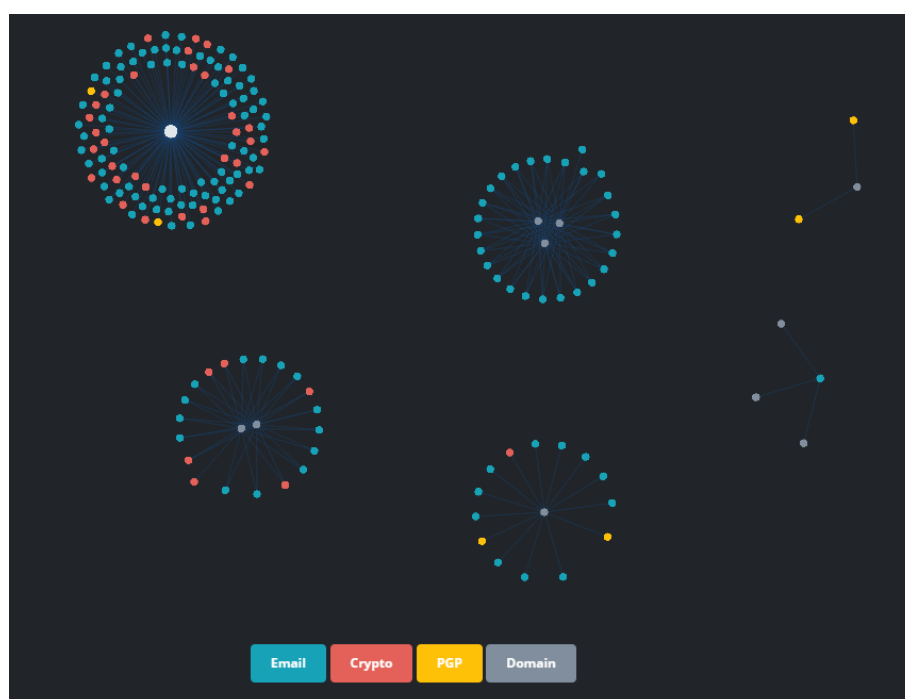
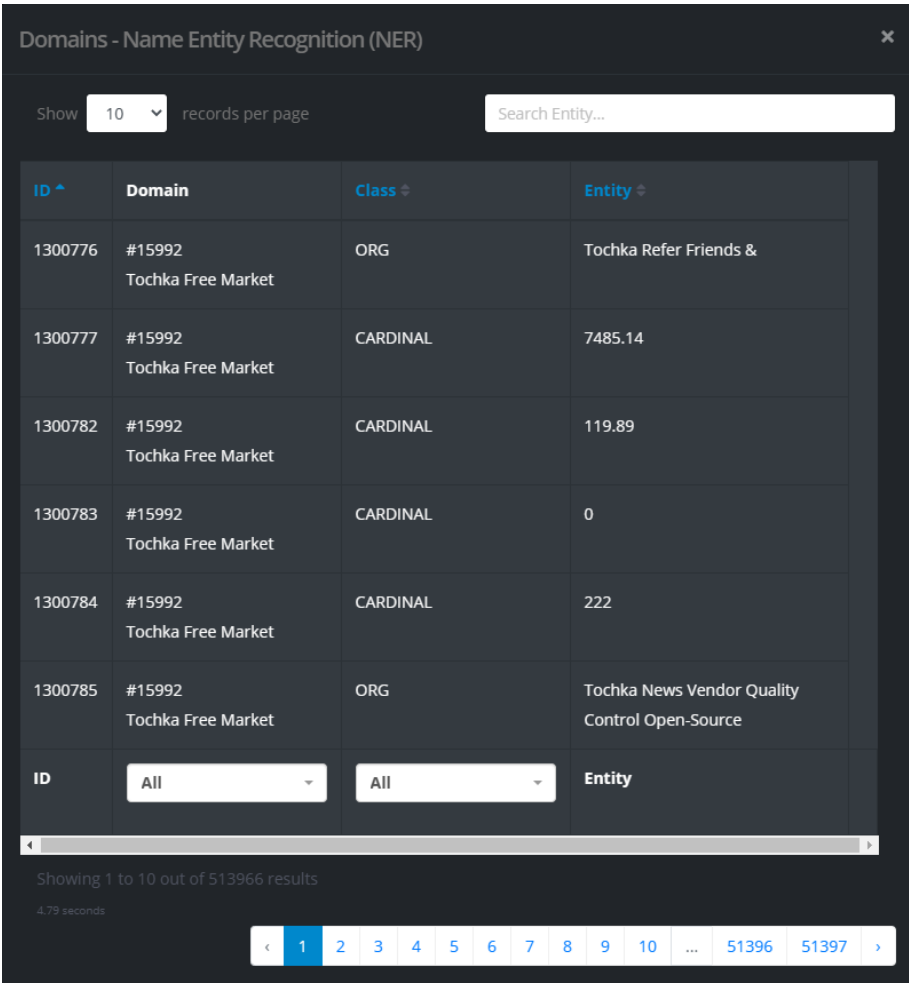


