Hackers Are Moving

A Quantitative Analysis of Darknet Markets Products Related to Cyber Security

Master Thesis

Ricardo Yaben Lopezosa

Aalborg University
Center for Communication, Media and Information Technologies

AALBORG UNIVERSITY

STUDENT REPORT

Department of Electronic Systems
Aalborg University
A.C. Meyers Vænge 15
DK - 2450 København SV
http://www.aau.dk

Title:

Hackers Are Moving: A Quantitative Analysis of Darknet Markets Products Related to Cyber Security

Theme:
Cyber Security

Project Period:
Spring Semester 2022

Project Group:
1

Participant(s):
Ricardo Yaben Lopezosa

Supervisor(s):
Emmanouis Vasilomanolakis
Dimitris Georgoulias

Copies: 1

Page Numbers: 99

Date of Completion:
June 2, 2022

Abstract:

Recent global events such as COVID-19 have boosted our use of technology, however, cyber security is not yet a priority for most. Meanwhile, cybercriminals profit of this situation by selling their malware and our hacked information on Darknet Markets (DNMs). To date, research has tended to focus on the importance of drugs in DNMs rather than in Cyber Security-Related Products (CSRP). However, few researchers have been able to draw methods to identify CSRP in DNMs. This thesis collects data from DNMs to carry out a quantitative analysis of CSRPs. The data is collected using our own crawler called Midnight Sea (MS), and labelled using models of Natural Language Processing (NLP). Contrary to common belief, the results revealed that CSRPs are inexpensive, showing that products such as ransomware can be bought for less than 20 EUR.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

While we accelerate digitisation and increment technological dependence, cyber security is relegated to second place in our list of priorities. The COVID-19 pandemic highlighted some of the consequences of overlooking cyber security and information security, with organisations and agencies rushing digital transformations to adapt their businesses. This situation has a general weakening effect, introducing new expenses, unplanned solutions, and ever-changing security and privacy requirements. On the other hand, cybercriminals have found a lucrative sector with a low entry barrier and the opportunity to commercialise their resources on digital black marketplaces, commonly known as **Darknet Markets (DNMs)**.

DNMs are *hidden* in the **Darknet**. The *Darknet* is the collection of proxy anonymising networks (also called *Anonymous Networks*) which can only be accessed through additional software, featuring privacy and security enhancements to harden the anonymity of their users and services. Additionally, DNMs further strengthen the anonymity of their user base, for example, by limiting their payment methods to crypto currencies only and, in most cases, insisting on using additional encryption methods for their internal communications, e.g. Pretty Good Privacy (PGP).

Researchers suggest that DNMs have a short lifespan of one to three years on average [1], and once a DNM perishes, the market remains unavailable. Moreover, Law Enforcement Agencies (LEA) regularly conducts infiltrating and seizing operations to disrupt criminal activities in DNMs. All these factors reduce the popularity of the other DNMs, as they struggle to expand to their full potential and create new barriers to entry for new markets. To overcome these difficulties, DNMs, such as the recently reopened AlphaBay, openly discuss their intention to move their services to the Invisible Internet Project (I2P) network. These events show cybercriminals transitioning from TOR to neighbour networks, searching for better market stability and increased security against LE.

Furthermore, observing DNMs contributes to a strategically advantageous position for LEA, businesses, and researchers equally, for example, by efficiently identifying leaked and listed databases in DNMs. It is almost certain that analysing DNMs from multiple networks will further contribute to mitigating the impact of DNMs **Cyber Security-Related Products (CSRP)s**. Multiple studies have investigated the Darknet economy, topology, and DNMs structure. However, little work has been done to quantify and evaluate the impact of CSRPs, and even fewer that include multiple networks.

Moreover, most of these sources either include a description of a specialised crawler for a certain DNM or introduce an analytical framework. Consequently, researchers must develop new methods to collect data from any DNMs, independently of the network where the DNM can be accessed. This method must allow for continuous collection of information, having the closest to complete and updated historical records, and identifying CSRPs efficiently. **Ebrahimi et al. [2]** previously mentioned this challenge, stating that the impact of a given risk can change over time, and knowledge of it can only be valuable at a certain point in time. Additionally, they acknowledge the difficulties of current methodologies in identifying CSRPs that rely on Machine learning (ML).

Regarding the crawler, in our previous work we introduced **Midnight Sea (MS)**, a generic crawler capable of capturing data from TOR DNMs. Our ambition now is to continue the work on the crawler and supplement it with new capabilities to crawl DNMs accessible through other anonymous networks. Furthermore, a research gap has been identified in the lack of an efficient methodology to identify and evaluate CSRPs. Moreover, researchers have not considered DNMs accessible from multiple networks in much detail. This issue repeats in the case of DNMs exclusively accessible through anonymous networks other than TOR. Therefore, the main research question reads as follows:

How can we evaluate DNM Cyber Security-Related Products?

To break down our problem formulation and set the scope of our research, we propose the following sub-questions:

- *Why there are DNMs accessible from multiple networks?*
- *How can we use MS to capture data from these networks?*
- *How can we use machine learning to identify DNM Cyber Security-Related Products?*
- *How can we measure the impact of these listings?*

This study attempts to answer these questions. The main objective of this thesis is to contribute with a method to evaluateCSRPs. The thesis studies these questions by performing a quantitative analysis of CSRPs from the two major anonymous networks, I2P and TOR. Finally, we used the crawler mentioned above to create a snapshot of eight English-based DNMs containing over two months of data. It is essential to mention that the dataset and source of the MS systems remain in the custody of the Aalborg University (AAU). A copy of the dataset and a working version of the software can be requested directly from the university.

This thesis is organised as follows. We start in **Chapter 2**, providing background knowledge and a qualitative comparative analysis of the I2P and TOR networks. In **Chapter 3** includes a description of the research methods applied during this thesis. **Chapter 4** includes technologies and methods previously utilised to analyse DNMs. Then, in **Chapter 5** we introduce the list of project requirements to benchmark our progress. In **Chapter 6**, we include an internal structure and critical characteristics of the MS system, and its proposed enhancements. In addition, the chapter includes a description of the proposed NLP models to label the dataset. Continuing in **Chapter 7**, the chapter summarises the development enhancements made for MS and the implementation of the NLP models. Then, **Chapter 8** covers in detail the experiments realised on the collected data, differentiating between vendors and products. Furthermore, **Chapter 9** evaluates requirements and procedures, proposing new challenges and future improvements. Finally, we conclude in **Chapter 10** with a brief overview of the thesis results.

Chapter 2

Background

The establishment of DNMs on multiple networks is not new; traditionally, we could find markets with mirrored addresses for Tor and I2P on several occasions. This does not necessarily mean that the content offered is different, but rather that the same service is provided through other channels. Some of the currently available DNMs, such as AlphaBay and Kingdom, or other services, such as Kilos, offer this option.

However, diversification might come with a price. First, the adoption of new technologies faces barriers to entry for both the market and its users. For example, understanding how anonymity is achieved and its limitations is a complex task that requires strong a foundation of cyber security knowledge. Contrary to common belief, anonymity is not an absolute property and must be treated in context; previous research on this topic suggests that it is impossible to achieve total anonymity [3].

Furthermore, increasing the complexity of a system will increase the attack surface. Darknet users are more likely to encounter issues with smaller networks. Nonetheless, widely used networks are not exempt from flaws. For example, TOR has not been able to solve issues with Denial of Service (DoS) attacks [4].

The following sections describe the properties of networks I2P and TOR, giving a context for the use of the networks. The chapter concludes in **Section 2.3** with a summary of the differences between the networks.

2.1 TOR

TOR is the largest anonymous network with more than two million concurrent users. And whether knowing this number implies a threat to its users is a question about the anonymity TOR is designed to offer. The scope of the project can be found in the *Threat Model* for TOR, originally introduced in 2004 by **Dingledine et al. [4]**, in which the security goals and anonymity limitations of its design are stated. For example, TOR does not consider a threat to be distinguishable while used. In their work, TOR is introduced as a low-latency, **circuit-based** anonymity system with a trade-off between anonymity, usability and efficiency. The network has three main components: Client, Relay, and Directory Authority [5]. The roles of the components are described as follows.

- **Local Client:** A **node** that acts on the user's behalf, routing connections over **relays**.
- **Relay:** A publicly-listed node that forwards traffic on behalf of clients and registers itself in the **directory authorities**.
- **Directory Authority:** A relay that maintains a list of available relays in consensus with other directory authorities.

In TOR, users construct **circuits** to communicate with other services. Circuits are paths through the network consisting of three randomly selected relays, a *guard*, a *middle relay*, and an *exit*. The number is prefixed, and any alteration might have undesired consequences, e.g. an attacker could determine your identity by filtering circuits of the same length. These circuits will age as they are used, and once they expire, the client will build new ones. The process of building a circuit path between the client and the relays can be overly simplified in two consecutive steps:

1 **Create.** The client randomly chooses a relay from a directory authority, the *guard*. The client will then send a *create* message to communicate securely with the relay.

2 **Extend.** To expand the circuit, the client will send a *relay extend* message to the *guard* with the address of the next relay. The *guard* will create a secure connection and associate the new relay.

Any further expansion will repeat the second step, where the client will tell the current last associated relay in the circuit to include a new one. In this way, the client will always know which are the relays on his circuit, but the relays on the circuit will never know any other than the previous relay and the next.

To start a TCP connection, the client will encrypt the destination address once for each hoop in the circuit, starting from the *exit* relay to the *guard* relay. Then, it will send the encrypted message through the circuit, *peeling* a layer of encryption on each relay. Finally, the *exit* relay will behave as a proxy, connecting to the address, which is decrypted and can be read in clear text.

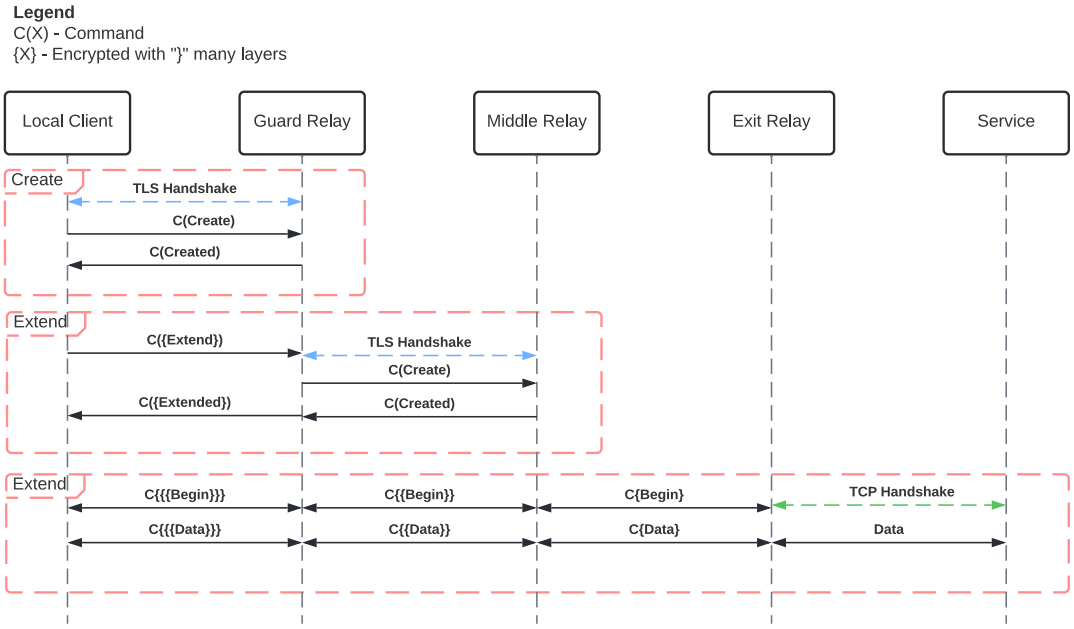

Figure 2.1: TOR circuit main activities: Create, Extend and Transmit

We summarised the three main activities that occur within a circuit in **Figure 2.1**, separating activities into blocks; however, the illustration does not represent the typical lifecycle of a circuit. Each block contains a simplified description of the activity process, indicating when the messages transmitted, named *cells*, contain commands and the encryption layers at each step.

However, this process only protects the anonymity of one side of the chain. To overcome this issue, communications between clients and inside services typically involve two circuits meeting at the so called *Rendezvous Point*. We call the services that use this method *hidden services*, and their *exit* relays *introduction points*, which the service will choose to advertise as it pleases. The process of communicating with a hidden service occurs as follows.

1 The client chooses a *rendezvous point* and builds a circuit with it as the *exit* relay.

2 The client will communicate the identity of the *rendezvous point* to the service through one of the service's *introduction points*, utilising the client's usual circuit.

3 The service will create a circuit to the *rendezvous point* and start the communication.

Figure 2.2 illustrates the connections between a client and a hidden service. The hidden service introduction point in yellow requests to create a circuit to the rendezvous point. The hidden service and his circuits to the introduction and rendezvous points are in violet. The **blue relay** represents the *rendezvous point* chosen by the client and the circuit to it, connecting to the hidden service circuit created to communicate with the client. The **red line** represents the newly created connection.

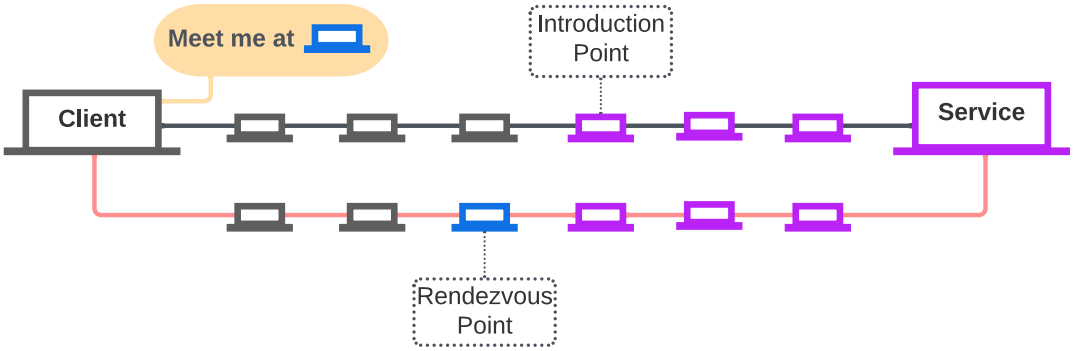

Figure 2.2: Connecting to a TOR hidden service

This design prevents observers from learning the services a client visits. However, an observer can still learn information depending on his position in the network. We projected the information observable at each stage of the communication into **Figure 2.3**. The figure shows that, unless an observer can acquire information at the **service** and some point before or at the ***guard*** relay (i.e., global threat), it is not possible to identify the client. Another example is when an observer controls the guard relay and the exit relay, the observer will know that the client is visiting a service using TOR. It is important to remember that an observer will always be able to see that a relay is using TOR.

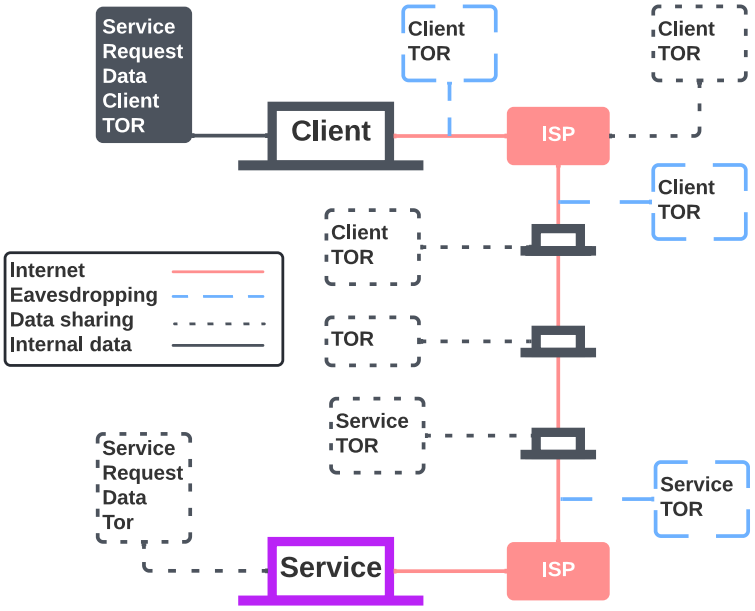

Figure 2.3: Map of the information observable at each stage of a TOR communication, inspired by [6]

2.2 I2P

In general, I2P is similar to TOR regarding privacy and security goals. Both TOR and I2P are capable of forwarding encrypted messages through multiple **nodes** to hide the identity of sender and receiver, with their major difference being the communication scheme. Unlike TOR, I2P is a **message-based**, fully distributed Peer To Peer (P2P) network intended for internal communications between peers [7, 8]. In this context, message-based means that every message is treated equally throughout the network. Clients just send data to the I2P router, and the router takes care of the data to reach its destination. This is different than TOR, where data needs to be bundled using the SOCKS5 protocol. This architecture is more flexible than TOR, requiring fewer roles for its components. Furthermore, I2P terminology changes introducing three new components: Router, Tunnel, and NetDb.

- **Router:** We refer to users as routers, since every user has an I2P router installed.
- **Tunnel:** Unidirectional virtual channels to relay messages through a series of routers. Each router has inbound tunnels to receive messages, and inbound tunnels to send messages.
- **NetDb:** A database containing information for each peer in the I2P network. The database is managed and maintained by a network of super-peers called

floodfill peers, each responsible for information about a portion of the I2P network.

When the user connects to I2P, it will create inbound and outbound tunnels of his desired length. The length of the tunnel typically involves another 2-3 routers, although this number can be extended further or reduced to none; however, it is essential to note that the length of the tunnel impacts the performance and the anonymity they might offer.

To create a tunnel, the user will send layered encrypted requests to the selected routers, so the peers can only access information relevant to them. Unlike in TOR, the routers participating in the tunnel do not have access to the reply or even know the number of participating routers in the tunnel. Furthermore, tunnels are short-lived and last between ten and twenty minutes [9]; then, the user replaces the tunnel with a new one through different routers. Tunnels themselves vary in terms of their use:

- **Exploratory:** Utilised to discover *underutilised high capacity* routers.
- **Client:** Tunnels utilised to relay messages. This tunnel is built from *fast and high capacity* discovered routers.
- **Ordering:** Used to keep an order within the pool of participants in the tunnel. Routers are selected in strict order.

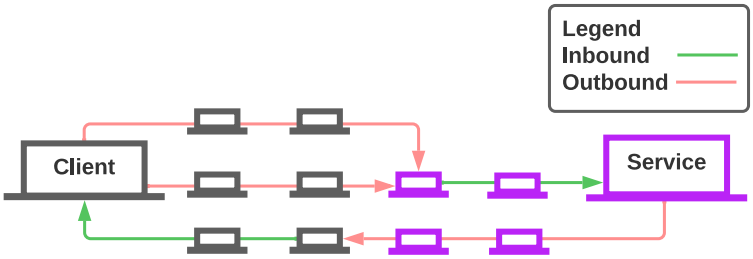

Figure 2.4: I2P outbound and inbound tunnels information flow

Before communicating with another peer, users must query the NetDb to get information about the router. NetDb contains two types of records, *RouteInfo*, which includes information on the destination router; and *LeaseSet*, documenting the entry points for some router destinations. It is the latter, the *LeaseSet*, the one that opens the possibility for *hidden services*. When an encrypted *LeaseSet* is published on NetDb, no one except the holder of the keys is capable of decrypting the content of the destination. Then, encrypting a *LeaseSet* will make it impossible for an unauthorised user to connect to the destination, and will prevent potential visitors from discovering the service in the first place.

Messages are transmitted through client tunnels, encrypted with as many layers as participants in the tunnel. The process is relatively similar to the TOR circuits, with the distinction that many tunnels can participate in sending a fragmented message. However, clients can not use the same tunnel to transmit messages in both directions. **Figure 2.4** shows this relationship, where multiple outbound tunnels send a message, reconstructed in the receiver inbound tunnel *gateway*. The receiver sends a response through his outbound tunnel to the client's inbound *gateway*.

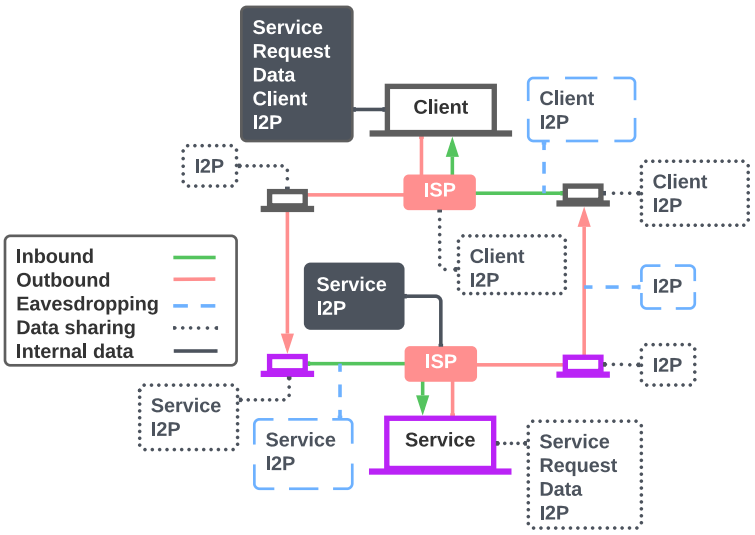

Figure 2.5: Map of the information observable at each stage of an I2P communication

Privacy-wise, I2P offers the possibility of hiding the location of its users by customising tunnels, routing policies, and encrypting *LeaseSets*. Moreover, I2P encrypts every message by default, giving end-to-end protection against non-global adversaries. However, we recognise that regular consumers may not necessarily have the knowledge required to protect their privacy within the network. **Figure 2.5** displays the information visible at each point of a default communication. The flow has been reduced to 1-hoop tunnels for simplification.

2.3 Comparison

Throughout this chapter, we have learnt that both networks include basic features to chain proxy servers. Each network implements its layered encryption protocol differently. TOR uses the **onion routing** mechanism, using a 3-hoops **circuit** to send messages. On the other hand, I2P uses **garlic routing**, manipulating and arranging messages into **cloves** that include delays in addition to message padding.

		TOR	I2P
Structure	Architecture	Client-Server	P2P
	Scheme	Circuit-based	Message-based
	API	SOCKS	I2P
	Encryption	Onion	Garlic
	Messages	512 bytes	* 512 bytes plus padding Messages are bundled Random Delays
	Routing	* Guard Node Middle Node Exit Node	User-defined nodes with the same role
Privacy & Security	Node Selection	Semi-random	Performance-based
	Security	Layered	Layered
	Multihoming	Requires additional software	Yes
	Censorship Resistance	Bridges	* Reseeding through TOR Offline seeding
	Outproxy Security	Dependant of the protocol	Not intended to navigate the Internet
	Hidden Services	Rendezvous Points	Encrypted LeaseSet
Usability	Usage	Dedicated browser	Requires third party browser configuration

Table 2.1: Summary of TOR and TOR properties

However, both networks support hidden services. TOR includes this feature through **introduction points** and **rendezvous points**. I2P uses encrypted **LeaseSets** with a symmetric key or Diffie-Hellman, depending on the use case.