Figure 6.11: Cross-Matching information web graph of the dataset 2.

In the figure 6.11 we can see the 11 domains from the dataset 2 in grey and the detailed information represented by the nodes coloured in blue, red and yellow (email, cryptocurrency addresses and PGP keys, respectively). As the dataset is not big, in terms of domains, we could conclude a lot from the figure above.

- We have 3 domains in the middle surrounded by emails which led us to think that those domains might be related in terms of content as they have the same “customers”.
- On the top left corner of the figure, we have a single domain surrounded by a lot of emails, PGP keys and cryptocurrency wallets, which by looking into this dataset and comparing with the other clusters induced us, correctly, to believe that this domain has the biggest community.
- On the far right, we have 1 domain connected to two PGP keys, and below it, 3 domains connected by a single email. Those are hard to understand what they represent, but we can interpret the 1 domain alone connected to just 2 PGP keys as a web page protected (i.e. login required), and the 3 domains connected by the same email as the email of someone important for those 3 domains.
- On the bottom part of the figure, we have two clusters, one formed by 2 domains and the other by a single domain. This says again that those clusters are made of possible customers using similar web pages to full fill their needs.

Another interesting study was the detection of entities names on the text scraped from the HTML pages provided in the dataset for each of the domains. As explained in chapter 5 section 5.2.2, the algorithm used allowed the detection of 18 different entities. The objective was to create a form of indexing information quicker with the help of filters that would be the entities.



Domains - Name Entity Recognition (NER)

Show records per page

ID ^	Domain	Class	Entity
1300776	#15992 Tochka Free Market	ORG	Tochka Refer Friends &
1300777	#15992 Tochka Free Market	CARDINAL	7485.14
1300782	#15992 Tochka Free Market	CARDINAL	119.89
1300783	#15992 Tochka Free Market	CARDINAL	0
1300784	#15992 Tochka Free Market	CARDINAL	222
1300785	#15992 Tochka Free Market	ORG	Tochka News Vendor Quality Control Open-Source

ID Entity

Showing 1 to 10 out of 513966 results
4.79 seconds

< 1 2 3 4 5 6 7 8 9 10 ... 51396 51397 >

Figure 6.12: Name Entity Recognition indexation tool.

The figure 6.12 shows the tool developed that has more than 500 thousand entities and is capable of being filtered by an entity and searched by inputting any keywords.

This was a brief description of some of the visualisations and results displayed at the web application, however, we invite the reader to visit the web application and play with the web graphs or explore the name entity tool.

CONCLUSION

A project of this nature, that is based on giving datasets, is propitious to have good results or bad results depending on the “quality” of the datasets, however, it should not affect the quality of the project, because, in the end, we are studying real data that is not perfect and susceptible to noise. This thought is important as we should not give the responsibility to a dataset for the quality of a project, but we must understand the importance and the limitations that a dataset can inflict on a project. On this project, we had two datasets that prove this thought. The first dataset (chapter 4) was limited as it only had URLs and the relation between them (which URL found which URL), however, we were able to increase the information about this dataset, which allowed us to make a deeper study on it. On the other hand, the dataset 2 (chapter 5), which was made of specific information from 11 domains belonging to the Dark Web, was small in terms of domains but huge in terms of information, as we had all the HTML pages from each of the domains and some specific information found on these domains such as emails, cryptocurrency wallet addresses and PGP's keys.

One of our focuses was also to show that we are surrounded with a lot of technologies, algorithms and data models that can be applied to these types of problems, nevertheless, it is important to understand which are relevant to be applied here and that was an important criterion we had in mind when it came to choosing interesting approaches to take for both datasets. As explained before, in the chapter 1 section 1.2, these datasets give a different vision from the Dark Web, one is way broad and tries to show how is the structure of these darknets (dataset 1), while the other is more targeted on specific domains which make us looking into it with a different vision (dataset 2).

For the first dataset we scraped all of the domains (chapter 4 section 4.2.2), while using Optical Character Recognition (OCR) (chapter 4 section 4.2.3) to also extract text from images, to increase the information about each domain. We created a temporal

analysis script (chapter 4 section 4.2.4) to check how was the variability of the availability of each domain. We developed an automatic classifier (chapter 4 section 4.2.5) using machine learning neural networks to, with the text scraped from the domains, give possible classifications about each domain's content. We also computed the usual social network measures (chapter 4 section 4.2.6) and applied a Core-Periphery model (chapter 4 section 4.2.7) to the network.

For the second dataset we created a cross-matching web graph (chapter 5 section 5.2.1) to reveal which information was common between domains and we also applied a Name Entity Recognition algorithm (chapter 5 section 5.2.2) to give a form of indexing words easily with some predefined labels.

After the independent analysis of each dataset, we created a web application to merge all the results from every technique and approach developed. The idea was to create a platform that is accessible to anyone and could display the project in an understandable and natural form with the help of relevant visualisations. The majority of the domains analysed were about pornography which also shows one of the biggest types of content available on the Dark Web.

The web application allowed us to summarise results and combine them into a coherent vision of our datasets. We found that images are very relevant in these domains, and they can be an interesting object of study for future work. The biggest part of these domains changed their availability multiple times when we analysed them, which proves that this can be an important characteristic from the Dark Web to study in the future. This study helped to reveal some of the characteristics of the Dark Web, studied in previous chapters, such as the fact that these domains have low in-degree centrality. The dataset edges distribution also showed something similar to a power-law distribution which is something seen before as one of the main characteristics of the Dark Web. This is also related to the fact that these domains are hard to find as compared with the domains living on the surface web.

In terms of challenges, we had a few during the project. However, the hardest ones were the development of the automatic classifier, since it was difficult to manage the problems associated with the training dataset and the development of a form of displaying the web graph of the dataset 1 on a web browser with some responsiveness and quickness, due to the fact of we're talking about a web graph with 10 000 edges and almost 8000 nodes. We add the extra difficulty of changing the colour dynamically according to some filters for visualising the availability of the network, categorisation and Core-Periphery analysis.

There are obviously some possible improvements to highlight. Some of them were already mentioned previously, but there are some other interesting aspects that can be important for future work. Using clustering and communities detection algorithms could be something interesting to study as it could be a different way for categorising domains and reducing a big and complex network into understandable and manageable groups. Something also interesting related to categorising domains could be using similarity

algorithms to understand how different or similar these domains are from each other. Another final optimisation note could be towards the framework used here because we were given two different types of datasets, and we worked them in a generalised way that we believe could be transformed into an automatic framework tool to apply the same analysis on any given dataset. This framework could also culminate the results on an interactive web application just like this project culminated on.

We hope that our line of thinking was well explained through the document, as the study was also driven by the “obvious” paths to choose for every step of the analysis. Obviously, our final remarks from the study can not have some direct assumptions of the Dark Web structure or its topology as we only had access to a small part of it. Nevertheless, according to all the studies we read, like the ones from the chapter 3, we can observe that some of theirs conclusions were also seen on this dataset which makes our project also coherent with other studies from the Dark Web. We observed the fact that Dark Web domains are certainly harder to find when compared with domains from the surface web, we observed the common power-law distribution for the network social network analysis degree metric distribution, and we also observed something interesting in this project related to the Literature Review Method article, described in chapter 3 section 3.4, where unconsciously we ended up using the same techniques described there as the ones most used on these type of projects, Network Analysis Methodologies and Marketplace Scraping. On our project, we did not only ended using both techniques described as the most used ones on the article, but we also used the same architectural framework analysis to proceed with the project: Data Collection, Data Preprocessing, Data Processing and Results. This was not surprising as these types of problems almost drive us on that logical path.