However, these networks solve different problems. TOR goal is to provide anonymity while navigating the Internet. Instead, I2P is intended to navigate services within the network safely. Due to this distinction, both networks' privacy and security requirements are also different. **Table 2.1** summarises the differences between the networks, divided by the category of the property: structure, privacy and security, and usability differences.

As mentioned previously, these networks are not transparent, i.e., an observer will be able to identify nodes using the networks. For example, a country like China with an aggressive censorship policy will be able to take action against users contacting these networks. TOR attempts to solve this issue by introducing **bridges**, obfuscated non-listed non-public relays. In contrast, the visibility of I2P is only higher when connecting to the network. This process is called **reseeding**, to get an initial list of **routers** to connect to. However, users can hide this process by providing a seed file or re-seeding through TOR.

I2P major drawback resides in the software accessibility and the scalability issues presented by the lookups to netDb. The user is responsible for addressing his threat model through manual configuration. In this regard, I2P lacks the level of documentation necessary to make informed decisions while configuring the software. This issue can lead to unwanted leaks and an overall challenging experience. However, the network architecture helps to mitigate accessibility issues. Therefore, I2P is currently better suited to be used as an alternative for services with accessibility issues, providing alternative access points for their users.

Chapter 3

Methodology

To maximise the quality of our results, we choose a project management methodology adapted to our possibilities. Although the capacity of our team is limited, we considered using a flexible project management methodology such as Agile [10]. This methodology is usually found in projects that expect changes in their list of requirements, having multiple iterations throughout the project to tailor the final experience; however, it is not best suited for smaller and time-gated projects, where multiple iterations over the simpler processes are time-consuming and do not add substantial value. Other variations of this methodology include Kanban and Scrum. Kanban divides processes into smaller tasks, categorised in columns that represent the state of the tasks. Scrum works in sprints, a short period of time in which the team works to complete an agreed amount of work. In Scrum, the tasks are created by stakeholders after the approval of the *Scrum Master*, who is also responsible for creating and organising the sprints.

For this project, we chose to use a combination of methodologies between Scrum and Critical Path [11], a methodology that focuses on prioritising and bundling requirements to create milestones. This combination works well for self-managed projects with weekly supervision, shared agendas and personal diaries. Diaries are helpful for personal knowledge management and used as regular transcriptions, similarly to Scrum [12] daily meetups. The combination does not solve the impracticality's of Agile but instead helps to prioritise tasks that give the best results. For example, a typical workflow would find research gaps, implement the changes, analyse the results, and repeat. Similarly to Scrum, this methodology helps us focus on achieving short-term milestones and adapt to further changes.

We use **Obsidian** [13] as the leading software to manage the project. Obsidian is a note-taking software used for knowledge management based on local Markdown files. Therefore, the project's documentation is also written in the Markdown formatting language. In addition, we used plugins to create Gantt charts [14] and a backlog attached to a Kanban board to organise tasks and sprints.

Furthermore, Obsidian was used extensively to keep a knowledge base of notes taken when reviewing the literature. This setup helps organise the project into daily and weekly tasks adjusted to the project expectations. Most tasks overlapped with one another; for example, collecting data was often coupled with other tasks, such as documenting the project or preparing further experiments.

Moreover, GitHub [15] was used as a version control tool for the project codebase. The codebase has been made private under the AAU account, although a stable version may be available upon request. Lastly, Docker [16] has been extensively used throughout the project as a virtualisation tool for development and production purposes.

Then, since the objective of this thesis is to provide a **quantitative analysis** of listings related to cyber security, we divided the project into three main phases and their respective methods: research, implementation, and analysis. The remainder of this chapter is structured to explain how we plan for each of these phases. The first, covered in **Section 3.1**, includes the methods used for investigating how other authors approach similar challenges, identify research gaps, and prioritise their implementation. Then, in **Section 3.2** summarises the methodology used to collect data, including the implementation of the system and research ethics to collect data from DNMs. Finally, **Section 3.3** includes the methods for analysing the collected data.

3.1 Research

The project further divides this process into the following two steps regarding research. The first to collect related work and previous relevant research that attempted to answer our problem formulation to some extent. Then, to conduct the requirements gathering and prioritisation based on the findings from the previous step. The remainder of this section includes a description of the research methods used in these steps.

3.1.1 Literature Review

The primary method to gather information was to read academic publications available through IEEE Xplore [17], ScienceDirect [18], and ACM [19]. These sources were searched to find studies on the identification of DNM CSRP. The search also included methods to analyse listings using text-based entries.

3.1.2 Requirements gathering and Prioritisation

The list of requirements was built to satisfy this project's problem formulation and address significant gaps found in the related work. These requirements were built using Scrum *stories*, short descriptions of the problems that we wanted to solve and consider achievable within our scope. Scrum stories are small pieces of context that define some action or result. In addition, stories need to meet some criteria, such as being verifiable and unique. Most of these stories were created during the literature review in the form of notes containing reactions and thoughts. Once one or more stories are approved, these are formalised as a requirement and further categorised as functional or non-functional.

To decide which requirements to add to our roadmap, we use the MoSCoW prioritisation method, dividing the requirements into *Must* have, *Should* have, *Could* have, and *Won't* have. Although this method does not provide a clear path to classify the requirement nor imposes a responsibility for the lower tiers, this method helps us to realistically organise the project requirements and define which ones provide the most value.

3.2 Collecting Data

On a review of the different approaches for collecting data from the Internet, **Bar-Ilan [20]** mentions the qualities and disadvantages of this type of data. The authors refer to the dynamic nature of the medium to explain how any data collected from the Internet only represents a snapshot of the Web, further expanding on the limited utility and lifespan of the stored documents. This issue has been echoed by **Ebrahimi et al. [2]** for the specific case of identifying CSRPs in DNMs. Finally, the study concluded that general purpose data collection tools and crawlers are not suitable for the specific requirements of researchers that depend on some of these properties.

Regarding DNM crawlers, there are a plethora of dedicated systems to answer the authors' demands [21, 22, 23, 24]. However, to the best of our knowledge, there is no generic solution that can be used to crawl any DNMs. We responded to this problem by developing MS, a generic crawler for DNMs. The crawler has been used to collect a snapshot from eight DNMs accessible through I2P or TOR. The snapshot covers two months of data from listings found in categories related to cyber security, such as *fraud*, *malware*, or *hacking tools*. At the beginning of this project, the crawler had been in development for three months and required further enhancements to meet our current requirements.

The remainder of this section is divided into individual explanations of our methods used to implement changes in the MS system (**Section 3.2.1**) and research ethics towards crawling and storing collected data (**Section 3.2.2**).

3.2.1 Implementation

Our implementation method is based on test-driven development [25]. This methodology focuses on efficiency and reliability by supplementing the system with test cases that explain the expected behaviour. Each test case is implemented using the minimum amount of code to pass the test. However, this methodology requires a large amount of time dedicated to developing and documenting high-quality tests. In addition, tests do not guarantee the absence of errors; instead, they are used to know the possibilities covered by the application. We use this methodology as a way to document our progress and expectations. Although, we found it the most useful when we had to remember the logic behind some processes.

3.2.2 Ethics

Multiple sources discuss the potential implications of storing and processing widely available data from the Darknet [26, 27, 28, 29]. In this section, we give an overview of the privacy properties of our system and some of the considerations to take into account when using our method.

Our work on the Darknet is limited to collecting widely available information that can be acquired by navigating through DNMs and analysing the results to find listings related to cyber security. Due to the nature of DNMs, it is not common to find Personally Identifiable Information (PII) that can deteriorate the anonymity of the DNMs users. However, we cross-referenced vendors and users who used the same or similar usernames in multiple DNMs, which might weaken the initial privacy of the users, similarly to how third-party cookies are used for profile reconstruction. With that in mind, we still consider this risk to be reasonably low. Therefore, we decided to refer to these DNMs and their vendors by their names throughout this thesis.

The system collects responses from two main page structures within a DNM: *product* pages and *vendor* profiles. Unlike other solutions, our system is not prepared to request external links from the responses (e.g. images links), indirectly reducing the possibility of storing sensible content. In addition, we collect and store telemetry data, such as response times and status codes, to measure the efficiency of our data collection system. Because our system is meant to be as efficient as possible, we do not consider fairness over the network; however, we do not know whether this decision could impact the availability of crawled services.

Furthermore, the data has been processed to fit our database schema; this is further discussed in **Chapter 7**, along with the data points stored in the system and a description of the data processing. We must mention that all data have been stored on a local database on the same system that crawls and scrapes the data. The data have been handed to the university and will be deleted from our system after a period no longer than three months after handing in on June 2, 2022, after which the data will only be accessible upon request to the university under their terms.

3.3 Analysis

In the case of the analysis methods, this thesis conducts a descriptive quantitative analysis using records of DNMs listings. The analysis uses probabilistic methods to describe the population and differences between DNMs. In addition, the dataset is separated into category clusters, that is, groups of listings with similar characteristics that respond to specific topics. For example, the dataset contains malware records and tutorials; therefore, these listings need to be analysed separately to understand the differences between clusters and study each cluster's characteristics. Moreover, the analysis includes statistics to examine the description of the products. This is important to understand the supply and how vendors use DNMs to advertise their products. These statistics include wordclouds, studying the word frequency of the listings, and sample entropy analysis to estimate the similarity correlation between listings, that is, how similar are the product descriptions in a sample.

Chapter 4

Related Work

In this chapter, we present previous research related to our problem formulation to identify the issues and challenges of identifying cyber security related products advertised in DNMs. First, **Section 4.1** enumerates other sources that try to crawl DNMs accessible through I2P. Then, **Section 4.2** dives into previous work on analysing the content of DNM products. In addition, the chapter includes descriptions of the state-of-the-art technologies and methods developed in this area, arguing for possible weaknesses and further improvements. Finally, **Section 4.3** combines the result of our findings, presenting how to improve the current state. This information will help us to build our requirements in **Chapter 5**.

4.1 Crawling I2P

Although there has been a significant amount of work measuring the I2P network [30, 31, 32], there is a lack of resources that explain the implementation of crawlers for this network. The studies included in this section briefly mention their approach to implement an automated method to crawl I2P sites.

Iliou et al. [33] implemented a crawler capable of navigating multiple networks seamlessly, including I2P. Their system uses four proxy modules that route connections through the right network. When the crawler finds a new hyperlink on a site, the crawler will request the link and use the TLD (Top Level Domain) part of the hyperlink URL to determine which network to use. For example, ".onion" will be routed through TOR, and ".i2p" through I2P. However, the purpose of this crawler is to create a map of the networks rather than collecting information from the sites.

Moreover, **Magán-Carrión et al. [34]** introduces a crawler named **c4i2p** for I2P sites. The crawler has been developed to analyse the structure of the I2P network and discover the characteristics of *eepsites*.

This crawler shares similarities with MS, such as programming language and multiple modules used for similar purposes. Although the crawler serves a different purpose, it is interesting to look at the routing implementation for I2P sites. Their implementation follows a simple pattern of cleaning the URL by adding the URL of the proxy service to route the requests.

4.2 DNM Analysis

With regard to analysing DNM, a considerable amount of literature has been published on studying the behaviour of DNM vendors and the characteristics of their products [23, 26, 35, 36]. Many contributions limit their study to qualitative methods to describe their findings, while other sources may focus on forums and other channels rather than on DNMs [37]. Other interesting sources reveal vendors' behaviour across DNMs, attempting to identify their trajectory over time [38]. However, the amount of work on studying CSRPs alone is somewhat limited.

Most of the contributions that study CSRPs focus on building machine learning models capable of classifying the listings simply by looking at the title and description of these listings [39, 40, 41]. Other sources introduce new tools to categorise the listings [42]. This section summarises some of the most relevant sources that attempt to give an understanding of the products advertised on the Darknet.

In 2015 **Macdonald et al. [43]** published an analysis of cyber threads from common forums for hackers, focussing on threats to critical infrastructure. Their method is based on *sentiment analysis* in the body text of each thread. The procedure would assign a "bad sentiment score" based on the repeated appearance of nouns in a dictionary of jargon hackers use. Furthermore, more research has been conducted using this method for the same purpose [40], improving this methodology to consider more than 100 forums and building a historical reference of events based on sentiment. The authors used three different models to calculate the average sentiment for each post on the forum. We believe their work could benefit from linking posts to threads and calculating thread-wide sentiments. Although the system is relatively simple and effective, some of the limitations of using this approach are related to using a fixed dictionary, a list of permanent words that might change over time. Moreover, the context needs to be evaluated separately for each threat.

Shortly after that, **Nunes et al. [44]** modernises previous work by introducing a semi-supervised SVM ML approach to identify cyber threats from DNMs and forums. To train their model, they used a manually labelled dataset, using binary labels of whether or not the entry is a cyber threat. This simplistic approach introduces what became the trend for cyber threat analysis.

More recently, **Ebrahimi et al. [45]** introduced a method to detect cyber threats in non-English DNMs. Their approach utilises a deep cross-lingual model that does not rely on translating tables. They trained a model using BiLSTM, a bidirectional machine learning neural network model used for natural language processing in problems requiring contextual meaning. In this case, the model is used to learn from multiple languages simultaneously, transferring the knowledge learnt from English DNMs to Russian-based DNMs. However, this approach suffers from drawbacks of using supervised ML models, the need for adequately labelled data to train the model, and in this case, for both languages. Finally, the proposed framework required further work for short entries at the time of publication. The authors published a similar work later [46], using a customised version of BiLSTM for the cyber threat classification, a semi-supervised TSVM model to label the necessary data, which required a small 3% of pre-processed data for training. The result from the TSVM labelling model is fed into the BiLSTM to enhance the performance of the overall method.

4.3 Summary

Year	Study	Labelling	Classification	Result	Supervision	Forums	DNMs
2015	[43]	NLP	Sentiment analysis	Threat score	Semi-supervised	✓	
2016	[44]	Manual	CT-SVM	Threat score	Supervised	✓	✓
2018	[47]	SGNS	SVM	Predicts vulnerabilities	Semi-supervised	✓	
2018	[40]	NLP	Sentiment analysis	Threat score	Unsupervised	✓	
2018	[45]	CLKT	BiLSTM	Threat score	Semi-supervised	✓	✓
2020	[46]	TSVM	BiLSTM	Threat score	Unsupervised	✓	✓

Table 4.1: Overview of recent work identifying cyber threats in DNMs

Table 4.1 shows a summary of the most relevant literature. Previous research tend to focus on forums rather than DNMs. These studies use ML to identify and label their dataset using different versions of sentiment analysis. Most of the work gives a *threat score*, that is, the coefficient of whether a listing is a cyber threat. Moreover, there hasn't been an effort to classify and measure the threat level to the best of our knowledge. This implies a limitation for their systems, as it still requires further labelling to understand the content of the dataset.

Chapter 5

Requirements

This chapter introduces the system requirements gathered from the related work and other issues found in the previous version of the MS system. Furthermore, the chapter includes a summary of the *scrum user stories* that motivated the requirements and the reasoning for their prioritisation.

The remainder of the chapter is structured as follows. **Section 5.1** summarises the system requirements collected throughout the project. Then we use the MoSCoW method to prioritise these requirements and further divide them according to project expectations. Finally, **Section 5.3** briefly reviews the requirements introduced in this chapter.

5.1 System requirements

Turning now to the system requirements, this section organised the requirements into **Functional Requirement (FR)** and **Non-Functional Requirement (NFR)**. **Glinz [48]** studies the definition of those two terms and concludes that *functional requirements* are those that define some functionality of the system. On the other hand, *non-functional requirements* are those that define attributes, performance, and other qualities of the system. Some requirements are simple and straight forwards; however, other requirements are better explained by their context. Therefore, this section includes a description of the requirement to clarify its motivation and classification. The requirements are stated as follows.

Functional Requirements

FR1 Crawl DNMs accessible through TOR or I2P

As mentioned in the project goals, MS must be able to collect data from DNMs on different networks.

FR2 Identify CSRP using NLP

The system must be able to reliably differentiate CSRPs. As we have learnt from related work (**Section 4.2**), NLP is currently the best method to label text-based entries that require contextual meaning.

FR3 Classify CSRP using NLP

To analyse the products, the system must be able to classify products by their type of content, distinguising cyber security apart. For example, to differentiate malware from hacking tutorials.

FR4 Resume-crawling feature

MS lacks the ability to resume crawling from previous sessions. This function will allow researchers to perform separated crawling sessions over time. The system must adapt to changes such as new listings being introduced at the beggining of the DNM category section, which will shift the position of products not yet seen by MS.

FR5 Real-time labelling and classification using dynamic learning

Due to changes in the vendors' lexicon, machine learning models will require to be retrained in the future. It would be beneficial for the system to include methods that identify lexicon changes over time. The best known method to future proofing a machine learning model is to dynamically train it using a real-time stream of data.

FR6 Use multiple client nodes for crawling routing

To further improve the performance of the crawler, we could include Multi-homing techniques, overcoming bottlenecks that may happen while crawling at a higher rate.

Non-Functional Requirements

NFR1 Complete crawls from DNM including CSRPs

During the analysis phase of this thesis, we will use only complete snapshots of DNMs containing CSRPs. These snapshots include records of products found in relevant and neutral categories. For example, categories primarly used for advertising drugs will not be crawled; however, miscellanea and fraud will be crawled.

NFR2 Utilise only English based DNMs

This thesis includes exclusively English-based DNMs. However, the dataset may include products which description is written in another language.

NFR3 Risk analysis on the listings

The content of each category may include a large variety of products. Therefore, this study could include a brienf risk analysis of the products.

NFR4 Improve the usability of MS

The MS complexity can be reduced significantly. Reducing the number of services involved in the system would simplify many tasks, such as maintaining the project and using the system.

NFR5 Reduce the verbosity of MS

While crawling, MS has proven to be too conservative, stopping on each failed request. This creates overhead and fatigue on the researcher. We propose reducing the level of verbosity or improving the validation to require less attention from the researcher.

5.2 Requirement prioritisation with MoSCoW

Regarding requirements prioritisation, this section reintroduces the list of requirements distributed into the four MoSCoW categories: *Must*, *Should*, *Could* and *Won't*. *Must* contains these requirements that are critical for our project. Without these, we would not be able to generate results to answer our research question. In *Should*, includes requirements that would add a relative value to the project without depending on them. *Could* requirements are good additions to be considered in this project or in future improvements. Finally, *Won't* includes requirements outside of the current scope, although these could be implemented in the future.

Table 5.1 summarises the system requirements distributed using the MoSCoW method. Most of the requirements fall into the category *Must*, including **FR1, FR2, FR3, NFR1** and **NFR2**. This thesis will focus on those requirements that are vital for the project. The category *Should* contains the requirements **NFR3, NFR4**. These requirements have a substantial impact on the result on the project. Although, these requirements do not represent the main objective of the project. The *Could* category contains the requirements **FR4, NFR5**, which do not have direct impact on this thesis. Category *Won't* includes the requirements **FR5, FR6**. These requirements would have a sound impact on the MS system; however, these requirements are unreachable within this project.

Category	ID	Requirement
Must	FR1	Crawl DNMs accessible through TOR or I2P
	FR2	Identify CSRP using NLP
	FR3	Classify CSRP using NLP
	NFR1	Complete crawls from DNM including CSRPs
	NFR2	Utilise only English based open markets
Should	NFR3	Risk analysis on the listings
	NFR4	Improve the usability of MS
Could	FR4	Resume-crawling feature
	NFR5	Reduce the verbosity of MS
Won't	FR5	Real-time labelling and classification using dynamic learning
	FR6	Use multiple client nodes for crawling routing

Table 5.1: List of the project system requirements prioritised using the MoSCoW method.

5.3 Summary

This chapter began including the list of system requirements necessary to overcome the research gaps found in the previous chapter 4. The requirements were divided into *functional* and *non-functional*. Furthermore, the requirements were prioritised using the MoSCoW method. The MoSCoW category was assigned based on how necessary is the requirement to complete this project. In summary, eleven requirements were identified, five of which were categorised as *Must*, two as *Should*, two more as *Could*, and another two as *Won't*.

The content of the requirements is atributed to two main problems: improving MS to collect DNMs data from multiple networks, and developing NLP models to label the data. In the chapter that follows, we will present the system design that attempts to solve these issues, explaining how we planned to implement the requirements introduced in this chapter.

Chapter 6

Design

In the following pages, we will present the system design used to collect and label DNM data. As previously stated, this project uses the MS system to collect data. Therefore, this chapter begins with a brief overview of the MS system in **Section 6.1**. This section will be used to reference further changes in the original design. The chapter continues defining the system design planned for this project in **Section 6.2**. The design describes the changes to MS and includes an introduction to the models used to label the data. Finally, **Section 6.3** includes the list of heuristics used during this project.

6.1 Midnight Sea: A Generic Crawler For Darknet Markets

The MS system was designed to collect data from multiple TOR DNMs simultaneously. This system uses multiple processes to continuously crawl, scrape, and store data as efficiently as possible. Using a microservice architecture MS separates the duties of each service, guaranteeing the constant flow of information between crawling and scraping.

The original MS architecture used five services: *core, crawler, storage, scraper* and *planner*, plus a shared external volume. Initially, the *core* service was intended as a mediator between services. Each service neededed to communicate with the *core* service to route information from one service to another, simulating the structure of a centralised system. The *storage* service would be in charge of storing the crawled content. This content is preemtively stored in the shared folder without formatting, and pending to scrape. Once the data points are collected from the content, these are stored in a relational database, and the raw file is compressed and moved into a parsed folder in the shared drive.

Furthermore, the system required two types of especial files: *blueprints* and *plans*. *Plans* YAML structured files containing instructions of how to crawl a DNM. For example, *plans* contain instructions to identify CAPTCHA pages. These files are sent from the *planner* to the *crawlers* at the beggining of each crawl. Similarly to a *plan*, a *blueprint* contains instructions to identify relevant HTML elements and how to extract information out of them. This type of file is used in the *scraper* service to extract the data points from the raw content. However, these files are manually written, requiring time and resources from the researchers. To create these files, it is required basic knowledge of the HTML structure of the targeted DNM, and real samples used for validation.

Among other features, the *crawler* service included methods to maintain an open session with DNMs, and control the crawling behaviour. This last feature, called *budgets*, regulates the crawling speed and the number of simultaneous connections allowed, attempting to mimic human behaviour. The *crawler* included a smart budget that would change the budget parameters based on previous responses. Furthermore, the *crawler* features *strategies*, a module meant to plannify the crawling process. Generally, the *crawler* would stop crawling when it finds a page that does not contain non-crawled products. For example, in the scenario of a category divided into ten pages, showing ten products per page, and crawled from the second page onwards, the *crawler* will stop at the second page, knowing that anything further has already been crawled. In addition, the *crawler* will stop to ask for human input when it encounters a problematic page, such as a CAPTCHA or login pages. For further reading on the original system, refer to the ***Midnight Sea: A Generic Crawler for Darknet Markets*** project.

6.2 System Design

Turning now to the system design for this project, this section describes the changes intended for MS and a brief introduction to the models used to label the data. **Figure 6.1** shows the intended system design, where the MS system architecture has been simplified to reduce the number of services.