To finish our final thoughts, we also believe in the power of using web applications as monitors to display achieved results during these types of projects as it can offer a form of passing a message about a study easily, intriguingly and most importantly, understandably. We demonstrated the power of using different technologies such as machine learning neural networks, natural language processing, images processing, simple python scripts and web application technologies to create a final product that, more than giving interesting knowledge about the datasets in study, gives a completed perspective of the project from the beginning where we just had a few excel files and some HTML pages to a whole platform with a lot of data analysis results and incredible visualisations.

BIBLIOGRAPHY

- [1] M. W. Al-Nabki, E. Fidalgo, E. Alegre, and L. Fernández-Robles. “ToRank: Identifying the most influential suspicious domains in the Tor network.” In: *Expert Systems with Applications* 123 (2019), pp. 212–226. ISSN: 0957-4174. DOI: <https://doi.org/10.1016/j.eswa.2019.01.029>. URL: <https://www.sciencedirect.com/science/article/pii/S0957417419300296>.
- [2] A. Alharbi, M. Faizan, W. Alosaimi, H. Alyami, A. Agrawal, R. Kumar, and R. A. Khan. “Exploring the Topological Properties of the Tor Dark Web.” In: *IEEE Access* 9 (2021), pp. 21746–21758. DOI: [10.1109/ACCESS.2021.3055532](https://doi.org/10.1109/ACCESS.2021.3055532).
- [3] A. AlQahtani and E.-S. El-Alfy. “Anonymous Connections Based on Onion Routing: A Review and a Visualization Tool.” In: *Procedia Computer Science* 52 (Dec. 2015). DOI: [10.1016/j.procs.2015.05.040](https://doi.org/10.1016/j.procs.2015.05.040).
- [4] “Anonymity communication VPN and Tor: a comparative study.” In: *Journal of Physics: Conference Series* 983 (Mar. 2018), p. 012060. DOI: [10.1088/1742-6596/983/1/012060](https://doi.org/10.1088/1742-6596/983/1/012060).
- [5] Arma. *Possible upcoming attempts to disable the Tor network*. 2014. URL: <https://blog.torproject.org/possible-upcoming-attempts-disable-tor-network>.
- [6] F. Astika, I. Nadhori, and B. Barry. “Detecting and blocking onion router traffic using deep packet inspection.” In: *2016 International Electronics Symposium (IES)* (Sept. 2016), pp. 283–288. DOI: [10.1109/ELECSYM.2016.7861018](https://doi.org/10.1109/ELECSYM.2016.7861018).
- [7] T. Baumeister, Y. Dong, Z. Duan, and G. Tian. “A Routing Table Insertion (RTI) Attack on Freenet.” In: *Proceedings of the 2012 ASE International Conference on Cyber Security, CyberSecurity 2012* (Dec. 2012), pp. 8–15. DOI: [10.1109/CyberSecurity.2012.8](https://doi.org/10.1109/CyberSecurity.2012.8).
- [8] Bee. *How Does Tor Really Work? The Definitive Visual Guide*. 2020. URL: <https://skerritt.blog/how-does-tor-really-work/>.
- [9] M. Bernaschi, A. Celestini, S. Guarino, and F. Lombardi. “Exploring and Analyzing the Tor Hidden Services Graph.” In: *ACM Trans. Web* 11.4 (July 2017). ISSN: 1559-1131. DOI: [10.1145/3008662](https://doi.org/10.1145/3008662). URL: <https://doi.org/10.1145/3008662>.