In this iteration, the services *core* and *planner* has will be replaced with internal processes. This change requires the system to allow communication between the rest of the services. However, the number of messages is reduced significantly, lowering possible throttling on the system. In addition, common resources are stored in a shared volume accessible from all services. The volume contains *plans*, *blueprints*, and parsed and pending files among others.

Figure 6.2 shows a diagram representation of the two main processes that occur simultaneously between the *crawler, storage* and *scraper*, using the proposed design. The process loop that happens on the top simulates the behaviour of crawling one or more pages simultaneously. First, the *crawler* will check with the *storage* whether

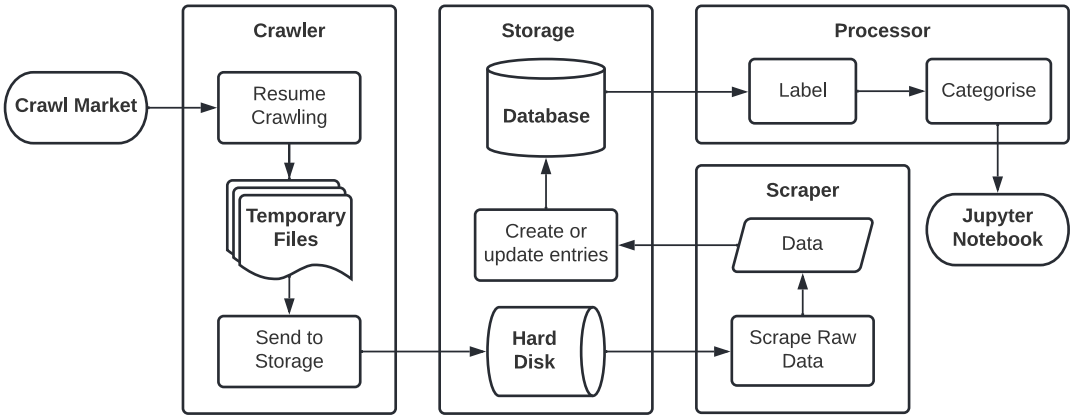

Figure 6.1: Research Design using MS system, and an external module for processing and label data.

the pages are already in the database. The *crawler* will continue to crawl the ones that are not in the *storage*, sending shortly after the raw data and extra metadata, such as the type of page and the name of the DNM. The process at the bottom is a loop that occurs on the *storage*. The *storage* will query the database for pages pending scrape and send this information together with the raw data to the *scraper*. Finally, the *scraper* responds with the data points scraped from the data. Further changes to MS will be discussed in **Chapter 7**.

In regards to the processor module shown in **Figure 6.1**, this module is not connected to the MS system. Instead, the module is designed to be a separated entity that utilises a backup of the data stored in the MS database. The module contains two machine learning models to label the data. The first model will label the records to determine whether they are CSRPs. The second model categorises the positive results using one of six common labels to differentiate CSRPs.

Table 6.1 summarises the processor models, including the labels used for each model and a description of the content found under each label. The first model (M1) performs a sentiment analysis with a threshold to determine whether the entry is relevant or not. This is a binary classification with two labels, *Positive* (B1) and *Negative* (B2). The second model (M2), uses a set of labels to categorise the listings. Since each DNM uses their own classification method, and vendors will list their products in the categories they find most convenient, we decided to use a different set of labels. On the other hand, our classification is rather simple, based on the type of item that can generally be found in some category. In early stages of the project we used a tree of labels to classify products. However, products that belong to smaller groups turn challenging to classify. Therefore, the categorisation was built over the previous work [39]. Then, the labels were generalised to cluster items with similar characteristics, resulting in the following set of labels.

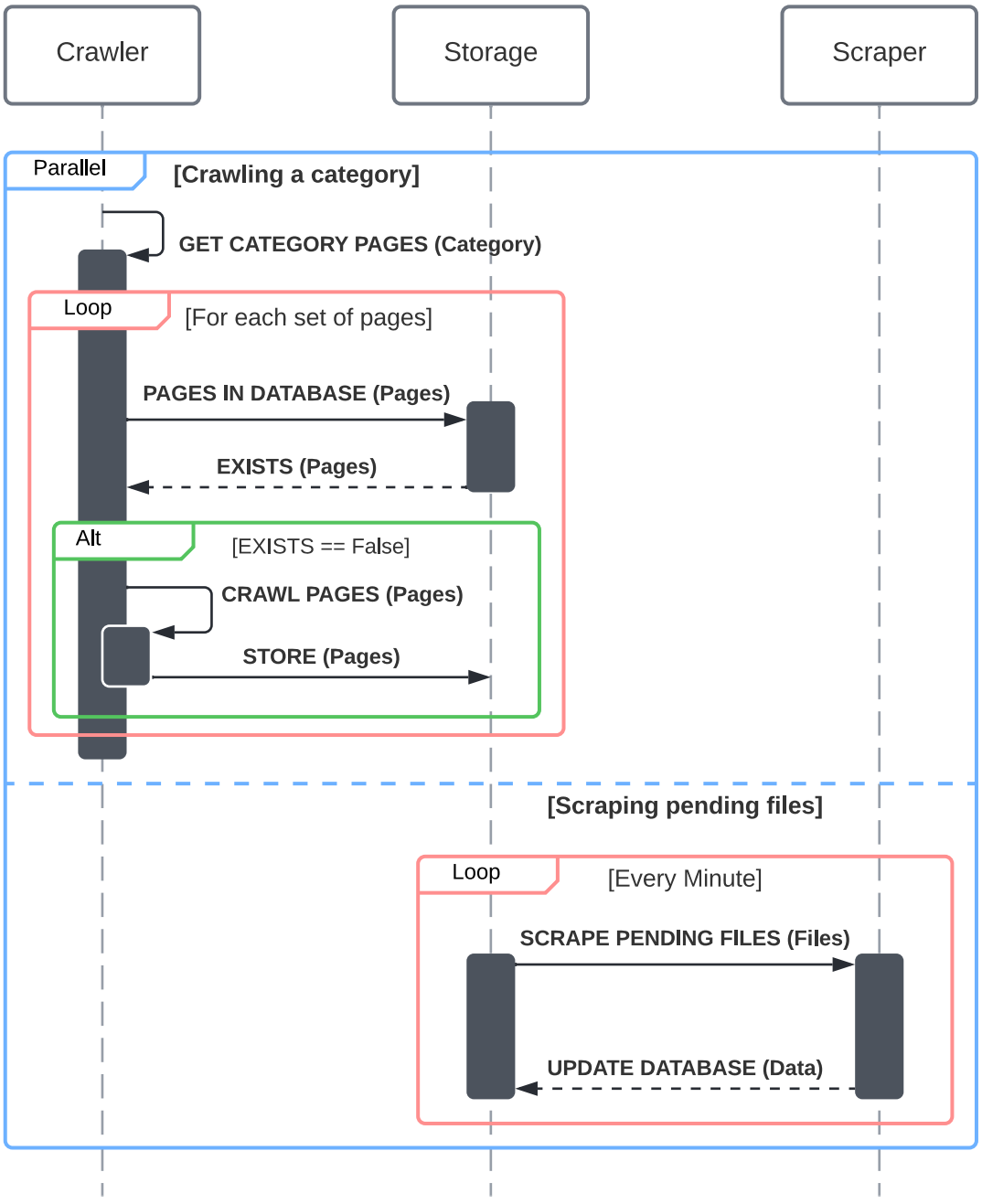

Figure 6.2: MS flowchart of two parallel processes: crawling categories and scraping pending files

Model	ID	Label	Content
M1	B1	Positive	Cyber Security related listings
	B2	Negative	Non-related listings
M2	C1	Media	Tutorials, E-Books and Videos
	C2	Malware	Keyloggers, RAT, Ransomware, Botnets and Exploits
	C3	Software	Tools and Security Applications
	C4	Fraud	PII, Credentials, Tokens, Documents and Databases
	C5	Physical	Telephones, Servers and Burners
	C6	Services	Hosting, DDoS, Spam and Carding

Table 6.1: Labels used to group CSRPs based on their type of content

C1 **Media:** Contains digital media content, such as tutorials, e-books, and videos.

C2 **Malware:** Software prepared for offensive cyber security intended to disrupt other machines. The label contains RAT kits, ransomware, exploits, keyloggers, and botnets.

C3 **Hacking Software:** Tools and other utilities for hacking, such as messaging applications or carding tools. This label also contains bundles of tools and other related software.

C4 **Fraud:** The label contains listings that advertise content with personal information. For example, credentials, accounts, bank details, databases, etc.

C5 **Physical:** Physical devices used in cyber. Burners (one-time use devices), physical servers, or telephones for satellite communications can be found under this label.

C6 **Services:** Cybercrime-as-a-Service (CaaS). For instance, DDoS services, Pay-per-install services, Spam campaigns, or Bulletproof Hosting.

While previous researchers used SVM models for NLP text classification tasks and sentiment analysis, recent developments in the areas of linguistics and machine learning have introduced major improvements that outperform this model. For example, models such as Bidirectional Encoder Representations from Transformers (BERT) [49], a transformer based ML model, that work towars solving issues in text processign by including contextualisation techniques, greately outperform SVM in almost every task according to the SuperGLUE benchmark [50]. These models transform senteces into a statistical problem in which our sentences are defined vectors on a plane. By calculating the distance between these vectors, we can estimate a likelihood parameter, an indicator of how close a vector is to the epicentre

of a group of vectors with similar characteristics. That is, to form correlations between words, giving them context, something not possible when using SVM.

Moreover, BERT based models are pre-trained using large amounts of data, which can be further trained to solve specific tasks. However, State-of-the-art BERT based models present instabilities and often introduce proportionally large computational requirements [51, 52]. In addition, a model may have been pre-trained using heuristics that will affect the precision of our results. For example, T5 uses the Colossal Clean Crawled Corpus (C4), which does not contain offensive content [53]. Other BERT based models such as RoBERTa may offer a runner-up performance but are more reliable [54, 55].

The details of the implementation of the models are introduced in the following in the next chapter (**Section 7.2.1**), including a performance comparison for each task. The summary of the classification results can be found in **Chapter 8**, as an introduction to the quantitative analysis.

6.3 Heuristics

This section introduces our heuristics, a list of assumptions and guesstimates that we used to simplify tasks or general decision making. These decisions are not empirical; however, it is important to consider these items, as they may have impacted our results.

H1 **Categories Prioritisation**

We intentionally prioritised those categories that were more likely to contain CSRPs. Furthermore, we did not collect information from other categories such as *Drugs* and the like.

H2 **Selecting Relevant Markets**

We do not crawl DNMs that do not contain CSRPs.

H3 **Crawling using the *Simple* budget**

In the past, we have had mixed experiences with other budgets. We recommend further improvements before using more complex budgets; at the time of this project, the simple budget was efficient enough. However, crawling speed could be significantly improved if we had more time to experiment with budgets.

H4 **Crawl using the I2P option when available**

As can be seen in our experiments **Appendix B** and previously in **Chapter 2**, I2P averages a better download speed than TOR. Therefore, we decided to crawl from the I2P options when possible.

H5 **Labelling heuristic**

We manually labelled a small set of entries to train our model. Each entry requires one to two labels, as previously seen in this chapter. The first is whether the entry is related to cyber security. The last is the category to which the listing belongs.

H6 **Holding Prices heuristic**

Soska et al. [56] developed a heuristic to remove bias from their analysis. In their study, they observed a phenomenon in which vendors used a technique denominated *holding prices*. This technique is used by vendors to retain a product advertisement by replacing the price with an extremely high value or offering the product for free. This may also be used to indicate the vendor is restocking.

This heuristics is based on the probabilistic distribution of their observations, discarding records with prices below the 25 percentile and greater than five times the median of the observed prices. Furthermore, they discarded products with a higher price than $10,000 USD However, this is not an accurate view for our dataset. Using their heuristic, the upper bound of our dataset is approximately 100 EUR, not enough to capture many legitimate observations that are worth studying. Therefore, our heuristic uses an upper bound of 100 times the median, yielding a maximum price of approximately 744 EUR. Similarly to [56], we discarded any observation less than 25% of the median. The lower bound is then located at 1.93 EUR.

Chapter 7

Implementation

This chapter focuses on the results of the development process. The structure of this chapter is organised as follows. **Section 7.1** summarises the changes made to the crawler MS, following the design choices described previously in **Chapter 6**. The section is further divided into subsections to introduce the changes to each internal service of the application. Next, **Section 7.2** includes the implementation details of the processor module. This section is divided into the implementation of the NLP models and the analysis setup.

7.1 Midnight Sea Improvements

Turning now to MS, it is important to remember the previous architecture to understand how our changes impact the system's logic. In the previous version of MS, the services communicated through a centralised service called *core*. In our build, the services can communicate between themselves by following each other's API endpoint message structure. These new pipelines required implementing client and server packages on each of the services and modifying the API endpoints to reflect these changes. Moreover, the application has been reduced to a single container to improve the usability of the system, directly targeting requirement **NFR4**. In consequence, using SSL certificates to create channels between services is now an optional feature, since the services are understood to be under the same *host*. As previously mentioned in **Section 6.1**, shared resources are now situated in a shared volume, which replaces the previous *planner* service, and prepares the system to use standard packages.

This new setup includes two companion applications to monitor the status and resources of the system: *PgAdmin* [57], and an automated backup application. PgAdmin is a PostgreSQL administrative platform with support to perform backups on-demand and complex queries to the database from a simple GUI interface.

Although these software are not required for the system to function, they add a significant value when using MS. Lastly, the *proxy* service has been replaced with a version supporting routing through I2P and TOR using preconfigured options.

Figure 7.1 shows the new internal structure of the system divided by networks. Each cube represents a service by name, with the distinction of the *proxy* service, coloured **pink**, which is the only service capable of communicating with the Internet; and the database, represented as a cylinder. The dotted lines represent the perimeters of the network in which each service can communicate. The **green line** represents the network dedicated to MS, while the **blue line** belongs to storage services that need to communicate directly with the database, and the **red line** contains services that need to communicate with the *proxy* service. It is important to remember that the services are only accessible within the perimeter of their networks.

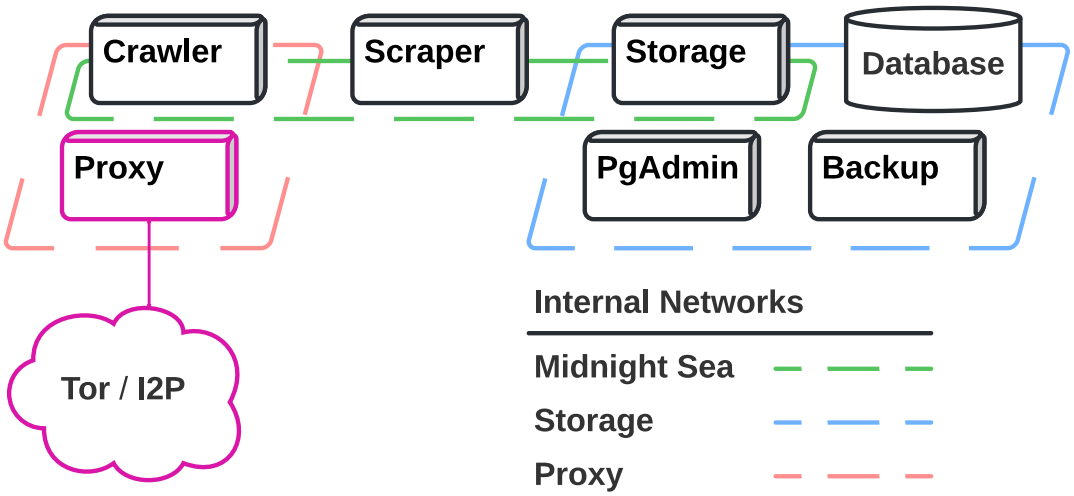

Figure 7.1: Perimeters of the internal networks on the Midnight Sea system

The following sections provide a more detailed description of the individual services. **Section 7.1.1** introduces the changes to the *crawler* service. Then, **Section 7.1.2** covers the changes made to the *storage* service. Finally, **Section 7.1.3** includes changes in the *scraper* service.

7.1.1 Crawler

Three main issues were identified in this service when implementing our requirements **FR1**, **FR4** and **FR5**:

- **The crawler could not access I2P services**.

  Previous versions of the *crawler* could only crawl TOR sites, which clashes with requirement **FR1**. In addition, the *crawler* was unable to differentiate TOR from I2P sites. Therefore, all the packets were routed through the same getaway using the SOCKS protocol.

- **High verbosity**

  Next, the level of verbosity was causing fatigue. As stated in requirement **FR4**, the service used to stop frequently to request human input. Moreover, the service logged each interaction by default. These messages are mainly for debugging purposes and add little value to the typical user.

- **Unable to resume previous crawls**

  The *crawler* is not capable of resuming previous crawls, in conflict with the requirement **FR5**. Since the goal of MS was to recreate the visible part of DNMs databases through continuous crawling, the *crawler* did not consider the time since previous crawls. This lack of time perception created an issue when resuming crawls in highly active DNMs. For example, if the difference between crawls was sufficiently large, resuming a crawl on a particular page will only result in already recorded items, which in turn will stop the crawl.

Therefore, the following internal packages received improvements to address these issues.

Session package

This package contains relevant methods for maintaining a session between DNM and *crawler*. Additionally, this package manages the crawling *budget*, the set of resources that determine the number of concurrent connections and delays between requests, to mimic human behaviour.

The package has been enhanced with a network module that manages the routing of outgoing traffic. This module includes functions to determine the port, protocol, host address and methods to build URLs to the respective proxy. In addition, two networks have been hard-coded in the application with the location of the *proxy* service to access I2P and TOR addresses. It is worth mentioning that the module will automatically route requests through the right network based on the structure of the requested URL.

However, the current implementation does not support using multiple proxies for the same type of network. These enhancements solve **FR1**, allowing the system to crawl I2P and TOR sites simultaneously without the need for additional changes.

Additionally, the *budget* module received changes to store metrics from each request response to a local file. These metrics include the delay suggested by the budget, the response time, the URL of the request, and an ID of the current budget. Metrics such as delay, budget ID, and the number of maximum connections is given in the form of a budget *recommendation*. When the crawler requests an address, it consumes one connection from the budget recommendation. Then, when the budget is depleted, a new *recommendation* is created for the next batch of requests. Each budget implementation includes these metrics with different values. However, these values can easily be changed using environment variables. For example, a typical simple budget may have a maximum of ten simultaneous connections, and a maximum delay time before each request of five seconds. Moreover, the maximum delay has been improved to be the ceiling for each request, instead of a shared fixed number. These changes allow for further customisation on each crawl, a necessary change to maximise the crawler's performance. In addition, the number of budget implementations has been reduced to one from three, since other implementations have not been tested and may yield inconsistent results.

Crawlers package

This package is responsible for cleaning requests, validating responses from the DNM and considering reattempts in case of failure. The package features new validation methods to distinguish between required and invalid content. The required content is the one that, if not found in the response, the validator will raise an error, whereas invalid content will raise an error when seen in the content. Furthermore, the validation conditions now vary between response codes and page content elements. For example, request timeouts and responses with status codes greater than 300 no longer force the crawler to ask for human input; instead, the crawler will retry to crawl these pages just once, since the *crawler* service can crawl them at a later time as long as the *storage* service contains a record of those pages. These changes drastically reduced the amount of supervision required by the *crawler* service, increasing the level of automation of the general system, and having a direct impact on the implementation of requirements **NFR4** and **FR4**.

Strategies package

This package contains the algorithms used to crawl the different sections of a DNM called *strategies*. The current implementation contains two different strategies. The first is a generic algorithm that crawls from a list URLs using using a budget to control the crawling speed. This implementation will create pools of connections given by the budget to request pages simultaneously, storing their responses in temporary files. When the budget is depleted, the content of the temporary files is bundled and sent to the *storage* service. This allows the *crawler* service to crawl pending pages. The second strategy is a dedicated algorithm to crawl DNM *category* pages.

Before continuing, it is worth explaining how the DNM category pages work. DNM category pages use pagination methods to divide the content presented to the user in multiple consecutive *pages*, instead of a single large page with all the category products. To avoid confusion, we refer to these pages as Indexed Pages (IDP). It is important to note that the content of the IDPs is not static, that is, the content of a IDP will change over time as new entries are added to the category. The canonical method to paginate products is to order them by date in descending order, unless other filters are applied. Therefore, newer products will appear in the first IDP, while the rest of the products will be shifted according to the maximum number of products per IDP.

In this iteration, the strategy used to crawl products across category IDPs has been enhanced with a new feature called *elastic windows* to fulfil requirement **FR5**. This feature gives the crawler a margin of IDPs to consult before stopping searching for products not yet crawled, further referred to as Non-Crawled Products (NCP). The elasticity property is given by the capacity to expand it's size when a NCP is found in a IDP. Doing so, the window will grow linearly as long as the IDPs contain NCPs. The window growth factor is equal to the original size of the window. In addition, when a NCP if found within the window, the crawler will create another window after this one. Consequently, the crawler will only stop when no NCP is found within the entire window.

Figure 7.2 shows this feature in a practical example using a window of five IDP, where there are only NCP on IDP three and IDP eleven. Because of this, the first window grows to be of size eight, and since the first window contains NCPs, the crawler creates a new window starting on IDP nine. Similarly to the first window, NCPs can be found on IDP eleven, making the second window grows from IDP eleven to sixteen, since the initial window size is five.

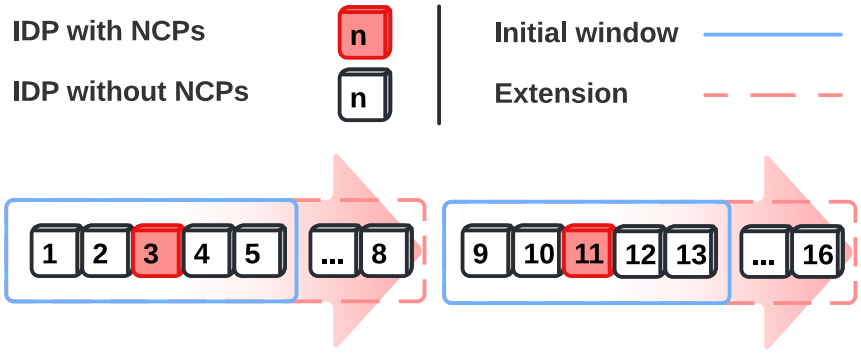

Figure 7.2: Example of an elastic window of size five used to capture NCPs from IDPs

Moreover, the initial size of the window is determined by the difference between the previous crawl and the current date, with an upper bound of size five and a lower bound of two. These numbers are chosen purely on expectations and directly impact the crawler's performance. Reducing the bound range will restrict the margin for finding new listings on further IDP. However, DNM crawled on daily basis could use a lower range, e.g., between one and two.

Returning to the *plan* files, previously introduced in **Section 6.1**, now features grouping of multiple categories under a single name, as an enhancement to the usability of the system. This solves an issue in where DNMs would use subcategories, adding overhead when creating new *plan* files. Categories can now be defined using URLs with placeholders for variables. These placeholders will be replaced with the values of a list or a range of numbers. **Code block 7.1** illustrates this change, using two grouped category pages. Each page path is replaced with the variable values, resulting in a list of pages with similar characteristics.

```
pages:
  - name: Fraud # Name of the grouped category
    path: listing_category?id=${range} # variable 'range'
    vars: # variables to use as replacements
      range: [2, 9] # values to replace in the path
  - name: Security and Hosting
    path: listing_category?id=${list}
    vars:
      list: [125, 129, 132]
```

Listing 7.1: Example of plan category pages grouped using variables on the path string

7.1.2 Storage

Since the *core* service is no longer included, the *storage* service requires several changes to communicate with the rest of the services. For example, the service now includes methods to send pending files to the scraper, in order to receive the data points. This has been achieved by including separated processes called *commands* that can be used while the service is active. These *commands* have been developed to cover a number of maintenance tasks. For example, due to changes in multiple DNMs and the development of the MS system, we required a method to re-scrape the content of the parsed files. *Commands* can either be triggered manually or used as parallel processes to the service in the form of a constant loop, such as sending pending files to the scraper, or cleaning the database of invalid entries.

Furthermore, the service has been enhanced with an internal API system to perform generic operations on the database. This system separates the implementations for each model while maintaining a standard structure that other services can use through a single interface. The change allows complex operations such as updating entries with deep relationships using JSON formatted messages. The result is seamless communication with the services *scraper* and *crawlers*, where these services no longer need to know the structure of the database.

In addition, the database schema now includes unique restrictions for multiple pair of fields such as page URL and DNM name. Furthermore, some tables have been removed favouring simple fields. For example, the table *category* is now a field of the listings, referring to the category in which the listing was originally found. Moreover, the schema contains events that trigger when an instance with relationships is deleted from the database to produce cascading effects when needed.

Figure 7.3 illustrates the renewed schema in the form of a diagram, showing the relationships between the internal tables. This schema is meant to be used with placeholders or incomplete page records. Placeholders are usually created when *crawlers* need to check whether a sequence of listings is already in the database. Those listings that are not found in the database are stored as placeholders, avoiding the situation in which crawlers fail to crawl some listing without creating a record.

7.1.3 Scraper

The *scraper* service now is able to capture fields in the form of lists and groups. This has been achieved by introducing structural changes to the *blueprints*. In a *blueprints*, each data point is now treated as a group that may contain child fields. This is a recursive property that can reach an unlimited depth, allowing the scraper to populate relationships between tables in the database. Moreover, each group may contain the property “many”, indicating that the data point is expected to be a list of values. In addition, each data point may include a “clean” property, using a regular expression to target relevant information.

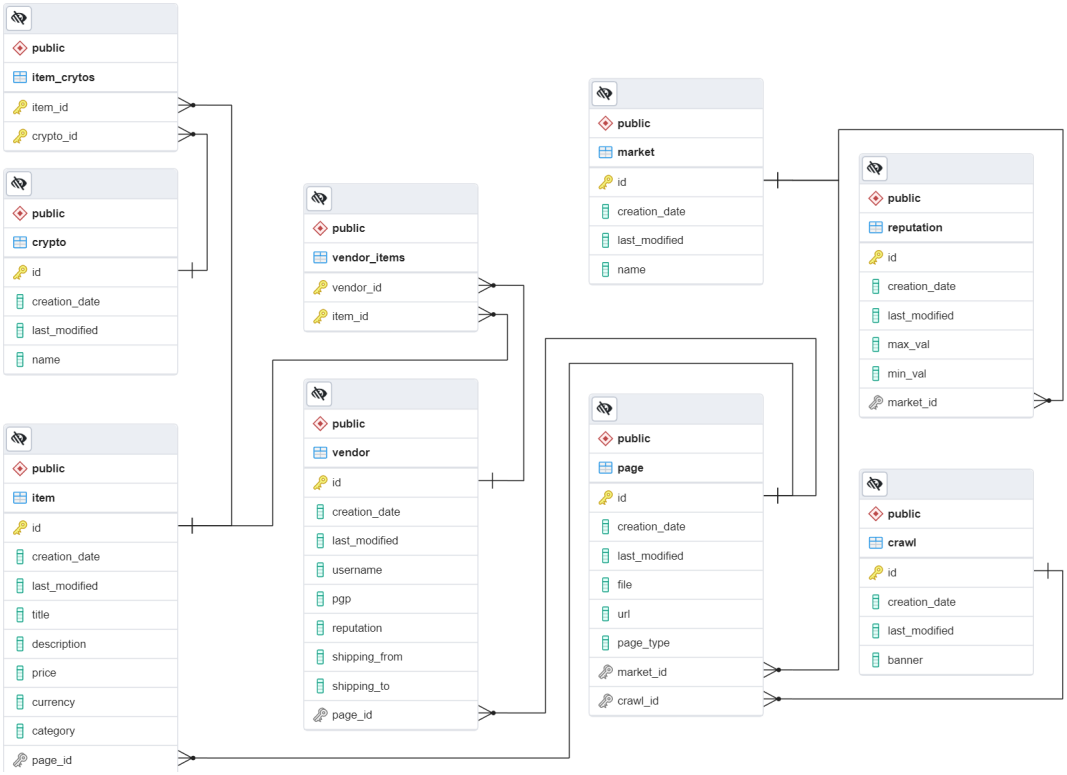

Figure 7.3: ERD (Entity Relationship Diagram) of the current database schema generated using the pgAdmin tool

Listing 7.2 gives an example of a *blueprint*. The example includes a group "comments" which contains two fields, "tags" and "user". An example of the data captured can be seen in **Listing 7.3**, a dictionary capturing two "comments" from two different users.

```
scraper: groups
struct:
  - name: comments # Group Name
    many: true # Expected many comments
    fields: # Group Fields
      - name: user
        instructions:
          - props:
              name: div
              attrs:
                  class: username
              limit: 1 # Limit to the first result
            attrs:
              - text
      - name: tags
        many: true # Expected multiple tags
        clean: \w+$ # Pre-processing
        instructions:
          - props:
              name: a
              attrs:
                href: /profile/view/
            attrs:
                - text
```

Listing 7.2: Example of blueprint structure used to group elements

```
{
  "comments": [ // Group
    {
      "user": "some user", // "user" Field
      "tags": ["tag1", "tag2"] // "tags" Field
    },
    {
      "user": "another user",
      "tags": ["tag3", "tag1"]
    }
  ]
}
```

Listing 7.3: Example of result using the listing 7.2 as a blueprint

7.2 Processor

This section summarises the development of the processor, previously introduced in **Chapter 6**. This module aims to label and categorise the listings stored in the database. The module uses a backup of the database formatted as a CSV file, which can be obtained using the CSV dump feature of *pgAdmin* and an SQL query to format the rows. The listing 7.4 shows the query used to extract the data, joining the tables *Page* with *Item*, *Market* and *Vendor*.

```
SELECT *, market.name market, vendor.username vendor
FROM public.page
    JOIN market ON page.market_id = market.id
    JOIN item ON item.page_id = page.id
    INNER JOIN vendor_items ON item.id = vendor_items.item_id
    INNER JOIN vendor ON vendor_items.vendor_id = vendor.id
```

Listing 7.4: SQL query to format listing and their relationships into a single table

This module is also used to visualise the results from labelling and categorisation in the form of tables and graphs. The section is further divided to explain the development of the models in **Section 7.2.1**, and the visuals in **Section 7.2.2**.

7.2.1 Models

The NLP models were implemented using the Python packages *Hugging Face* [58] and *Simple Transformers* [59]. The packages provide wrapper interfaces to build and train AI models, giving access to community pre-trained NLP models. This project uses models that can be trained in text classification tasks with two or more labels.

Tables 7.1 and 7.2 show the models selected for training divided by the number of labels with which the models were trained. These models were selected by filtering the the list of available models with the best performance for sentiment analysis

and text classification tasks, and sorted by the community rating. Furthermore, models fine-tuned for specific use cases were removed from the results. The tables show the results of their training in terms of training loss, learning rate, and training time, which can be used to evaluate the trade-offs between the time required to train the model, the average errors, the speed at which the model will learn new information, and the performance of the model. Regarding the performance, this is given by the evaluation parameters *F1-Score* and *accuracy* using a scale from 0 to 1. *Accuracy* calculates the percentage of correctly classified inputs. In the other hand, *F1-Score* combines the precision and recall of the model into a single metric, that is, calculating how well the model classifies an input.

Furthermore, the pre-trained models and respective configurations and checkpoints are included in the processor module in case they are needed for further improvements. It is important to remember that only English-based DNMs were crawled; however, the dataset includes listings written in multiple languages. Moreover, the M2 models used an additional label called FP (False Positive), due to the need to allocate false positives from the M1 model labelling.

Labels	Model	Training Loss	Accuracy	F1-Score	Duration	Learning rate
2	DistilBERT base uncased SST-2 [60]	0.246	0.910	0.929	1h 37min	9.67e-7
	BERT base [61]	0.232	0.912	0.933	7h 50min	4.22e-7
	RoBERTa base OpenAI [62]	0.198	0.892	0.910	4h 4min	1.51e-7

Table 7.1: Results of training the M1 sentiment analysis models using two labels

Labels	Model	Training Loss	Accuracy	F1-Score	Duration	Learning rate
7	BERT base [61]	0.758	0.772	0.761	5h 24min	0
	XtremeDistil 16 h384 uncased [63]	0.984	0.736	0.709	2h 51min	0
	MiniLM L12 H384 uncased [64]	1.913	0.693	0.630	1h 41min	0

Table 7.2: Results of training the M2 text classification models using the six classification labels for CSRP

Table 7.3 shows the set of parameters used for the training. The models were trained twice to give a fair comparison, from which the best runs were selected. The reason to train each model twice is due to the variance introduced by the training parameters "early stopping" and "warm-up steps", which significantly impact the training result. Furthermore, the models were trained using the same training and evaluation sets, using a set of **2000 labelled listings** divided by a ratio of 80% for training and 20% for evaluation. This method was preferred over K-fold cross validation, to give a fair chance to the models to learn from the same datasets, The disadvantage of K-fold is that introduces randomness on each run by splitting the data into two random samples, using "K" as the term to define the percentage of inputs used for validation. These fine-tuning parameters were chosen by trial and

error in testing runs with three smaller datasets of labelled listings containing 100, 500 and 1000 listings.

Parameter	Value	Description
Training batch size	16	Batch size used for training
Evaluation batch size	64	Evaluates while training using a batch of the set
Warm up steps	600	number of first steps using a lower learning rate
Learning rate	3e-5	Determines the impact of newly acquired information
Weight decay	0.01	Weight penalisation
Early stopping	True	Stops the learning on the current batch earlier than expected
Sliding window	True	Creates windows for inputs overly large

Table 7.3: Parameters used to train the machine learning models

Figure 7.4 illustrates the training loss over time of the M1 models. It is worth mentioning that, for visualisation purposes, the graph lines have been smoothed to a ratio of 0.3, and the graph does not show the values from the warm-up steps. When comparing the graph with the results of 7.1, we can see that, while the BERT base model gives the best overall performance, the DistelBert learning is smoother and drastically shorter. However, increasing the number of times DistelBert sees the data (3 times by default) does not result in an equally significant precision growth; instead, the model becomes overfitted and fails to classify listings correctly. In the case of RoBERTa, increasing the weight decay and adding an extra epoch resulted in values similar to the BERT base model, while increasing the training duration by a third. This increase in performance is not significant enough to justify the time spent fine-tuning the model. Finally, the changes in learning rate and weight decay for the BERT base model showed promising results, although the model performance decreased slightly compared to the current setup. This may be improved with a considerably larger training set and increasing the number of epochs, although the training time will grow accordingly.

For the requirement **FR3**, we choose **DistilBERT** as the final **M1** labelling model with an approximately 93% F1-Score. This model is faster than competitors in training and labelling entries, whereas false positives are milder compared to the other two models. For example, it is reasonable that the model mislabels some listings with long descriptions that use a generic list of tags or passwords. However, it is critical that the model can identify listings that mention malware or database leaks. Although the other two models have a similar F1-Score, the models showed a more a more strict interpretation of the listing descriptions, responding worse to more detailed descriptions.

Unlike M1 models, M2 models were trained using five epochs. This change gives more space for the models to learn the different labels, considering some labels include fewer examples than other categories. However, **Table 7.2** indicates that the

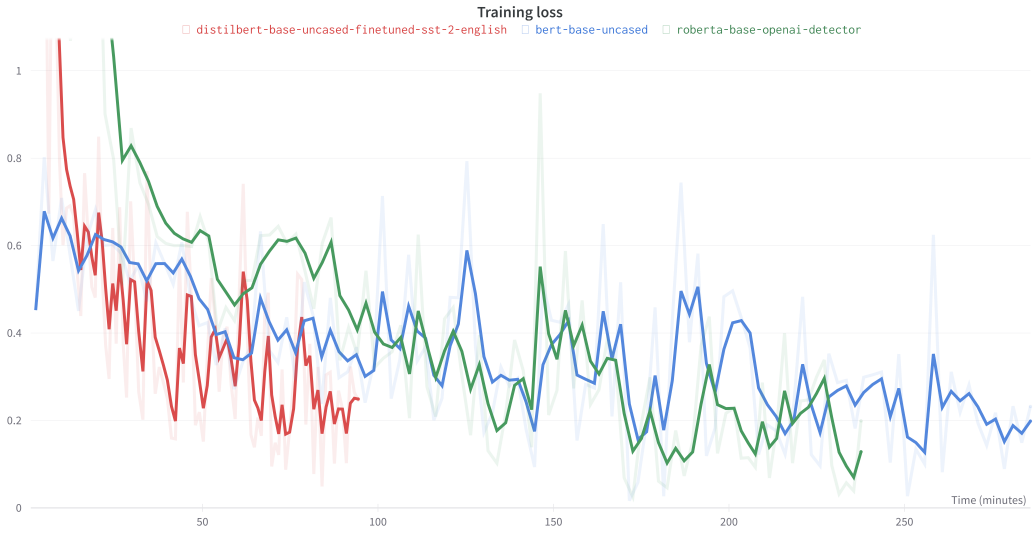

Figure 7.4: Training Loss over time from training M1 models

models show little to no improvement at the end of their training. This indicates that further training using the same training set would not yield significantly better results.

Moreover, **Figure 7.4** shows clear differences between the models. First, XtremeDistil is a promising model that cuts the training time by roughly five times compared to BERT base, while achieving a comparable accuracy using similar parameters. However, the model presents limitations using the current setup, although, it is not clear that XtremeDistil would outperform other models in equal conditions. A larger dataset and a smaller learning rate may return better results. Furthermore, the MiniLM model did not give an accurate enough estimation compared to the other two models. Although it is not the objective of the experiment, it is worth mentioning that this model performed remarkably well when classified listings written in languages other than English.

In summary, the BERT base model outperformed the competitors in terms of accuracy and F1-Score by a large margin, at the expense of requiring roughly four times longer to train than the others. The training of this model is considerably more consistent, suggesting that a larger dataset using a similar setup will significantly increase the model performance. Therefore, **BERT base** was the **M2** model selected to implement requirements **FR3** with an F1-Score of 77%. Future research could use the BERT base model checkpoint used in this project to classify listings.

This implementation method using two different models acts as an indirect validation device. Generally, the accuracy F1-Score with the M2 models is relatively lower than expected. Therefore, the M1 model can be used as a filter for false positives, filtering the dataset for entries labelled *Positive* or *FP*.

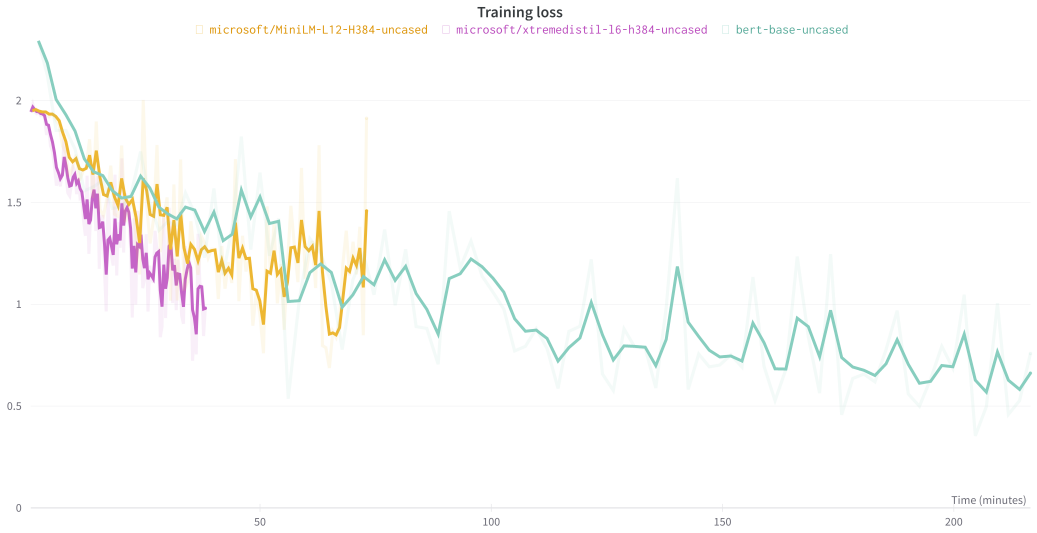

Figure 7.5: Training Loss over time from training M2 models

7.2.2 Visualisation

The module includes Python Jupyter Notebooks [65] with separated blocks for each study. Jupyter Notebooks are primarily used for interactive computing, allowing us to create virtual environments where we can run applications in any order while maintaining a common state of the memory. This is especially useful for visualising graphs and other figures while processing information. The following Python visualisation packages are used to display the data on the notebooks.

- **Pandas:** Data analysis and manipulation tool that vectorises datasets. It accepts inputs directly from CSV files.
- **Matplotlib:** Standard plotting Python library. This package is used primarily to control the graphs' positioning, colours, labels, and other visuals.
- **Seaborn:** Visualisation library built on top of matplotlib that provides simplified methods to draw figures.
- **Plotly:** Advanced graphs and figures with interactive interfaces. Valuable to see changes from time series and manipulate graphs in real-time.
- **Wordcloud:** Package to create wordclouds. Helpful in understanding the predominant terms in large text input datasets.

The graphs and other charts are included in the next chapter, **Chapter 8**, as part of the listings and vendor analysis. The chapter includes the complete list of figures used to study the dataset's content. Furthermore, the notebooks are included alongside the MS code.

Chapter 8

Analysis

This chapter summarises the analysis of the data collected throughout the project. The dataset contains DNMs listings and vendors captured from continuous crawls, covering March and April 2022. Due to the implementation of MS, the dataset does not need to be separated into snapshots. Instead, data points are collected and aggregated in the same database. However, it is possible to separate the data into smaller snapshots using the table *crawl* to group the entries, as specified in the database schema 7.3.

The list of DNMs utilised for this research was selected utilising basic criteria. From the grand total of DNMs, the initial list included only English-based DNMs, as stated in requirement **NFR2**. Then, DNMs must contain CSRPs, although this does not necessarily need to be the focus of the DNM. Lastly, the list is ordered by market size. **Table 8.1** includes the list of DNMs considered for this project, along with the criteria mentioned above. Large DNMs contain more than 20k listings; medium DNMs contain between 10k and 20k listings, and small DNMs contain less than 10k listings. Note that some DNMs closed during this project, which had an impact on the overall results, namely *Versus*.

The remainder of this chapter focuses on the listings and vendors captured from the DNMs. The general content of the labelled listings is analysed in **Section 8.1**, which offers a closer view of the characteristics of the listings found in each category. **Section 8.2** dives into the vendor distribution and briefly analyses the correlation between the vendors and DNMs. Finally, the chapter concludes in **Section 8.3** collecting the most important key-points learnt throughout the chapter.

DNM	I2P	Tor	Cyber	Size
Alphabay	✓	✓	✓	Large
DarkFox	✗	✓	✓	Large
Tor2Door	✗	✓	✓	Large
Versus	✗	✓	✓	Large
Bohemia	✗	✓	✗	Large
ASAP	✗	✓	✓	Medium
Vice City	✗	✓	✓	Medium
Incognito	✗	✓	✗	Medium
Kingdom	✓	✓	✓	Small
Cypher	✗	✓	✓	Small
Archetyp	✗	✓	✗	Small

Table 8.1: List of DNMs available sorted by size

8.1 Listings

Turning now to the listings, this section uses the crawling results after applying the heuristic H6. However, a table containing a summary of the results before applying the heuristic H6 can be found in **Appendix A**. In addition, the content of this section aims to provide an understanding of the population. Therefore, the figures presented in the remainder of the section do not include outliers and, in many instances, take a closer look at median, mean, and variance values. As a reference, we include some of the figures with outliers in **Appendix A**.

Regarding the results of the crawl, **Table 8.2** summarises the content of the dataset after applying the H6 heuristic. The table shows the total number of listings crawled per DNM, representing the number of listings found in categories that were more likely to contain CSRPs. The next column identifies the number of *Positive* listings, showing a much lower number compared to the initial results. The reason for this difference is that even in categories presumably containing CSRPs, there is a proportional amount of products non-related to cyber security, such as passports, templates, or fake money. The rest of the columns include the number of positive products found in each category. It is important to mention that due to the extra label *FP*, the sum of the category columns is not equal to the number of positive listings. This is due to false positives when labelling the listings using the M1 model.

DNM	Fraud	Malware	Media	Services	Software	Positives	Total Products
ASAP	630	52	104	36	93	915	**2834**
Alphabay	2275	125	232	134	176	2942	**5406**
Cypher	384	120	213	6	103	826	**1790**
DarkFox	5171	255	400	101	677	6604	**13436**
Kingdom	688	123	181	117	71	1180	**2685**
Tor2Door	1743	197	370	119	280	2709	**4679**
Versus	1683	69	102	139	251	2244	**5367**
Vice City	1110	172	308	44	121	1755	**3684**
Grand Total	**13684**	**1113**	**1910**	**696**	**1772**	**19175**	**39881**

Table 8.2: Total amount of listings per market, positive listings, and positive listings per category

Figure 8.1 depicts the total distribution of positive listings across the DNMs. It is interesting to see that despite the differences in market sizes, the number of positive listings remains closely similar among the larger DNMs. Compared to the total amount of listings, the concentration of CSRPs is similarly large for **Alphabay**, **Kingdom** and **Tor2door** with roughly the 74% of the listings marked as positive. On the other hand, **Versus** and **ASAP** fall to 52% and 44% respectively. This alone indicates that there is a degree of specialisation in regard to cyber security.

Figure 8.2 expands on the previous chart, illustrating the percentage of the total number of listings per category and DNM, and the internal distribution of the categories. The figure is a treemap of the DNMs and the distribution of their listings. The sizes of the boxes represent the percentage of the total listings for some paths. For example, Alphabay has 30% of the listings, and of that percentage, 21.12% are fraudulent listings. The value of this illustration lies in the influence of the DNMs on each category. However, the data suggest that the most traded asset is fraudulent information in all DNM.

Figure 8.3 offers another perspective to the distribution of the listings, showing the amount of listings per category on each market. Looking at the height differences between the categories is sufficient to verify that fraud constitutes the majority of the listings offered. However, it is worth mentioning that the percentage of fraud listings is much lower on Kingdom, roughly 49.6% compared to the 68.7% on average. For the rest of the categories, the trend shows that media and software are the second most popular listings, occupying the 10% of the listings in each case. Finally, software and malware constitute 5% of the listings each.