- [10] A. Biryukov, I. Pustogarov, F. Thill, and R. Weinmann. "Content and Popularity Analysis of Tor Hidden Services." In: *2014 IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW)* (2014), pp. 188–193. DOI: [10.1109/ICDCSW.2014.20](https://doi.org/10.1109/ICDCSW.2014.20).
- [11] A. Biryukov, I. Pustogarov, and R. Weinmann. "Trawling for Tor Hidden Services: Detection, Measurement, Deanonimization." In: *2013 IEEE Symposium on Security and Privacy* (2013), pp. 80–94. DOI: [10.1109/SP.2013.15](https://doi.org/10.1109/SP.2013.15).
- [12] S. Borgatti and M. Everett. "Models of Core/Periphery Structures." In: *Social Networks* 21 (Nov. 1999), pp. 375–395. DOI: [10.1016/S0378-8733\(99\)00019-2](https://doi.org/10.1016/S0378-8733(99)00019-2).
- [13] J. P. Boyd, W. J. Fitzgerald, M. C. Mahutga, and D. A. Smith. "Computing continuous core/periphery structures for social relations data with MINRES/SVD." In: *Social Networks* 32.2 (2010), pp. 125–137. ISSN: 0378-8733. DOI: <https://doi.org/10.1016/j.socnet.2009.09.003>. URL: <https://www.sciencedirect.com/science/article/pii/S0378873309000513>.
- [14] Brian N. Levine, Marc Liberatore, Brian Lynn, and Matthew Wright. "Statistical Detection of Downloaders in Freenet." In: *Proceedings of the Third IEEE International Workshop on Privacy Engineering* (2017).
- [15] Cburnett. *Illustration of the topology of a generic Artificial Neural Network (ANN)*. URL: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg.
- [16] ChaTo. *Scale-free network sample*. 2006. URL: https://en.wikipedia.org/wiki/File:Scale-free_network_sample.png.
- [17] N. Christin. "Traveling the Silk Road: A Measurement Analysis of a Large Anonymous Online Marketplace." In: *Proceedings of the 22nd International Conference on World Wide Web* (July 2012).
- [18] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley. "Protecting free expression online with Freenet." In: *IEEE Internet Computing* 6.1 (2002), pp. 40–49. DOI: [10.1109/4236.978368](https://doi.org/10.1109/4236.978368).
- [19] E. Conrad, S. Misener, and J. Feldman. "Chapter 6 - Domain 5: Cryptography." In: *CISSP Study Guide (Second Edition)* (2012). Ed. by E. Conrad, S. Misener, and J. Feldman, pp. 213–255. DOI: <https://doi.org/10.1016/B978-1-59749-961-3.00006-6>. URL: <https://www.sciencedirect.com/science/article/pii/B9781597499613000066>.
- [20] F. Della Rossa, F. Dercole, and C. Piccardi. "Profiling core-periphery network structure by random walkers." In: *Scientific reports* 3 (Mar. 2013), p. 1467. DOI: [10.1038/srep01467](https://doi.org/10.1038/srep01467).
- [21] R. Dingledine, N. Mathewson, and P. Syverson. "Tor: The Second-Generation Onion Router." In: *Paul Syverson* 13 (June 2004).