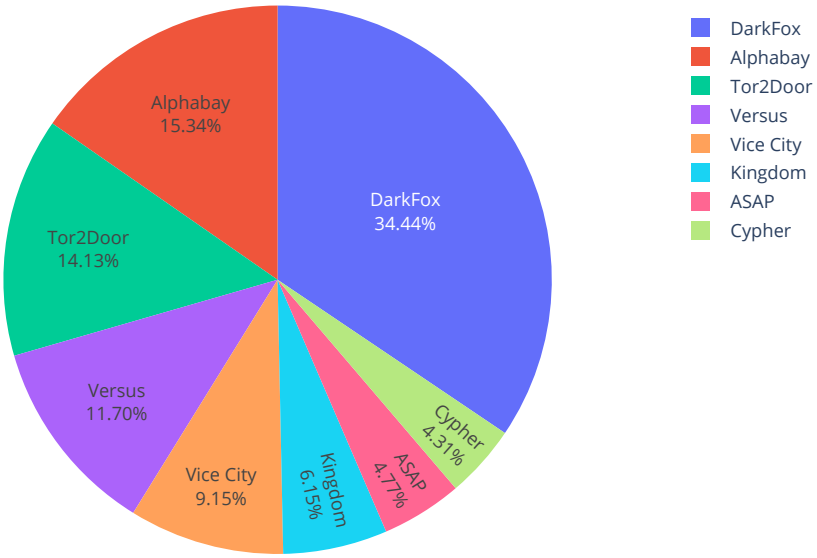

Figure 8.1: Listings distribution per DNM

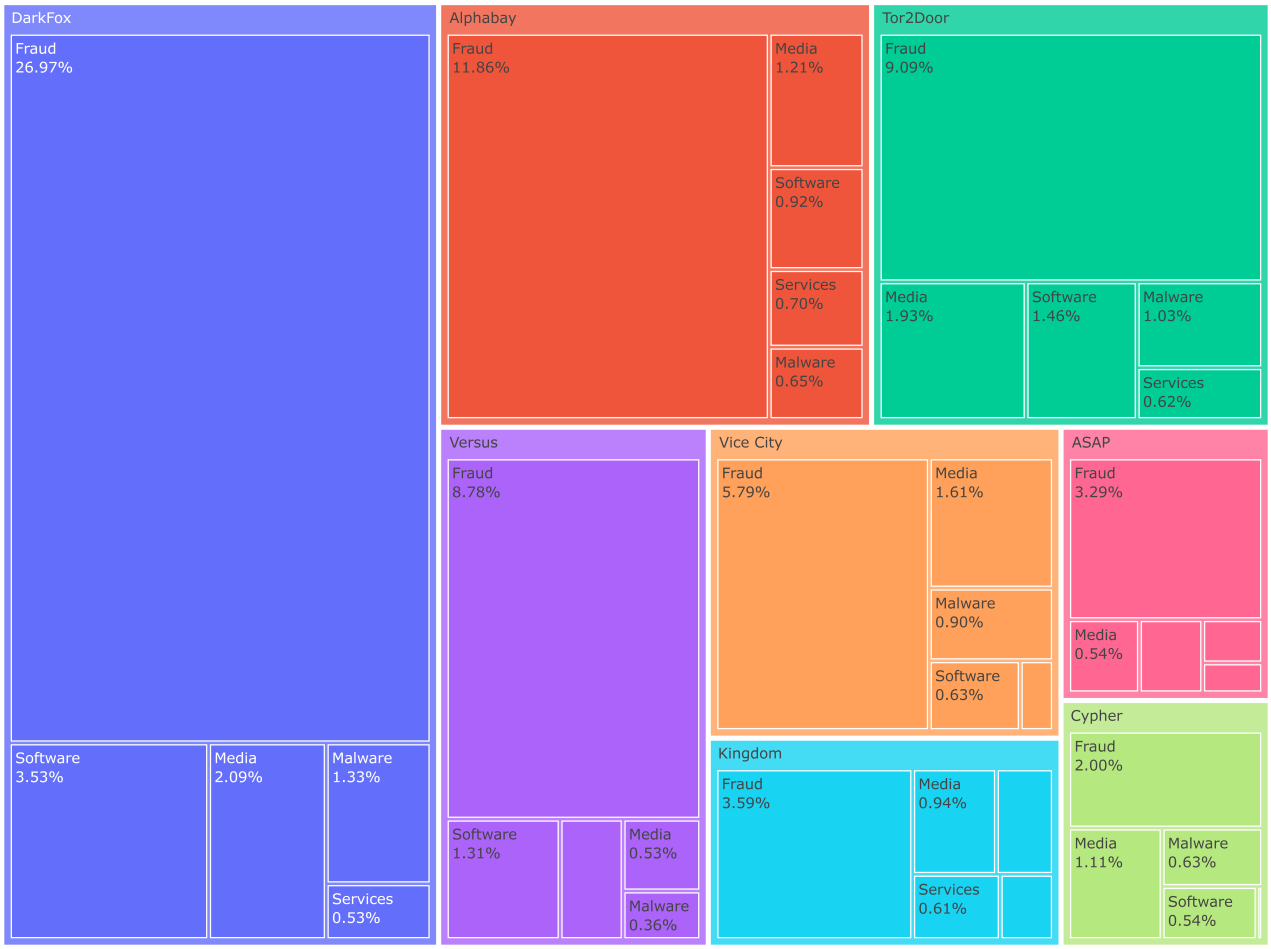

Figure 8.2: Listings distribution per DNM per category

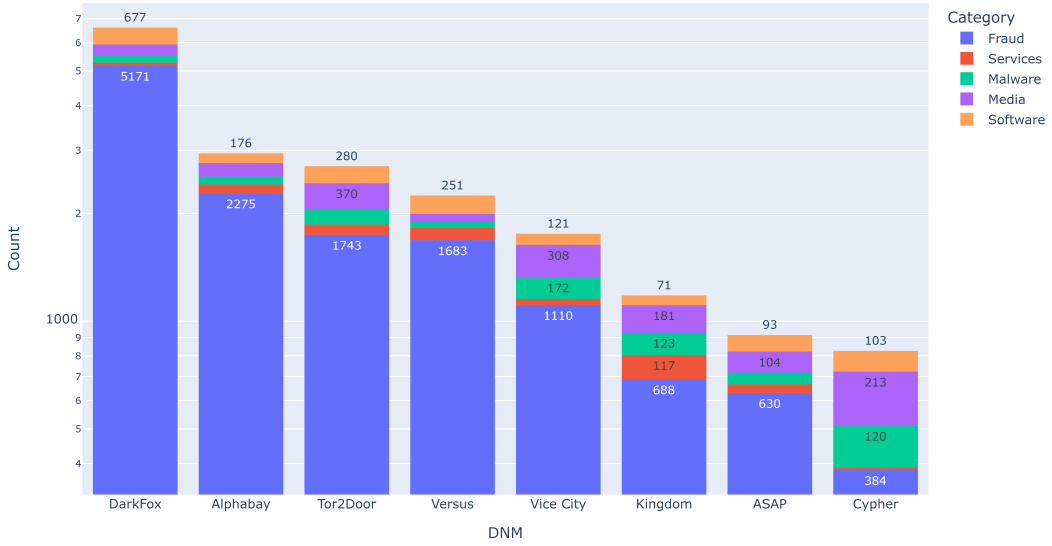

Figure 8.3: Count of listings per DNM and category

Furthermore, **Figure 8.4** reveals that the DNMs that show the least diversity are, in turn, the most expensive. The *boxplot* boxes determine the interquartile range or the range in which most values can be found. Furthermore, the red lines at the extremes of the boxes and the black vertical lines are called *whiskers*, and determine the maximum and minimum values of the confidence intervals. For this figure, a 95% confidence was chosen to represent the maximum and minimum values displayed. Finally, the blue lines within the boxes represent the median value, that is, the most common value in the dataset. As can be seen in the figure, Versus and ASAP contain the most expensive listings, reaching up to 160 EUR and, in the case of Versus, the median price is higher than in the rest of the DNMs, situated at 18 EUR. What is striking in this chart is the difference with Kingdom, the cheapest DNM, which is the most diversified.

Figure 8.5 shows this information in more detail, differentiating between the category prices. As expected, fraud products show the highest prices among the categories. It is interesting to see that the median price for any category oscillates between 15 and 20 EUR, suggesting that most products are inexpensive. However, the whiskers tell us that it is possible to find listings reaching prices up to 100 EUR in for most categories. For example, Kingdom peaks on the malware category with a large variety of prices, reaching up to 100 EUR, where the upper quartile is situated at 90 EUR. In comparison, the next highest quartile is held by ASAP in the category services, with a value of 50 EUR. Despite using the heuristic H6 to filter

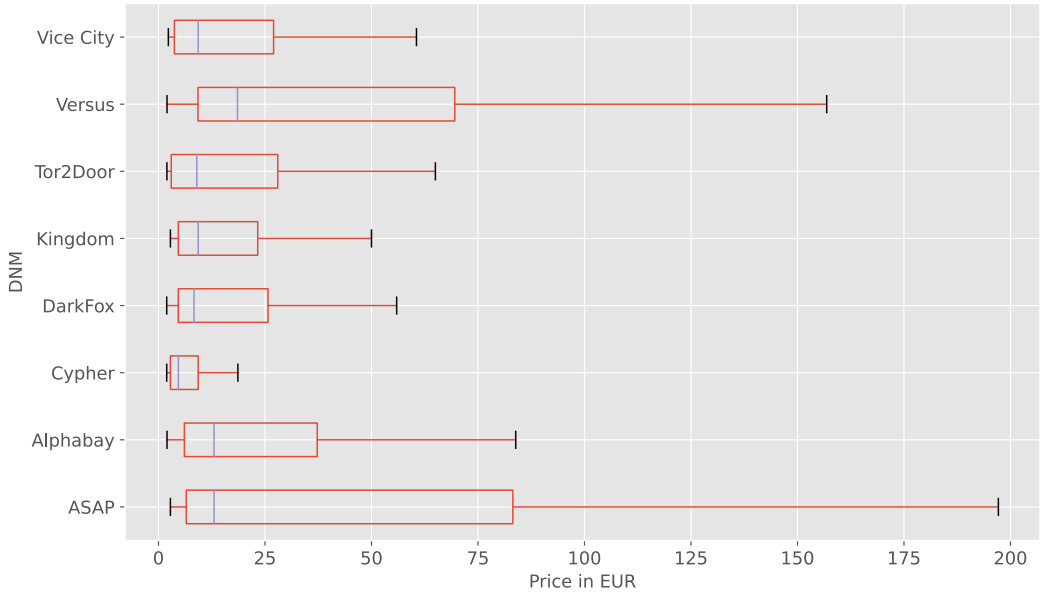

Figure 8.4: Price boxplot grouped by DNM without outliers

holding prices, the dataset contains some duplicated products with a surprisingly high price, which impacts the length of the whiskers, although not the size of the boxes.

Regarding duplicated listings, it is interesting to look at the entropy of the descriptions for each category. The entropy value is typically used to calculate the degree of disorder inside of a group, or the normal distribution of the content. This metric is often used to detect plagiarism [66]. By calculating the entropy of the description values in the category, we can estimate similarities between the products in the cluster. A greater value of entropy is similar to saying that there is small correlation between the inputs in the cluster. In addition, the entropy value also measures the amount of information in a group. Therefore, lower entropy values can be found in clusters with similar inputs. When referring to text, lower entropy is related to low variance between the inputs, i.e., texts are similar.

Table 8.3 summarises the calculation of the entropy values using base 10 for each category. The entropy has been calculated using the probabilistic distribution of the descriptions, i.e., the probability of finding an equal description in the cluster. Values between one and three are considered high correlation, while values higher than three suggest greater randomness.

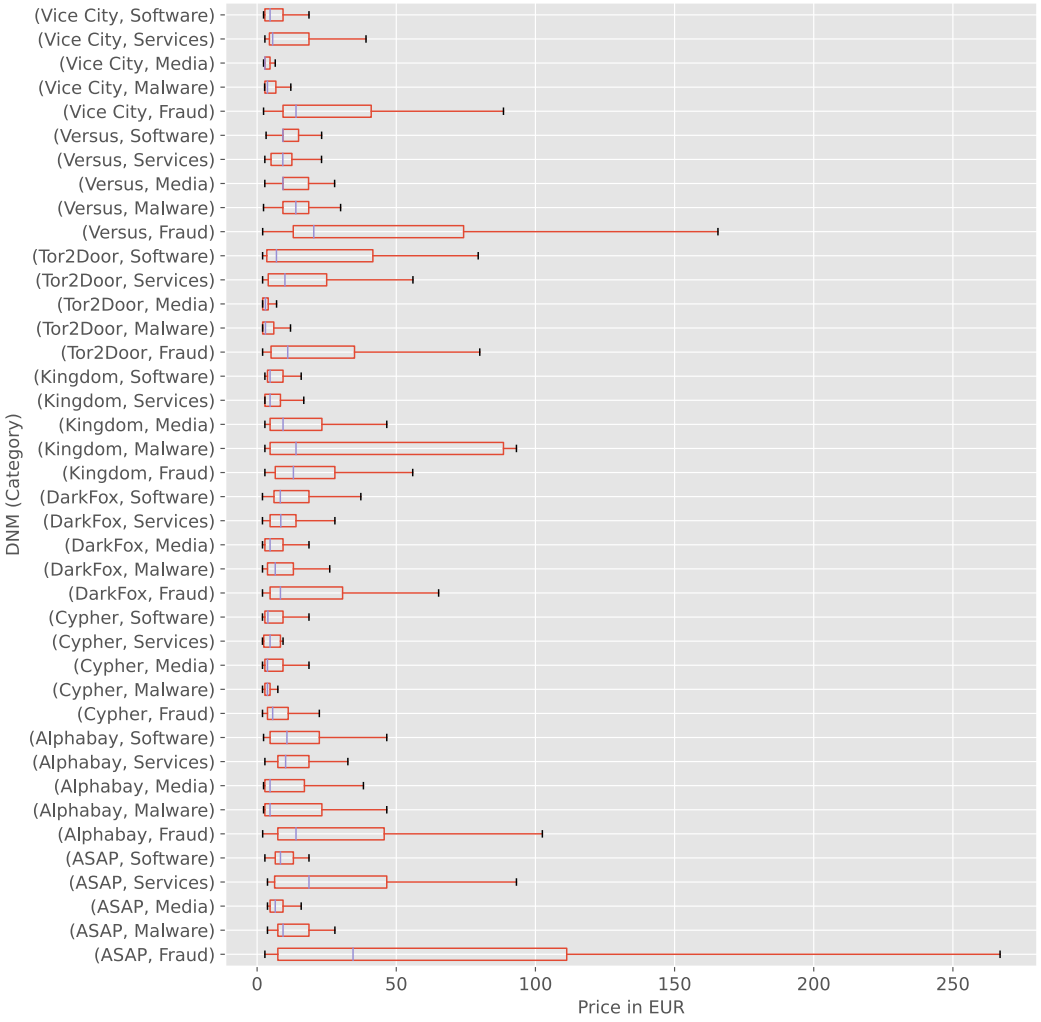

Figure 8.5: Price boxplot grouped by DNM and category without outliers

Comparing the results with **Table 8.2**, there is a mild correlation between the number of listings per category and their entropy. However, the entropy values are generally low, even for the bigger categories such as fraud, which would be expected to have a much higher entropy than other categories. The results suggest that vendors use similar descriptions for their catalog.

Category	Entropy
Fraud	3.690
Malware	2.786
Media	2.970
Services	2.494
Software	3.046

Table 8.3: Entropy of the category clusters

In light of this evidence, it is reasonable to assume that fraudulent information is the most lucrative business for hackers to trade in DNM. However, it is not yet clear the content of the products for sale and the implications of these listings. To answer this question, the remainder of this section has been further divided to analyse the content of each category using the name of the category as the title of the section. The category *physical* has been removed from the analysis, since only two records have been registered.

8.1.1 Media

Figure 8.6 illustrates the most common words for media listings using the title as data to generate the wordcloud. The opacity and size of the words represent the frequency of the words used in the title of the listings, while the colour strength is used to differentiate words. This wordcloud contains listings from all DNMs. As can be seen from the picture, the most common products are either tutorials about carding methods, mega-courses, and packs.

It is interesting to see that one of the most typical products is *background reports*. This article offers guides on how to find personal information from US citizens, such as credit reports, driver's licences, and addresses. The product can also be found with titles such as "Fullz USA". In contrast, there are no similar products that target other countries. Furthermore, there is a lower volume of direct hacking tutorials and similar guides, in favour of mega-courses and packs. These listings contain large amounts of e-books on multiple topics. The dominant product in the

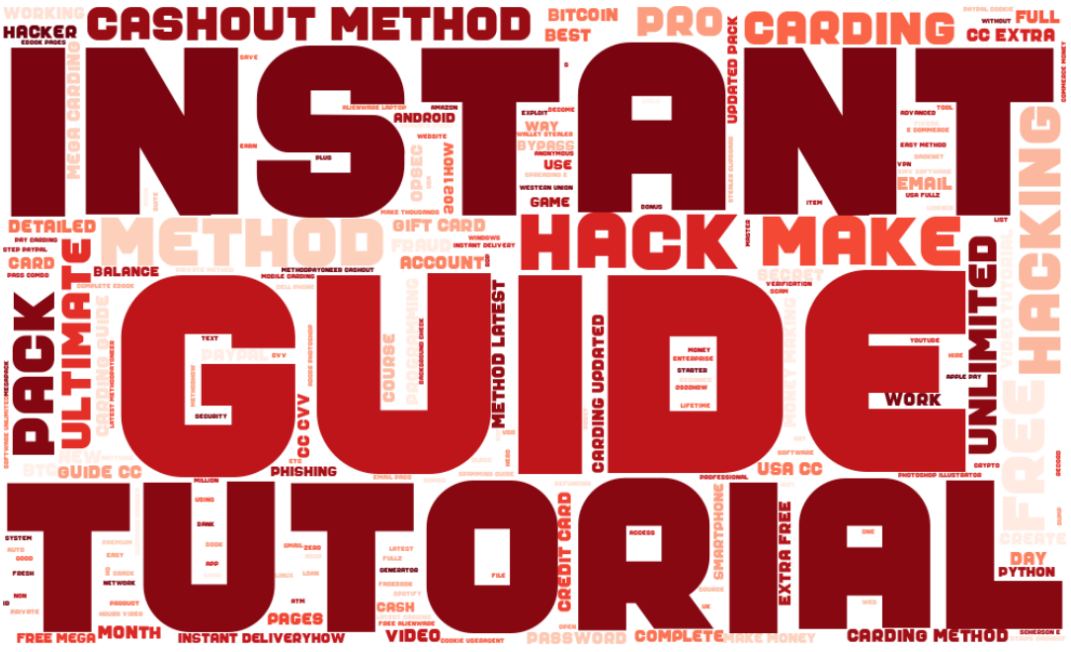

Figure 8.6: Wordcloud using the media listing titles

media category is carding methods. These guides cover many distinct services and technologies, with prices ranging from 1 to 200 EUR. After inspecting the media listings, the following threats were identified.

Hacking Tutorials

Although the majority of the advertised content is available on the clearnet, this type of content is of special interest for those whose access to the Internet is limited by any another entity. However, the content of these listings typically requires advanced technical knowledge, such as programming and networking. The learning curve may be dependant of the technology, but the fundamental knowledge requires a large time investment before it becomes useful to the user.

Background and PII Checks

Background checks and PII (Personal Identifiable Information) gathering methods are closely related to phising attacks. These guides do not require technical expertise, focusing on the methodology used to run scamming and spamming campaigns.

Carding Methods

Carding methods affect a large number of services, such as Paypal [67], Amazon [68] and Apple [69]. Carding is the common term referring to unauthorised traffic and withdraw of any kind of wallet, bank account, or digital token that can be traded for another product, such as gift cards, NFT or holding accounts. The volume of these products indicates that there is a high demand for NFT and crypto-currencies withdrawing methods. These methods are typically part of larger scams [70], involving a large range of steps and procedures. For example, NFT can be minted (transformed into a blockchain token, typically Ethereum) using a stolen credit card, deposited into a hacked wallet, and traded on the Darknet for another product. Tutorials involving NFT and Bitcoin are traded for 200 EUR on average.

8.1.2 Malware

As can be seen in **Figure 8.7**, malware prices follow the Darknet trend for digital goods. The figure illustrates the distribution of listings across DNMs using their price. In this case, prices can be seen crowded below 50 EUR, with various instances of small clusters below 100 EUR. As mentioned previously, Kingdom is a particularly interesting case in this category, which prices reach 100 EUR while retaining a similar concentration on the lower end of the prices.

This can be further seen in **Figure 8.8**, which represents the normal distribution of listing prices in DNMs. This figure includes a secondary graph that shows the concentration of the listings for every price range. These ranges form the bars represented above, and the lines show the normal curve or KDE (Kernel Density Estimation), i.e., the internal distribution of the listings. The curves on the lines will grow when multiple kernels or bars are closer to each other, and lower the kernels are smaller and sparse. This further supports the findings of the previous Figure 8.7, finding most of the listings on average at approximately 7 EUR.

The lower volume of listings on this category permit us to granulate the analysis of the content. This set presents a high concentration of multipurpose hacking tools, relegating targeted malicious software to a niche space. However, the list of advertised malicious software contains multiple instances of currently relevant known names. Regarding tool packages, the listings include a variety of software examples, such as keyloggers, carding software, and cracked software (hacked software to bypass licences and authentication). The majority of the products centre on cracking passwords and stealing crypto-wallets for an average of 10 EUR a pack, depending on the software advertised. These products do not suppose a major threat by themselves, as they do not have the capacity to infect or spread to other systems. Instead, they are generally used in systems already compromised to gather information. For example, carding software will be installed in the victims browser as an agent, reporting to the attacker when a form is filled.

Figure 8.7: Prices histogram for **malware** listings under the 95% confidence interval. Count represented using logarithmic values

Figure 8.8: Prices distribution for **malware** under the 95% confidence interval

The remainder of this analysis differentiates between the type of malware to describe the intention of the software. The following malware families have been identified among the set of listings: Crypto-miners, RATs (Remote Access Trojan), Botnets and Ransomware. In the following sections, we categorise the listings in relation to the above-mentioned malware families, including details such as average price and most targeted systems. Lastly, the list of malware found in the dataset is included in **Appendix A**.

Crypto-Miners

The main incentive to use crypto-miners is the ability to finance operations using compromised machines as a passive source of income, exploiting the resources of the system to the attacker's benefit beyond their normal use. Crypto-miners affect a large number of systems, from IoT devices with limited resources such as hospital beds [71]; to commercial systems such as personal laptops and telephones. The bigger issue with this type of malware is the resources consumed by the software, which induces a high CPU usage, stopping normal execution of other tasks on the system. The crypto-miners found in the dataset reference the cryptocurrencies Monero [72], Bitcoin [73] and Zcash [74].

Remote Access Trojan

This products attempt to exploit certain vulnerabilities on the targeted systems to create backdoors, and deploy other malware [75]. The kits typically contain software to monitor the victim's device and modify the system, and include other notable modules like hVNC (hidden Virtual Network Computing), which allows the attacker to control the state of the device remotely using the current user privileges on a virtual hidden desktop. Furthermore, the listings included in the dataset indicates that there is a small population of vendors dedicated to selling specialised malware. With few exceptions, the listings can only be found on single instances, primarily at Tor2Door and Alphabay with prices from 5 to 25 EUR, and rare cases of similar products sold for 200 EUR approximately. Most of the RAT kits available in the dataset belong to well-known families currently active. For example, *njRat* and *RedLine*, two of the most popular RAT kits [76].

Botnets

Three different botnets have been found for sale in the dataset. The products have been found on single instances in all DNMs with the exception of Kingdom, which contain none. Moreover, the botnet source code is sold for 10 EUR approximately a piece.

Ransomware

The dataset includes a large list of ransomware examples, sold in bundles or as standalone products. This type of malware is considered dangerous and a growing menace for organisations [77]. The malware is designed to encrypt the content of the system, blocking access to the device until a ransom has been paid. Bundles of older ransomware are typically found in lead DNMs such as Alphabay and Tor2Door, with prices lower than 10 EUR in all instances. However, multiple collections of newer ransomware were found in ASAP for roughly 200 EUR. Finally, standalone ransomware products could be found in Tor2Door and Kingdom, with prices up to 80 EUR.

8.1.3 Software

The listings found in the software category present characteristics similar to those of other categories. **Figure 8.9** illustrates the population of the category under the 95% confidence interval. The density of listings under 50 EUR is significantly higher than other clusters. Moreover, the figure shows interesting cases of listings priced between 120 and 140 EUR. However, looking at the listings reveals that these products are, in fact, duplicated posts of Bitcoin wallet crackers and software to clone credit cards. Although the figure does not show outliers, the products found at the extremes are, in most cases, cracked versions of commercial software, such as Adobe [78] and Microsoft [79] products. The consequences of selling these products on DNMs are mostly economical and detrimental to the image of the company.

Continuing now with **Figure 8.10**, the figure illustrates the prices distribution at the point of higher density, which are mainly situated at 10 EUR and below. The figure indicates that there are two distinct ranges for most DNMs, separated at approximately 5 EUR. **Figure 8.11** help to visualise the content of these listings. In addition to crypto-wallet crackers, credit card cloning and commercial software, the category contains a high volume of brute force password crackers and virtual machines prepared for hacking. The words *account* and *premium* refer to software supplemented with premium accounts, lifetime licences, etc. This suggests that DNMs have different prices for similar software products, as the number of different products is rather small, and their content only changes in relation to the name of the software advertised. For example, the dataset contains more than 130 similar listings referring to phising pages targeting different services for a prices between 1 and 9 EUR depending on the DNM. It is worth mentioning that this kind of duplicated listings is mostly attributed to the same vendor names across all DNMs.

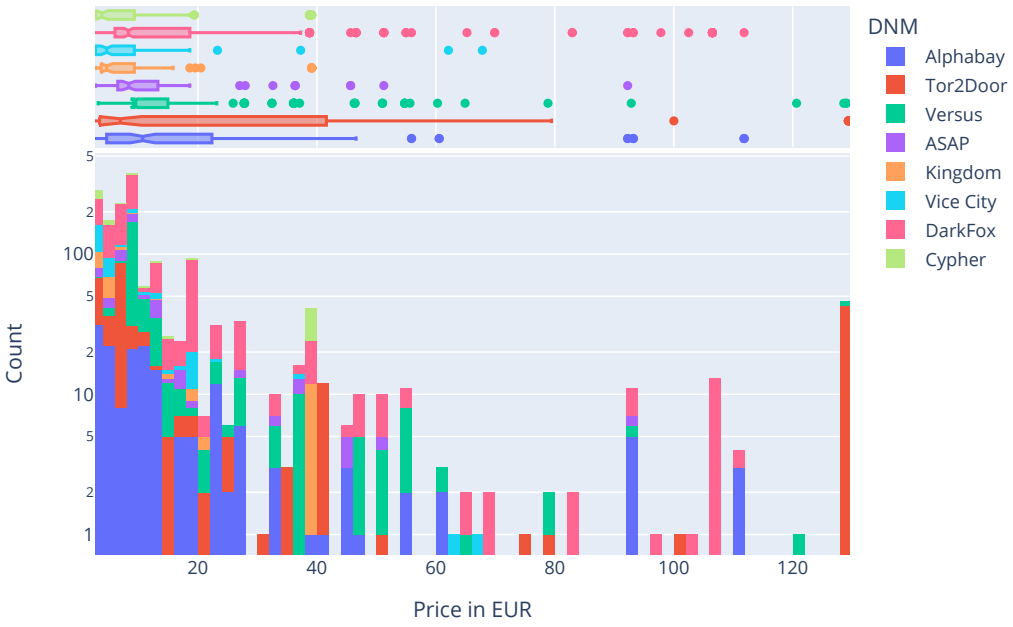

Figure 8.9: Prices histogram for **software** listings under the 95% confidence interval. Count represented using logarithmic values

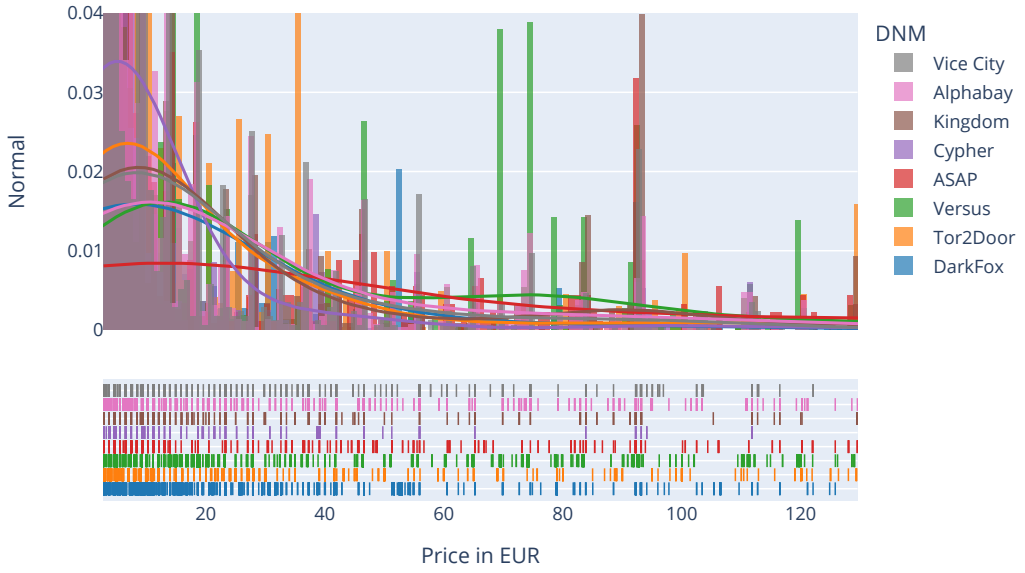

Figure 8.10: Prices distribution for **software** under the 95% confidence interval

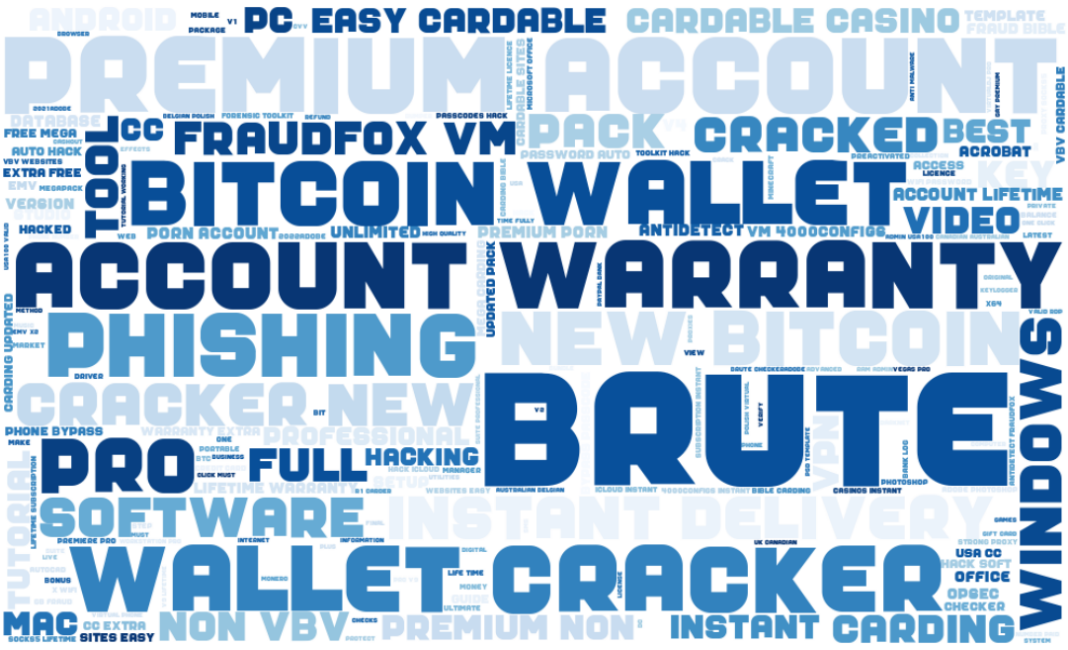

Figure 8.11: Wordcloud using the **software** listing titles

In conclusion, software products can be separated into two groups: hacking tools and hacked software. The hacking tools group consists of password crackers, crypto-wallet crackers, phishing tools and cloning software. Password crackers typically rely on brute-force methods seeded with dictionaries of common passwords. Having an account hacked leads to the disclosure of personal information and may give access to personal resources, such as bank accounts.

8.1.4 Fraud

The sample contains over 70% of the population, evenly distributed among the markets, with the exception of Kingdom, which percentage of listings is approximately 40%. The details of the fraud sample are illustrated in **Figure 8.12**. This figure includes prices up to 100 EUR, showing a large number or small clusters, suggesting that it is possible to find products outside of the normal distribution with relative confidence. However, the normal price for this category remains under 100 EUR.

Figure 8.12: Prices histogram for **fraud** listings under the 95% confidence interval. Count represented using logarithmic values

Figure 8.10 focuses on the distribution of the listings below 30 EUR. At 30 EUR, the concentration of listings starts to decay, indicating that most listings can be found below this price. As the figure suggests, the median average is approximately 13 EUR per product. The median of the sample is slightly higher than other categories, while containing many more examples than the rest. This is an indicator that the demand is higher for this type of product than for the other categories, thus, providing the most revenue.

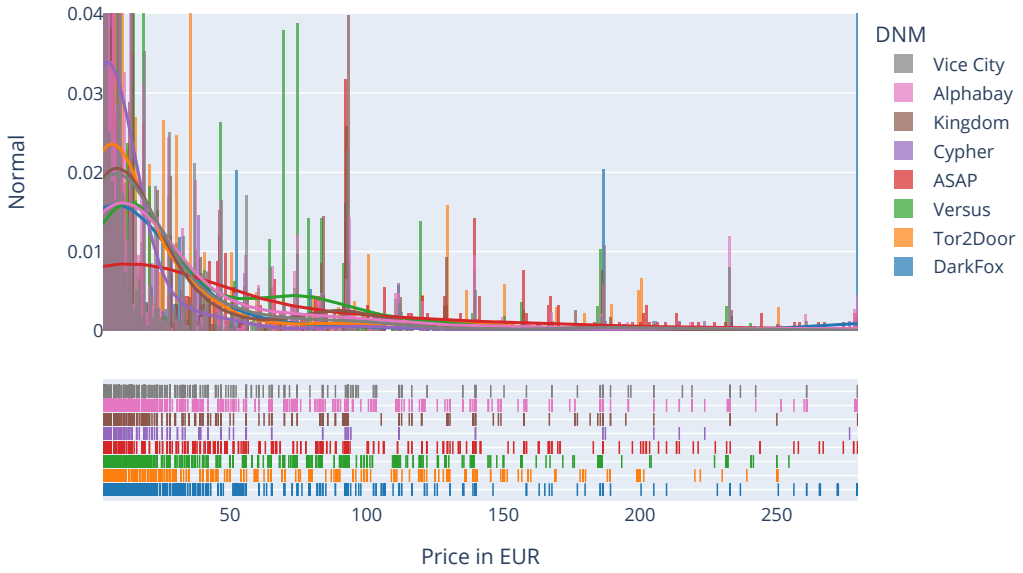

Figure 8.13: Prices distribution for **fraud** under the 95% confidence interval

As **Figure 8.14** indicates, the most common listings include carding, hacked databases and miscellaneous accounts. What is most interesting to see is that despite the amount of services dedicated to carding, the data contain a significant amount of listings specialised in this type of product. Most premium accounts require a payment method linked to the profile; therefore, it is not surprising to see a larger amount of bank details than accounts. Lastly, hacked databases seem to be an increasing phenomenon rather than a sporadic finding. The content of these groups is described in the following sections, where the groups are briefly examined.

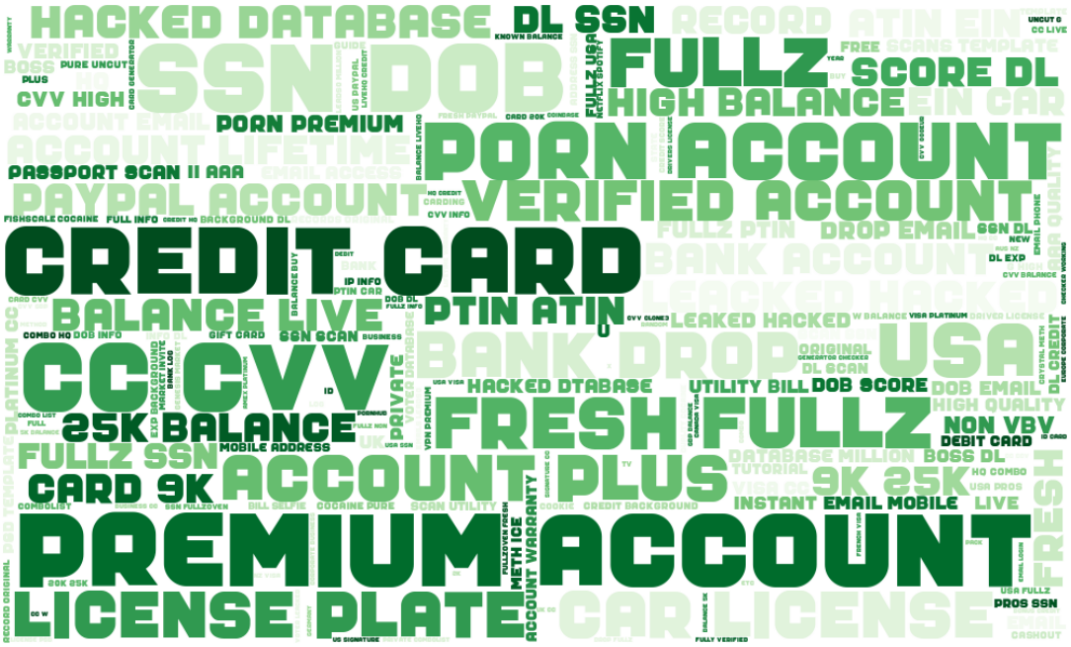

Figure 8.14: Wordcloud using the **fraud** listing titles

Carding

The fraud sample contains nearly 4000 records related to carding, approximately 45% of the set. Among those listings, the most popular are credit cards and banking details, however, it is possible to find multiple instances of gift cards. The price for a credit card highly depends on the balance, country of origin and type of the card. For example, a platinum VISA card with a balance of 20.000 dollars from the US, can reach prices of 70 EUR, while a similar card from Australia may be seen with prices close to 20 EUR. However, the price varies from vendor to vendor, therefore, it is only safe to say that banking details can be found for prices between 10 and 70 EUR.

Hacked Databases

The dataset contains approximately 700 database listings, roughly 8% of the total of the fraud sample. However, the sample is not evenly distributed across DNMs; database records are more likely to be found in Tor2Door and Versus, while Kingdom and ASAP contain the fewest database listings. In addition, most databases are published with prices between 10 to 50 EUR, where the content of the databases primarily includes information from Internet services, votes, US bank records. Although, it is possible to find instances of more expensive databases, reaching over 400 EUR. The price of the database seems to follow the date of the leak rather than

the content. For example, a *Bureau Van Dijik* [80] leak allegedly from 2022 can be found with a price of 416 EUR. Similarly to other previously seen groups, this type of product is mostly advertised by the same vendor names.

Miscellaneous Accounts

Approximately 45% of the fraud listings fall into this group. The content of this type of listings is typically offered in the form of username and password, for prices between 2 to 10 EUR. Most of the accounts belong to pornography and media streaming services which original prices start from 7 EUR onward. Therefore, it is not surprising to see listings advertised for less than the original price.

8.1.5 Service

This group contains a plethora of services for hire, ranging from spam attacks such as review bombing or phising emails, to hackers for hire. The regular price for this type of services oscillates between 5 and 10 EUR. **Figure 8.15** depicts this relationship, looking at the listings distribution for prices below 100 EUR. The figure shows that most products can be found below 50 EUR, however, there are multiple of examples of more expensive services. An example of outlier is a service offering followers for a social media platform for 100 EUR.

This can be further seen in **Figure 8.16**, which focuses on prices below 15 EUR to illustrate the curve of the distribution in detail. From the figure, it is possible to see how the DNMs Versus and Kingdom, have a lower median price compared to their counterpart. It is interesting to see how ASAP prices raise above the average. However, looking at the data reveals that DNMs offer similar services at different prices. In particular, Alphabay and Tor2Door include larger variety of hacking services, such as anonymous RDP (Remote Desktop Protocol), hackers for hire and background checks, although prices are similar in most cases with few exceptions. Moreover, the descriptions of *hacker for hire* listings suggest that the product is contact details rather than specific hacks. The dataset contains two similar records of DDoS attack services offered in Kingdom and Versus for 350 EUR, both listings belonging to the same vendor name.

8.2 Vendors

With regard to the vendors, each record was scraped to collect the vendor that offers the product. Each vendor found in the dataset has been crawled to acquire valuable information such as the vendor name, the trust level on the DNM and PGP keys. This information has been used to analyse the relationship between vendors and their products. Throughout this section, we explore the most relevant vendors for

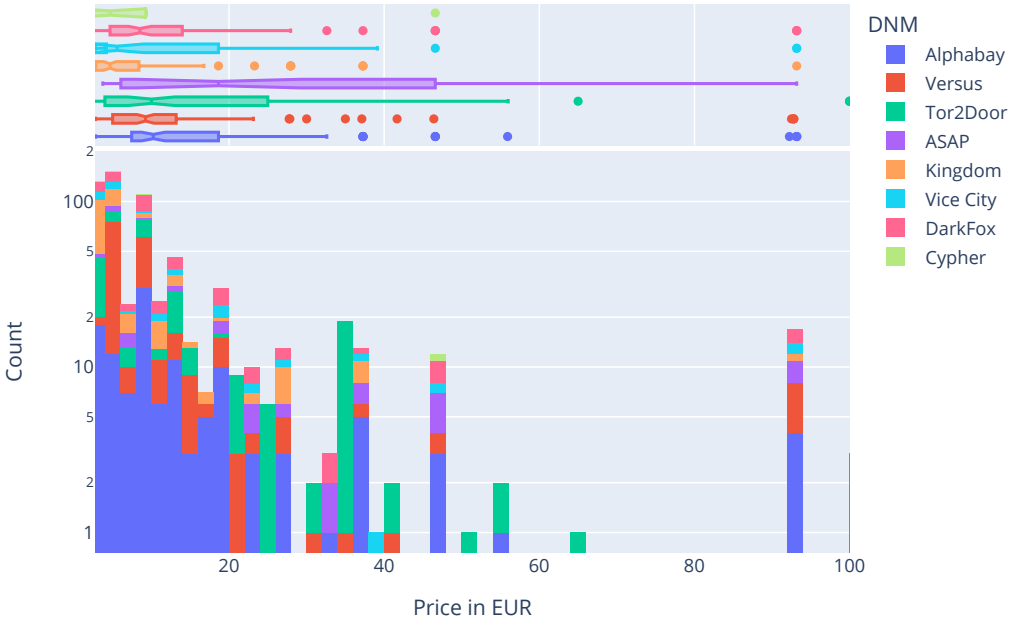

Figure 8.15: Prices histogram for **services** under the 95% confidence interval. Count represented using logarithmic values

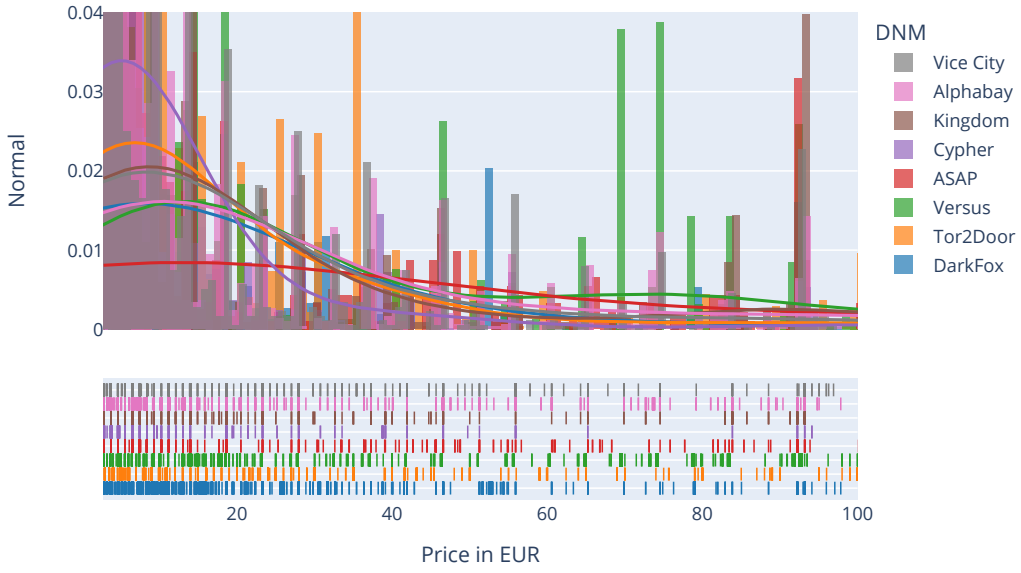

Figure 8.16: Prices distribution for **services** under the 95% confidence interval

each DNM, the products they advertise, and significant findings related to each category of products. The section concludes with a brief study on whether vendors can be found on multiple DNMs simultaneously, using vendor usernames and PGP keys to build the graph.

The results shown in **Table 8.4** represent the number of vendors identified per DNM. Looking at **Figure 8.17**, which represents the distribution of listings per vendor, it can be seen that a small group of approximately 12 vendors advertise more than 300 listings in their catalogue. It is interesting to see that, despite of *fraud* being the largest category and most exploited by vendors, two of the top 10 vendors with the highest number of listings accumulate almost a third of the total *media* listings. Most of these vendors diversify their operations to offer a wide variety of products.