- [22] M. Domenico and A. Arenas. “Modeling Structure and Resilience of the Dark Network.” In: *Physical Review E* 95 (Dec. 2016). DOI: [10.1103/PhysRevE.95.022313](https://doi.org/10.1103/PhysRevE.95.022313).
- [23] J. R. Douceur. *The Sybil Attack*. Ed. by P. Druschel, F. Kaashoek, and A. Rowstron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260. ISBN: 978-3-540-45748-0.
- [24] Duijn, A. C. Paul, Kashirin, Victor, Sloot, and M. A. Peter. “The relative ineffectiveness of criminal network disruption.” In: *Scientific reports* (2014), pp. 4238–4238.
- [25] A. Elliott, A. Chiu, M. Bazzi, G. Reinert, and M. Cucuringu. “Core–periphery structure in directed networks.” In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 476.2241 (2020), p. 20190783. DOI: [10.1098/rspa.2019.0783](https://doi.org/10.1098/rspa.2019.0783). eprint: <https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.2019.0783>. URL: <https://royalsocietypublishing.org/doi/abs/10.1098/rspa.2019.0783>.
- [26] X. Fu and Z. Ling. “One Cell is Enough to Break Tor’s Anonymity.” In: *Black Hat DC 2009* (2009).
- [27] J. Golbeck. *Chapter 3 - Network Structure and Measures*. Ed. by J. Golbeck. Boston: Morgan Kaufmann, 2013, pp. 25–44. ISBN: 978-0-12-405531-5. DOI: <https://doi.org/10.1016/B978-0-12-405531-5.00003-1>. URL: <https://www.sciencedirect.com/science/article/pii/B9780124055315000031>.
- [28] V. Griffith, Y. Xu, and C. Ratti. “Graph Theoretic Properties of the Darkweb.” In: *ArXiv abs/1704.07525* (2017).
- [29] Y. He, L. Hu, and R. Gao. “Detection of Tor Traffic Hiding Under Obfs4 Protocol Based on Two-Level Filtering.” In: *2019 2nd International Conference on Data Intelligence and Security (ICDIS)* (2019), pp. 195–200. DOI: [10.1109/ICDIS.2019.00036](https://doi.org/10.1109/ICDIS.2019.00036).
- [30] M. Henzinger. “Link Analysis in Web Information Retrieval.” In: *IEEE Data Eng. Bull.* 23 (Jan. 2000), pp. 3–8.
- [31] B. Kamil. “Dealer, Hacker, Lawyer, Spy. Modern Techniques and Legal Boundaries of Counter-cybercrime Operations.” In: *The European Review of Organised Crime* 2 (July 2015), pp. 25–50. ISSN: 2312-1653.
- [32] Katmagic. *Shallot*. 2012. URL: <https://github.com/katmagic/Shallot>.
- [33] S. Kaur and S. Randhawa. “Dark Web: A Web of Crimes.” In: *Wireless Personal Communications* 112 (Jan. 2020). DOI: [10.1007/s11277-020-07143-2](https://doi.org/10.1007/s11277-020-07143-2).
- [34] S. Kojaku and N. Masuda. “Finding multiple core-periphery pairs in networks.” In: *Phys. Rev. E* 96 (5 2017), p. 052313. DOI: [10.1103/PhysRevE.96.052313](https://doi.org/10.1103/PhysRevE.96.052313). URL: <https://link.aps.org/doi/10.1103/PhysRevE.96.052313>.

- [35] M. Koster. *A Standard for Robot Exclusion*. URL: <http://www.robotstxt.org/orig.html>.
- [36] A. Kumar and E. Rosenbach. "The Truth about the Dark Web." In: *Finance Development* 56.3 (2019). ISSN: 0145-1707. DOI: 10.5089/9781498316040.022.
- [37] G. L'Huillier, H. Alvarez, S. Ríos, and F. Aguilera. "Topic-Based Social Network Analysis for Virtual Communities of Interests in the Dark Web." In: *SIGKDD Explor. Newsl.* 12 (Mar. 2011), pp. 66–73. DOI: 10.1145/1964897.1964917.
- [38] L. Lymberopoulos, S. Kafetzoglou, and M. Grammatikou. *Deliverable D.6.1: ARGU-GRID Platform Design*. Jan. 2021.
- [39] B. Monk, J. Mitchell, R. Frank, and G. Davies. "Uncovering Tor: An Examination of the Network Structure." In: *Security and Communication Networks* 2018 (May 2018), pp. 1–12. DOI: 10.1155/2018/4231326.
- [40] S. Nazah, S. Huda, J. Abawajy, and M. M. Hassan. "Evolution of Dark Web Threat Analysis and Detection: A Systematic Approach." In: *IEEE Access* 8 (2020), pp. 171796–171819. DOI: 10.1109/ACCESS.2020.3024198.
- [41] H. Neal. *SVG Diagram of the "Onion Routing" Principle*. 2008. URL: https://commons.wikimedia.org/wiki/File:Onion_diagram.svg.
- [42] N. Negi. "Comparison of Anonymous Communication Networks-Tor, I2P, Freenet." In: *International Research Journal of Engineering and Technology (IRJET)* 4.7 (2017), pp. 2542–2544.
- [43] S. Norden. *HOW THE INTERNET HAS CHANGED THE FACE OF CRIME*. 2013. URL: <https://fgcu.digital.flvc.org/islandora/object/fgcu/%3A21423>.
- [44] I. Pete, J. Hughes, Y. T. Chua, and M. Bada. "A Social Network Analysis and Comparison of Six Dark Web Forums." In: *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)* (2020), pp. 484–493. DOI: 10.1109/EuroSPW51379.2020.00071.
- [45] E. Phillips, J. Nurse, M. Goldsmith, and S. Creese. *Extracting Social Structure from DarkWeb Forums*. 2015.
- [46] K. Poulsen. *Visit the Wrong Website, and the FBI Could End Up in Your Computer*. 2014. URL: <https://www.wired.com/2014/08/operation-torpedo/>.
- [47] X. Que, F. Checconi, F. Petrini, and J. A. Gunnels. "Scalable Community Detection with the Louvain Algorithm." In: *2015 IEEE International Parallel and Distributed Processing Symposium* (2015), pp. 28–37. DOI: 10.1109/IPDPS.2015.59.
- [48] Ryan Gallagher, Jean-Gabriel Young, and Brooke Welles. "A clarified typology of core-periphery structure in networks." In: *Science Advances* 7.12 (2021). DOI: 10.1126/sciadv.abc9800. eprint: <https://advances.sciencemag.org/content/7/12/eabc9800.full.pdf>. URL: <https://advances.sciencemag.org/content/7/12/eabc9800>.