DNM	Vendors Count
Alphabay	270
ASAP	181
Versus	113
Kingdom	70
Tor2Door	171
DarkFox	291
Vice City	67
Cypher	39
Total	1202

Table 8.4: Number of vendors per site

These vendors typically sell their products in multiple DNMs simultaneously. This can be seen in **Figure 8.18**, which provides an augmentation of an arc diagram showing the correlation between vendors and DNMs. Vendors are illustrated as black bubles, while DNMs are coloured in red. The figure focuses on a point where it can be seen that multiple vendors, including *fraudbuddy* (also seen in 8.17 as the vendor with the fourth highest amount of products), appear in three DNMs. The size of the bubbles represent the amount of listings; the opacity of the links, the amount of listings the vendor has on the DNM; and the thickness, the sum of the listing prices.

Furthermore, it is interesting to see that vendors will advertise different products and prices depending on the DNM. **Figure 8.19** illustrates this phenomenon using a sample of vendors with more than 200 listings in his catalogue that can be found on multiple DNMs. Another common factor for these vendors is that they do not tend to diversify, with most of their listings focused on *fraud* or *media*, with few exceptions of vendors that advertise an almost uniform number of listings for each category.

However, vendor usernames appearing in multiple DNMs do not necessarily belong to the same identity. Capturing the PGP keys from the vendor profiles

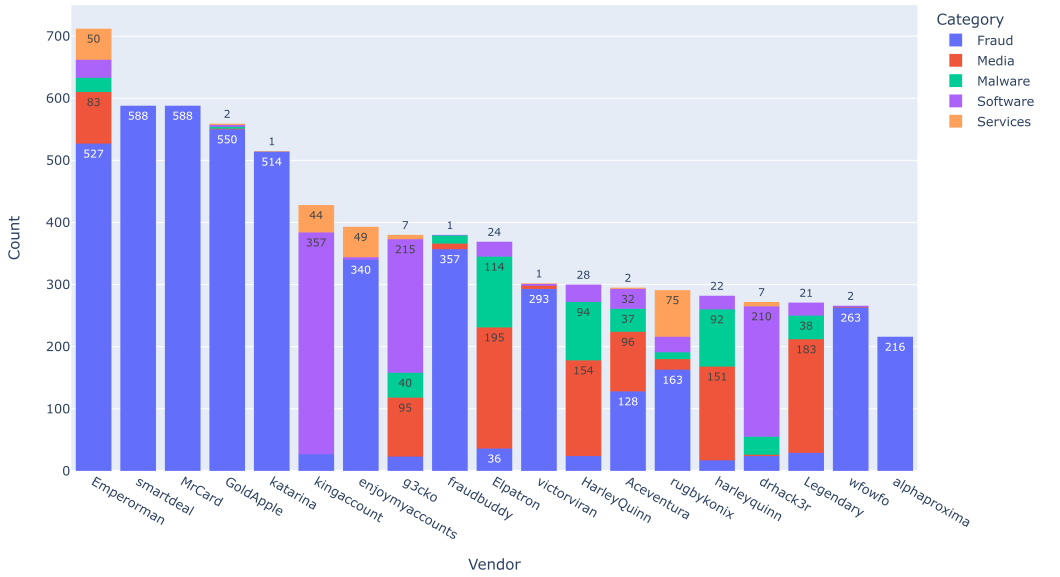

Figure 8.17: 20 most representative vendors showing the number of products they advertise per category

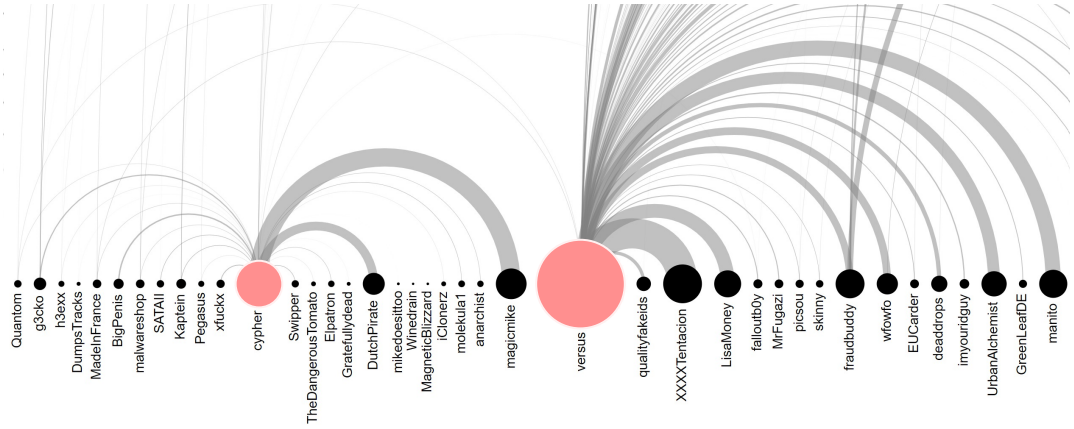

Figure 8.18: Augmentated arc diagram of the vendors in relation to the DNMs

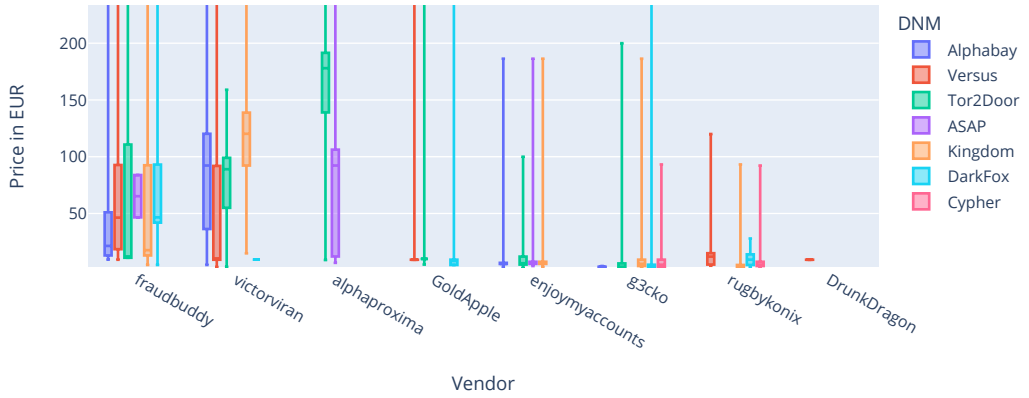

Figure 8.19: Boxplot showing a sample of vendors with more than 200 listings and appearing on multiple DNMs

on each DNM allows us to distinguish between different vendors using similar usernames. However, this theory can only be assumed since the real identity of the vendors is still unknown, i.e., a vendor may be using multiple accounts with different PGP keys. In the other hand, vendors using different username and similar PGP keys are proof of account ownership. **Figure 8.20** show a case of this event, where the vendor chooses to use different usernames for multiple DNMs. In many cases, also using multiple usernames inside the same DNM. The figure shows the relationship between PGP keys, as the outer level of the bubbles; DNMs, the inner circles; and vendor usernames, the coloured circles. Moreover, the size of the bubbles is given by the number of CSRPs related to the PGP key. However, looking at the data we can see that this may be a symptom of the user lacking understanding about PGP keys. For example, the dataset contains multiple observations of private PGP keys, instead of public ones. In other cases, vendors have been identified using more than one PGP key, or offering both public and private keys in the same page. What is striking, is that some of these vendor's reputation is rather high compared to the median.

Regarding reputation, vendors are ranked based on feedback from the users after they purchase a product. The actual classification method is different for each DNM. Therefore, the dataset has been normalised to use values from zero to one, representing the minimum and maximum possible values. This has been done calculating the percentage towards the maximum value. **Figure 8.21** illustrates the relationship between the vendor feedback, the average price of the vendor catalog, and the amount of listings advertised by the vendor.

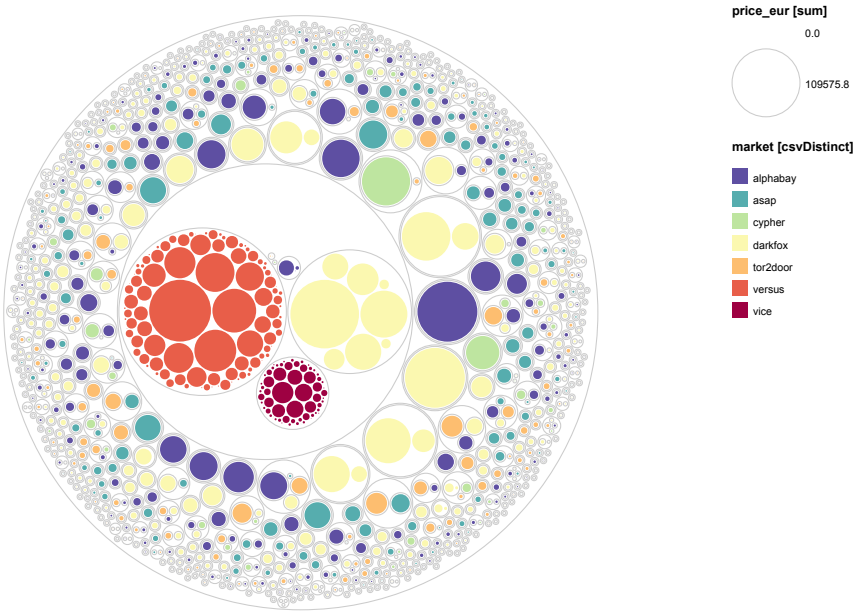

Figure 8.20: Circle packing of PGP keys rims containing DNMs and vendor usernames

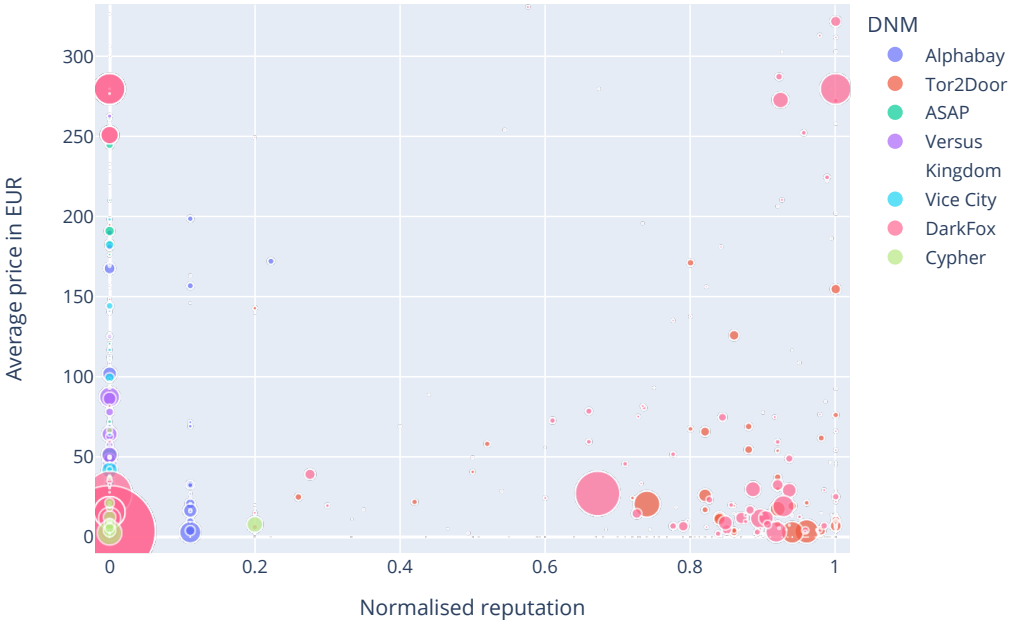

Figure 8.21: Bubble scatterplot of the vendor's reputation, average price of their catalog and amount of listings offered by the vendor

Furthermore, these same vendors will list their products at the lowest prices, as can be seen in 8.22. With regard to vendors selling fewer products, the data suggest that they are not likely to succeed. These vendors are often found offering similar products for higher prices than the competence. Furthermore, these vendors tend to have fewer reviews, resulting in a lower level of trust.

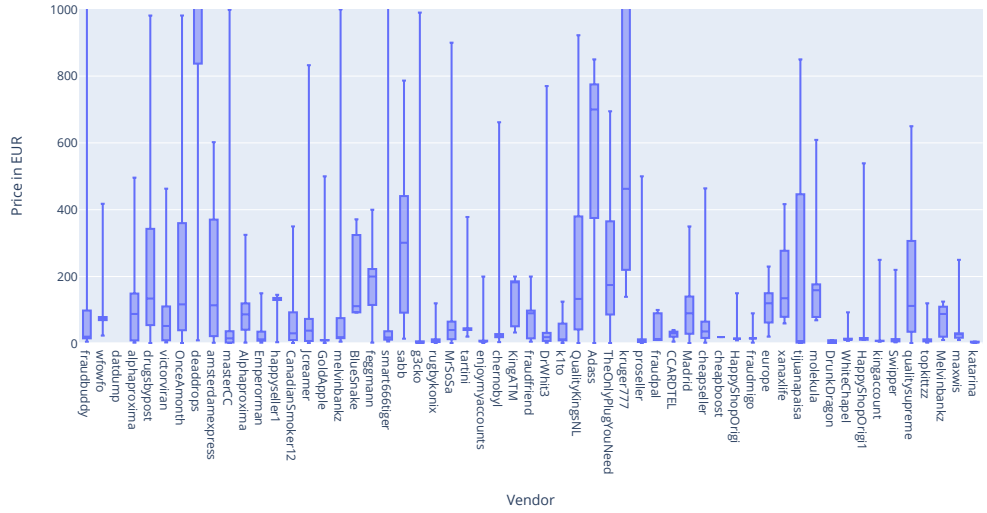

Figure 8.22: Boxplot showing the vendors with the highest variance

8.3 Summary

This chapter aimed to give a comparative analysis of DNMs and the products offered on each market. The analysis includes a detailed view of the listing divided into five categories, estimating their prices through the study of the sample distribution. Moreover, the chapter includes a brief entropy analysis of the listings description. The results of this study suggest that large concentrations of listings in certain category groups can be explained by vendors using similar or identical descriptions for their products. In addition, vendors will create placeholders for their listings using exaggerated prices. **Soska et al. [56]** noticed this phenomenon and created a heuristic to correct these records appearing on their analysis. The heuristic has been applied to the dataset to remove the holding prices. Furthermore, category listings were clustered into subgroups of similar characteristics when possible. Each category includes a brief description of the content of its subgroups.

The results shows that most items will be sold for under 20 EUR. Furthermore, fraudulent information may be the main source of income for vendors dedicated

to cyber security, due to the high volatility of the prices and soaring quantity of offerings, in particular for the carding subgroup. Moreover, the dataset contain traces of dangerous malware available for lower prices than, for example, hacking tutorials. Although the dataset contains a lower volume of malware records, the data suggest that there is a degree of specialisation on CSRPs in smaller DNMs. These DNMs present higher levels of entropy, indicating that vendors in these DNMs use tailored descriptions for their products.

Regarding vendors, the data reveals that most listings belong to a small number of specialised vendors. These vendors have been seen appearing on multiple DNMs simultaneously offering different products and prices on each DNM. Moreover, the dataset contains evidences of vendors using multiple usernames, and instances of similar vendor usernames with different PGP keys. Lastly, the chapter included a brief analysis of the relationship between the amount of listings per vendor and the trust level or average feedback. The rate of vendor feedback projected a positive correlation between the feedback and the number of listings offered. However, this study does not show any relationship between the category of the product and the feedback of the clients. Further studies are necessary to clarify the impact the type of product may have on the reputation of the vendor.

Chapter 9

Discussion

This chapter provides an assessment of the objectives and goals of the thesis. In the following section, we present a brief evaluation summary of the requirements introduced in **Chapter 5**. Then, **Section 9.2** gives an overview of the limitations of the project. Finally, this chapter concludes with a presentation of the future objectives in **Section 9.3**.

9.1 Requirements evaluation

In regards to the project requirements, this section includes an individual evaluation to estimate whether each requirement has been fulfilled. The requirements are organised into implemented and not implemented, which are summarised in **Table 9.1**. The following evaluation consists of a brief description of the implementation of requirements and limitations.

	Functional	Non-Functional
Implemented	FR1, FR2, FR3, FR4	NFR1, NFR2, NFR3, NFR4, NFR5
Not Implemented	FR5, FR6	

Table 9.1: Requirements fulfilment divided into functional and non-functional

Implemented

FR1 Crawl DNMs hosted in TOR and I2P

This requirement was included in **Section 7.1.1**, as part of the enhancements made to the *Session* package. The system is now capable of crawling I2P and TOR DNMs. However, these capabilities can now be seamlessly expanded to include other networks.

FR2 Identify CSRPs using NLP

A NLP model has been developed to label listings based on whether they are related to cyber security. The model presents an accuracy of a 91% on this binary classification task. Furthermore, this BERT model based model has been trained using a set of 2000 manually labelled records.

FR3 Classify CSRPs using NLP

A multi-label BERT classification model has been trained to label records in one of six categories related to cyber security plus one for false positives from the previous model. The model reaches a 77% accuracy with approximately 0.76 training loss. Lastly, the model has been trained using 2000 manually labelled records spread unevenly among the categories, due to the lack of records for some categories.

FR4 Resume-crawling feature

The *crawler* service was enhanced to include this feature by implementing a record of the current and the immediate previous page for each crawled category.

NFR1 Complete crawls from DNM including CSRPs

Every DNM was fully crawled for the relevant categories. The project finalised without pending DNMs.

NFR2 Utilise only English based DNMs

To the best of our knowledge, only one popular DNM used another language than English, called Hydra. However, this DNM was seized at the very beginning of this thesis. The content of that DNM was briefly assessed in the original work for MS.

NFR3 Risk analysis on the listings

Using the classification of listings in **Chapter 8**, the records have been divided into subgroups when possible, to analyse the risk they impose on the general public on the following basis: the likelihood of immediate success of the product after it has been acquired, i.e., how likely is the product to cause any harm; and the impact of the product, i.e., how much damage can be caused as a result of the product being used.

NFR4 Improve the general usability of MS

The application has been further simplified, including validation methods and improving the level of automation of the overall system. For example, the application is now placed in a single container. Furthermore, the system will now handle exceptions with more precision, which will require less attention from the user.

NFR5 Reduce the verbosity of MS

The *Crawler* strategies have been modelled to output fewer messages, reduce user fatigue when using the application, and give a better sense of the current state of the application. Moreover, the level of verbosity can now be changed using environment variables, in case the user requires to see, for example, debugging messages.

Not Implemented

FR5 Real-time labelling and classification using dynamic learning

As part of the scope of this thesis, the models have been implemented in a separate module. Therefore, the current application is now capable of using these models on real-time. Using these models on real-time will introduce throttling to the application, without adding a major value in the current setup.

FR6 Use multiple client nodes for crawling routing

The system does not include methods to distribute the load across multiple tunnels or circuits. I2P DNMs did not impose heavy restrictions on crawling speed. However, TOR DNMs would see this feature as beneficial in the future. A number of TOR DNMs frequently blocked our circuits rather than the account used to crawl. This issue was overcome refreshing the proxy container, perhaps future implementations could consider sending a signal to the proxy when the crawler sees an evidence of the path being blocked.

9.2 Limitations

In terms of the development process, the project had a slow start due to test-driven development and automated testing. This methodology allows us to create reliable code blocks; however, the process is rather long and complicated. Testing certain parts of the application that required connecting to the DNMs. Using tools such as Tox [81] could improve this situation, simplifying the stipends of getting used to the application and enhancing it with a clear vision of the desired workflow. However,

adding new technologies increase the complexity of the system, and therefore the barrier of entry for developers.

Regarding the issues faced during the thesis, the nature of DNMs has complicated the crawling process on several occasions. In spite of the improvements to the MS system, DNMs downtime and structural changes delayed completing many of the DNMs snapshots. For example, DNM *ASAP* changed the CAPTCHA challenge multiple times, and some DNMs experienced downtime for several consecutive days. In addition, we collect vendor information after finishing with DNMs products. This created an unexpected challenge, as some DNMs were now inaccessible. As a result, we were unable to collect vendor information from *Versus*.

Moreover, MS had a technical problem where the system would delete entries before scraping them. This issue was quickly identified and solved; however, approximately 30 records that were not accessible anymore were lost. Additionally, we run out of time before implementing mapping functions and support to manage content divided into multiple pages for *blueprints*. These features target vendors, where DNMs may split profiles into separate pages. A typical example is one where the PGP keys are on a page other than the vendor's description. As result, *ASAP* vendor reputation could not be collected, and *Alphabay* vendor reputation had to be measured using an internal metric called *vendor level*.

To solve this issue, we develop a reputation equation that can be used to evaluate vendors using a common metric, illustrated in **Figure 9.1**. This equation calculates the vendor's reputation considering received negative feedback and disputes in relation to the total amount of user feedback they received. We choose negative feedback since many DNMs allow users to give a neutral feedback. However, rating systems based on stars and others symbols may need further conversion. Since vendors can receive disputes on any product regardless of the feedback, the metric has been included as a weight for the overall reputation. This means that disputes will have a negative effect in the vendor's reputation. The resultant reputation is given as a positive decimal number in a range between zero and one.

$$Reputation = \underbrace{(1 - \frac{Negative}{Total})}_{\text{Positive feedback}} \times \underbrace{(1 - \frac{Disputes}{Total})}_{\text{Weight}}$$

Figure 9.1: Vendor reputation equation

On the question of models M1 and M2, the dataset needed approximately a 6% manually labelled records to issue a reasonable F1-Score. The performance of the models was sufficiently accurate for this project, due to the fact that the distribution of listings is skewed towards fraud products. Therefore, the majority of false positives and negatives could be easily fixed when analysing smaller categories

labelled with the M2 model. If anything, the models helped to reduce the time required to label the data. However, despite of using snapshots from eight different DNMs, the entropy of the category clusters is relatively low compared to the total number of observations. Therefore, the M2 model may struggle to label datasets with more diversity.

Moreover, our study is limited to the category labels we used for the model M2. A model capable of assigning multiple labels using a hierarchical system would greatly improve the granulation of the results. For example, assigning additional labels to products under the malware category, we could properly differentiate between botnets and ransomware products.

Lastly, when we began this research, we were prepared to include differences between I2P and TOR DNMs. However, to the best of our knowledge, there are no dedicated I2P DNMs to this date.

9.3 Future Objectives

So far this chapter has summarised the implementation of our requirements and the limitations of our research. In this section, we conclude with a list of further improvements to the MS system, and a brief discussion about possible future objectives for the project.

As previously seen, the MS system is a promising product for researchers interested in studying DNMs. However, the current state of the system could use further improvements to make MS more accessible. Therefore, we propose the following improvements for MS in no particular order.

- *Simplify the current design*

  The system would benefit from a simpler design, allowing the *crawler* service to send raw documents to the *scraper*, and this one to the *storage*, reducing the information round-trips.

- *Web service*

  The *storage* service should replace the functions of the intended *core* service, by implementing a web service to manage the status of the application in a visual interface. Moreover, using a framework such as Django [82] would make the database state more manageable, including automated migrations when making changes to the database schema.

- *Automated login and bypassing CAPTCHA*

  Although this feature was previously mentioned in the original MS, it is important to remember the importance of this feature, which would reduce to a minimum the level of maintenance required by the system.