-
- [49] S. Ríos and R. Muñoz. “Dark Web portal overlapping community detection based on topic models.” In: *ISI-KDD ’12* (Aug. 2012). DOI: 10.1145/2331791.2331793.
- [50] S. Sarkar, M. Almukaynizi, J. Shakarian, and P. Shakarian. “Predicting enterprise cyber incidents using social network analysis on the darkweb hacker forums.” In: *CyCon U.S.* (Nov. 2018).
- [51] Schullz. *Small world network example*. 2013. URL: <https://commons.wikimedia.org/wiki/File:Small-world-network-example.png>.
- [52] M. Schäfer, M. Fuchs, M. Strohmeier, M. Engel, M. Liechti, and V. Lenders. “Black-Widow: Monitoring the Dark Web for Cyber Security Information.” In: *2019 11th International Conference on Cyber Conflict (CyCon)* 900 (2019), pp. 1–21. DOI: 10.23919/CYCON.2019.8756845.
- [53] D. Seaton. *Understanding The Dark Web*. 2020. URL: <https://cyberauditteam.com/blog/identify/understanding-the-dark-web>.
- [54] The Tor Project. *Run Tor Bridges to Defend the Open Internet*. 2021. URL: <https://blog.torproject.org/run-tor-bridges-defend-open-internet>.
- [55] The Tor Project. *Tor at the Heart: Bridges and Pluggable Transports*. 2021. URL: <https://blog.torproject.org/tor-heart-bridges-and-pluggable-transports>.
- [56] The Tor Project. *Tor Metrics - Onion Services*. 2021. URL: <https://metrics.torproject.org/hidserv-dir-onions-seen.html>.
- [57] The Tor Project. *Tor Metrics - Servers*. 2021. URL: <https://metrics.torproject.org/networksize.html>.
- [58] The Tor Project. *Tor Metrics - Users*. 2021. URL: <https://metrics.torproject.org/userstats-relay-country.html>.
- [59] A. Turner. *Artificial Neural Networks*. 2015. URL: <http://andrewjamesturner.co.uk/ArtificialNeuralNetworks.php>.
- [60] P. Winter, A. Edmundson, L. M. Roberts, A. Dutkowska-Zuk, M. Chetty, and N. Feamster. “How Do Tor Users Interact with Onion Services?” In: *Proceedings of the 27th USENIX Conference on Security Symposium*. SEC’18 (2018), 411–428.
- [61] R. Yang, L. Zhuhadar, and O. Nasraoui. “Bow-tie decomposition in directed graphs.” In: *14th International Conference on Information Fusion* (2011), pp. 1–5.
- [62] A. T. Zulkarnine, R. Frank, B. Monk, J. Mitchell, and G. Davies. “Surfacing collaborated networks in dark web to find illicit and criminal content.” In: *2016 IEEE Conference on Intelligence and Security Informatics (ISI)* (2016), pp. 109–114. DOI: 10.1109/ISI.2016.7745452.
- [63] O. Çelik. “A Research on Machine Learning Methods and Its Applications.” In: *Journal of Educational Technology and Online Learning* (Sept. 2018), pp. 25–45. DOI: 10.31681/jeto1.457046.