- *Smarter budgets*

  Budgets are an important feature of the system. However, their simplicity imposes limitations that can only be overcome by manual manipulation of the configuration through trial and error.

- *Simplify blueprints and plan creation*

  In this iteration, blueprints and plans have received multiple enhancements that added a certain level of complexity necessary to adapt to the new structures of the DNMs. Most of the improvements are aimed to reduce the effort of creating these configuration files, while giving options to identify elements in the HTML pages. However, creating these configuration files is time consuming and do not guarantee that the system will capture the right information when DNMs make changes to their structure.

- *Differentiate pinned items*

  When capturing listings, it would be interesting to differentiate pinned items, i.e., listings that appear advertised at the beginning of each category page. The current implementation captures these items by default; however, being able to study these products could lead to further estimations of vendors and the revenue generated from these listings. To do this, the blueprints could include a new validation object to capture exclusively pinned items. These offers should not be captured by the regular "listings" validator. In addition, the database would need to be enhanced with a new column to represent this value.

In future iterations on MS, developers could enhance MS to create separate snapshots. Doing so, researchers can treat the dataset as a time series to measure changes on the pricing and product availability. For example, researchers could study seasonal products, identifying periods where hackers are the most active. Therefore, we propose the following research questions as an example of future research paths.

- *How can we use MS to identify ongoing hacking campaigns?*
- *How can we use MS to predict cyber attacks?*

In addition, the system could also be used to find correlations between vendors and DNMs. In **Section 8.2**, we briefly analysed vendors and the use of PGP keys. This research could be continued estimate the vendor's revenue and determine the demand of certain products.

As a final note, DNMs include forum sections in where users interact with vendors. These forums contain valuable information that MS is not prepared to

collect. For example, our dataset contains multiple observation of custom listings without a description. In addition, most of these observations are sold for higher prices than the average. Therefore, it would be interesting to identify these products and vendors who use this technique to advertise their catalogue.

Chapter 10

Conclusion

Though many researchers studied the economy of the Darknet using content extracted from forums and DNMs, few writers focused on CSRPs. However, few researchers considered DNMs accessible through I2P. Additionally, most studies of CSRPs have only produced an identification method, without examining the content in much detail.

To synthesise an answer to these problems, this project used the MS system to collect information about CSRPs and vendors from eight different DNMs. Furthermore, the project requirements were included in the system design used to enhance MS and create a labelling system consisting of two NLP models. Finally, these models were used to further classify dataset entries into a set of five clusters, one for each CSRP category. The category labels were named after a generalisation of the categories found in DNMs.

The main contributions of this thesis are summarised as a quantitative analysis of DNM CSRPs. The results of this analysis suggest that the main source of income for CSRP vendors is fraudulent information, such as hacked databases and credit cards. However, this information is inexpensive, with most of the products being sold for less than 20 EUR. Although ransomware and other malicious software can be bought for even cheaper prices. In addition, the majority of these products belong to the catalogue of very few unique vendors that appear in multiple DNMs using the same username and PGP keys. Nonetheless, the results revealed that it is possible to find vendors with multiple accounts within the same DNM, reducing even further the list of vendors capitalising on CSRPs.

In addition, this study included a brief qualitative comparative analysis between the networks I2P and TOR. The results show that while TOR is concerned with accessibility, I2P is focused on flexibility. Regarding I2P, this network is designed to meet the cyber security requirements of advanced users who are aware of their threat model. In addition, I2P is more resilient to cyber attacks than TOR, due to the network architecture. However, the network is not prepared yet to similar levels of traffic thanTOR, due to lookup scalability issues with netDb. Therefore, hidden services use I2P as an alternative network to overcome availability issues.

This thesis revealed that CSRPs such as ransomware are more accessible than conventionally thought. Furthermore, this study revised the reasons for DNMs to offer alternative access methods such as I2P mirrors. However, additional research is required to understand the evolution of DNM CSRPs over time. This knowledge will be useful for predicting future cyber attacks and mitigate hacking campaigns.

Glossary

AAU Aalborg University. 3

BERT Bidirectional Encoder Representations from Transformers. 29, 30, 76

CAPTCHA Completely Automated Public Turing test to tell Computers and Humans Apart. 26, 78

Client A node in the Tor network, typically running on behalf of one user, that routes application connections over a series of relays. 5

CSRP Cyber Security-Related Products. vi, 2, 3, 14, 15, 19, 22, 24, 27, 29, 30, 42, 46, 47, 48, 71, 74, 76, 82, 83, 93

cyber cyber security. 1, 4, 14, 16, 18, 19, 20, 22, 29, 31, 47

Directory Authority A special-purpose Tor relay that maintains a list of currently-running relays and periodically publishes a consensus together with the other directory authorities. 5

DNM Darknet Market. v, 1, 2, 3, 4, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27, 30, 34, 35, 36, 37, 38, 42, 46, 47, 48, 51, 54, 56, 59, 60, 65, 66, 69, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 82, 83, 93

DNMs Darknet Markets. 69

DoS Denial of Service. 4

FR Functional Requirement. 21

I2P Invisible Internet Project. iv, 1, 3, 4, 8, 9, 10, 11, 12, 15, 18, 19, 22, 24, 30, 33, 34, 35, 75, 77, 79, 82, 83

IDP Indexed Pages. iv, 36, 37

LE Law Enforcement. 1

LEA Law Enforcement Agencies. 1, 2

ML Machine learning. 2, 19, 20, 29

MS Midnight Sea. 2, 3, 15, 19, 21, 22, 23, 24, 25, 26, 27, 32, 33, 34, 38, 45, 46, 76, 77, 78, 79, 80, 82

NCP Non-Crawled Products. iv, 36, 37

NFR Non-Functional Requirement. 21

NLP Natural Language Processing. 3, 22, 24, 29, 32, 41, 76, 82

P2P Peer To Peer. 8

PGP Pretty Good Privacy. 1, 66, 69, 71, 74, 78, 80, 82

PII Personally Identifiable Information. 16

Relay A publicly-listed node in the Tor network that forwards traffic on behalf of clients, and that registers itself with the directory authorities. 5

SVM Supervised Vector Machine. 29, 30

TOR The Onion Router. iv, vi, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 15, 18, 22, 24, 25, 30, 33, 34, 35, 75, 77, 79, 83

Bibliography

[1] *EU Drug Markets Report 2019 | www.emcdda.europa.eu*. https://www.emcdda.europa.eu/publications/joint-publications/eu-drug-markets-report-2019. (Accessed on 03/21/2022).

[2] Mohammadreza Ebrahimi, Jay F. Nunamaker Jr., and Hsinchun Chen. "Semi-Supervised Cyber Threat Identification in Dark Net Markets: A Transductive and Deep Learning Approach". In: *J. Manag. Inf. Syst.* 37.3 (2020), pp. 694–722. URL: http://www.jmis-web.org/articles/1491.

[3] Kathleen A Wallace. "Anonymity". In: *Ethics and Information technology* 1.1 (1999), pp. 21–31.

[4] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. "Tor: The Second-Generation Onion Router". In: *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*. Ed. by Matt Blaze. USENIX, 2004, pp. 303–320. URL: http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html.

[5] *Glossary | Tor Project | Support*. https://support.torproject.org/glossary/. (Accessed on 03/24/2022).

[6] *When I'm using Tor, can eavesdroppers still see the information I share with websites, like login information and things I type into forms? | Tor Project | Support*. https://support.torproject.org/https/https-1/. (Accessed on 03/25/2022).

[7] Bassam Zantout and Ramzi Haraty. "I2P Data Communication System". In: (Apr. 2002).

[8] Bernd Conrad and Fatemeh Shirazi. "A Survey on Tor and I2P". In: July 2014.

[9] *Tunnel Implementation - I2P*. https://geti2p.net/en/docs/tunnels/implementation. (Accessed on 03/26/2022).

[10] Pekka Abrahamsson et al. "Agile software development methods: Review and analysis". In: *arXiv preprint arXiv:1709.08439* (2017).

[11] James E Kelley Jr and Morgan R Walker. "Critical-path planning and scheduling". In: *Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference*. 1959, pp. 160–173.

[12] Ken Schwaber and Jeff Sutherland. "The scrum guide". In: *Scrum Alliance* 21.1 (2011).

[13] *Obsidian*. https://obsidian.md/. (Accessed on 05/21/2022).

[14] James M. Wilson. "Gantt charts: A centenary appreciation". In: *European Journal of Operational Research* 149.2 (2003). Sequencing and Scheduling, pp. 430–437. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(02)00769-5. URL: https://www.sciencedirect.com/science/article/pii/S0377221702007695.

[15] *GitHub*. https://github.com/. (Accessed on 05/21/2022).

[16] *Home - Docker*. https://www.docker.com/. (Accessed on 05/21/2022).

[17] *IEEE Xplore*. https://ieeexplore.ieee.org/Xplore/home.jsp. (Accessed on 02/07/2022).

[18] *ScienceDirect.com | Science, health and medical journals, full text articles and books.* https://www.sciencedirect.com/. (Accessed on 02/01/2022).

[19] *ACM Journal*. https://dl.acm.org/journals. (Accessed on 02/01/2022).

[20] Judit Bar-Ilan. "Data collection methods on the Web for infometric purposes — A review and analysis". In: *Scientometrics* 50.1 (Jan. 2001), pp. 7–32. ISSN: 1588-2861. DOI: 10.1023/A:1005682102768.

[21] Sergio Pastrana et al. "CrimeBB: Enabling Cybercrime Research on Underground Forums at Scale". In: Apr. 2018, pp. 1845–1854. ISBN: 978-1-4503-5639-8. DOI: 10.1145/3178876.3186178.

[22] Bassel Alkhatib and Randa Basheer. "Crawling the Dark Web: A Conceptual Perspective, Challenges and Implementation". In: *Journal of Digital Information Management* 17 (Apr. 2019), p. 51. DOI: 10.6025/jdim/2019/17/2/51-60.

[23] Gengqian Zhou et al. "A Market in Dream: the Rapid Development of Anonymous Cybercrime". In: *Mobile Networks and Applications* 25.1 (Feb. 2020), pp. 259–270. ISSN: 1572-8153. DOI: 10.1007/s11036-019-01440-2.

[24] Michele Campobasso, Pavlo Burda, and Luca Allodi. "CARONTE: Crawling Adversarial Resources Over Non-Trusted, High-Profile Environments". In: *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. June 2019, pp. 433–442. DOI: 10.1109/EuroSPW.2019.00055.

[25] Kent Beck. *Test-driven development: by example*. Addison-Wesley Professional, 2003.

[26] Matthew Ball and Roderic Broadhurst. *Data Capture and Analysis of Darknet Markets*. 3344936. Feb. 2021. DOI: 10.2139/ssrn.3344936. URL: https://papers.ssrn.com/abstract=3344936.

[27] Shelby Davis and Bruce Arrigo. "The Dark Web and anonymizing technologies: legal pitfalls, ethical prospects, and policy directions from radical criminology". In: *Crime, Law and Social Change* 76.4 (Nov. 2021), pp. 367–386. ISSN: 1573-0751. DOI: 10.1007/s10611-021-09972-z.

[28] Russell Brewer et al. "The Ethics of Web Crawling and Web Scraping in Cybercrime Research: Navigating Issues of Consent, Privacy, and Other Potential Harms Associated with Automated Data Collection". In: *Researching Cybercrimes: Methodologies, Ethics, and Critical Approaches*. Ed. by Anita Lavorgna and Thomas J. Holt. Springer International Publishing, 2021, pp. 435–456. ISBN: 978-3-030-74837-1. DOI: 10.1007/978-3-030-74837-1_22. URL: https://doi.org/10.1007/978-3-030-74837-1_22.

[29] James Martin and Nicolas Christin. "Ethics in cryptomarket research". In: *International Journal of Drug Policy*. Drug Cryptomarkets 35 (Sept. 2016), pp. 84–91. ISSN: 0955-3959. DOI: 10.1016/j.drugpo.2016.05.006.

[30] Yue Gao et al. "Large-scale discovery and empirical analysis for I2P eepSites". In: *2017 IEEE Symposium on Computers and Communications (ISCC)*. July 2017, pp. 444–449. DOI: 10.1109/ISCC.2017.8024569.

[31] Peipeng Liu et al. "Empirical measurement and analysis of I2P routers". In: *Journal of Networks* 9.9 (2014), p. 2269.

[32] Nguyen Phong Hoang et al. "An empirical study of the I2P anonymity network and its censorship resistance". In: *arXiv e-prints* (Sept. 2018). arXiv: 1809.09086 [cs.NI] tex.eid: arXiv:1809.09086 tex.priority: prio1 tex.readstatus: skimmed, arXiv:1809.09086.

[33] Christos Iliou et al. "Hybrid Focused Crawling for Homemade Explosives Discovery on Surface and Dark Web". In: *2016 11th International Conference on Availability, Reliability and Security (ARES)*. Aug. 2016, pp. 229–234. DOI: 10.1109/ARES.2016.66.

[34] Roberto Magán-Carrión et al. "Unveiling the I2P web structure: a connectivity analysis". In: *arXiv e-prints* (Jan. 2021). arXiv: 2101.03212 [cs.CR] tex.eid: arXiv:2101.03212, arXiv:2101.03212.

[35] Yara Fareed Fahmy Bayoumy, Per Håkon Meland, and Guttorm Sindre. "A Netnographic Study on the Dark Net Ecosystem for Ransomware". In: *2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*. June 2018, pp. 1–8. DOI: 10.1109/CyberSA.2018.8551424.

[36] Alberto Bracci et al. "Dark Web Marketplaces and COVID-19: before the vaccine". en. In: *EPJ Data Science* 10.11 (Dec. 2021), p. 6. ISSN: 2193-1127. DOI: 10.1140/epjds/s13688-021-00259-w.

[37] Victor Benjamin et al. "Exploring threats and vulnerabilities in hacker web: Forums, IRC and carding shops". In: *2015 IEEE International Conference on Intelligence and Security Informatics (ISI)*. May 2015, pp. 85–90. DOI: 10.1109/ISI.2015.7165944.

[38] Tim M. Booij et al. "Get Rich or Keep Tryin' Trajectories in dark net market vendor careers". In: *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. Sept. 2021, pp. 202–212. DOI: 10.1109/EuroSPW54576.2021.00028.

[39] Rolf van Wegberg et al. "Plug and Prey? Measuring the Commoditization of Cybercrime via Online Anonymous Markets". In: 2018, pp. 1009–1026. ISBN: 978-1-939133-04-5. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/van-wegberg.

[40] Ashok Deb, Kristina Lerman, and Emilio Ferrara. "Predicting Cyber Events by Leveraging Hacker Sentiment". In: *Information* 9.11 (Nov. 2018). arXiv: 1804.05276, p. 280. ISSN: 2078-2489. DOI: 10.3390/info9110280.

[41] George Pantelis et al. "On Strengthening SMEs and MEs Threat Intelligence and Awareness by Identifying Data Breaches, Stolen Credentials and Illegal Activities on the Dark Web". In: *The 16th International Conference on Availability, Reliability and Security*. ARES 2021. New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 1–7. ISBN: 978-1-4503-9051-4. DOI: 10.1145/3465481.3469201. URL: http://doi.org/10.1145/3465481.3469201.

[42] Nolan Arnold et al. "Dark-Net Ecosystem Cyber-Threat Intelligence (CTI) Tool". In: *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*. July 2019, pp. 92–97. DOI: 10.1109/ISI.2019.8823501.

[43] Mitch Macdonald et al. "Identifying Digital Threats in a Hacker Web Forum". In: *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. ASONAM '15. Association for Computing Machinery, Aug. 2015, pp. 926–933. ISBN: 978-1-4503-3854-7. DOI: 10.1145/2808797.2808878. URL: https://doi.org/10.1145/2808797.2808878.

[44] Eric Nunes et al. "Darknet and deepnet mining for proactive cybersecurity threat intelligence". In: *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*. Sept. 2016, pp. 7–12. DOI: 10.1109/ISI.2016.7745435.

[45] Mohammadreza Ebrahimi et al. "Detecting cyber threats in non-english dark net markets: A cross-lingual transfer learning approach". In: *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*. Nov. 2018, pp. 85–90. DOI: 10.1109/ISI.2018.8587404.

[46] Mohammadreza Ebrahimi, Jay F. Nunamaker Jr., and Hsinchun Chen. "Semi-supervised cyber threat identification in dark net markets: A transductive and deep learning approach". In: 37.3 (2020). tex.bibsource: dblp computer science bibliography, https://dblp.org tex.biburl: https://dblp.org/rec/journals/jmis/EbrahimiNC20.bi pp. 694–722.

[47] Nazgol Tavabi et al. "DarkEmbed: Exploit Prediction With Neural Language Models". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.11 (Apr. 2018). ISSN: 2374-3468. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11428.

[48] Martin Glinz. "On Non-Functional Requirements". In: *15th IEEE International Requirements Engineering Conference (RE 2007)*. Oct. 2007, pp. 21–26. DOI: 10.1109/RE.2007.45.

[49] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv:1810.04805. arXiv:1810.04805 [cs] version: 2 type: article. May 2019. DOI: 10.48550/arXiv.1810.04805. URL: http://arxiv.org/abs/1810.04805.

[50] *SuperGLUE Benchmark*. https://super.gluebenchmark.com/leaderboard. (Accessed on 04/13/2022).

[51] Barret Zoph et al. "Designing Effective Sparse Expert Models". In: *arXiv:2202.08906 [cs]* (Feb. 2022). arXiv: 2202.08906. URL: http://arxiv.org/abs/2202.08906.

[52] Zirui Wang et al. "Towards zero-label language learning". In: *arXiv preprint arXiv:2109.09193* (2021).

[53] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *arXiv:1910.10683 [cs, stat]* (July 2020). arXiv: 1910.10683. URL: http://arxiv.org/abs/1910.10683.

[54] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *arXiv:1907.11692 [cs]* (July 2019). arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

[55] Pengcheng He, Jianfeng Gao, and Weizhu Chen. "DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing". In: *arXiv:2111.09543 [cs]* (Dec. 2021). arXiv: 2111.09543. URL: http://arxiv.org/abs/2111.09543.

[56] Kyle Soska and Nicolas Christin. "Measuring the Longitudinal Evolution of the Online Anonymous Marketplace Ecosystem". en. In: 2015, pp. 33–48. ISBN: 978-1-939133-11-3. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/soska.

[57] *pgAdmin - PostgreSQL Tools*. https://www.pgadmin.org/. (Accessed on 04/28/2022).

[58] *Hugging Face – The AI community building the future.* https://huggingface.co/. (Accessed on 05/02/2022).

[59] *Simple Transformers*. https://simpletransformers.ai/. (Accessed on 05/03/2022).

[60] *distilbert-base-uncased-finetuned-sst-2-english · Hugging Face*. https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english. (Accessed on 05/07/2022).

[61] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[62] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692 (2019). arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

[63] Subhabrata Mukherjee, Ahmed Hassan Awadallah, and Jianfeng Gao. *XtremeDistilTransformers: Task Transfer for Task-agnostic Distillation*. 2021. arXiv: 2106.04563 [cs.CL].

[64] Wenhui Wang et al. *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*. 2020. arXiv: 2002.10957 [cs.CL].

[65] *Project Jupyter | Home*. https://jupyter.org/. (Accessed on 05/03/2022).

[66] Xin Chen et al. "Shared information and program plagiarism detection". In: *IEEE Transactions on Information Theory* 50.7 (July 2004), pp. 1545–1551. ISSN: 1557-9654. DOI: 10.1109/TIT.2004.830793.

[67] *Betal, overfør og tag imod penge på nettet – PayPal Danmark*. https://www.paypal.com/dk/webapps/mpp/home?kid=p58218219526&gclsrc=aw.ds&gclid=Cj0KCQjwmuiTBhDoARIsAPiv6L9SsSwSMRj-HefosPUZFvIkNkNFAeK0qrAVR2PE4urcka2a7q0qfg0aAwcB. (Accessed on 05/10/2022).

[68] *Amazon.com. Gasta menos. Sonríe más.* https://www.amazon.com/. (Accessed on 05/10/2022).

[69] *Apple*. https://www.apple.com/. (Accessed on 05/10/2022).

[70] *Amazon.com: Tarjetas Regalos*. https://www.amazon.com/giftcardscams/b?ie=UTF8&node=15435487011. (Accessed on 05/10/2022).

[71] *Crypto Currency Miner Exploits Hospital - Catch of the Day | Netsurion*. https://www.netsurion.com/catches/crypto-currency-miner-exploits-hospital. (Accessed on 05/13/2022).

[72] *Home | Monero - secure, private, untraceable*. https://www.getmonero.org/. (Accessed on 05/13/2022).

[73] *Bitcoin – P2P-penge med åben kildekode*. https://bitcoin.org/da/. (Accessed on 05/13/2022).

[74] *Privacy-protecting digital currency | Zcash*. https://z.cash/. (Accessed on 05/13/2022).

[75] Dan Jiang and Kazumasa Omote. "An Approach to Detect Remote Access Trojan in the Early Stage of Communication". In: *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*. Mar. 2015, pp. 706–713. DOI: 10.1109/AINA.2015.257.

[76] *Malware Trends Tracker | ANY.RUN*. https://any.run/malware-trends/. (Accessed on 05/13/2022).

[77] Ronny Richardson and Max North. "Ransomware: Evolution, Mitigation and Prevention". In: *International Management Review* 13.1 (Jan. 2017), pp. 10–21.

[78] *Adobe: Creative, marketing and document management solutions*. https://www.adobe.com/. (Accessed on 05/14/2022).

[79] *Microsoft – cloud, computere, apps og spil*. https://www.microsoft.com/da-dk/. (Accessed on 05/14/2022).

[80] *Bureau van Dijk | Private company information – Orbis*. https://www.bvdinfo.com/en-gb/. (Accessed on 05/15/2022).

[81] *Welcome to the tox automation project — tox 3.25.1.dev5 documentation*. https://tox.wiki/en/latest/. (Accessed on 05/18/2022).

[82] *The web framework for perfectionists with deadlines | Django*. https://www.djangoproject.com/. (Accessed on 05/20/2022).

Appendix A

Analysis

Table A.1 contains a summary of the dataset before applying the H6 filter.

DNM	Fraud	Malware	Media	Services	Software	Positives	Total Products
ASAP	773	63	106	38	103	1083	**2834**
Alphabay	2472	185	495	151	213	3516	**5406**
Cypher	414	182	347	7	157	1107	**1790**
DarkFox	6809	375	736	138	846	8904	**13436**
Kingdom	750	134	233	144	251	1512	**2685**
Tor2Door	1821	209	448	128	387	2993	**4679**
Versus	1881	79	187	159	292	2598	**5367**
Vice City	1210	263	593	69	214	2349	**3684**
Grand Total	**16130**	**1490**	**3145**	**834**	**2463**	**24062**	**39881**

Table A.1: Total amount of listings per market, positive listings, and positive listings per category before applying H6

Figure A.1 and **Figure A.2** show boxplots of the prices per DNM with outliers after applying the H6 filter. The second is further divided by CSRP category.

Table A.2 contains the list of RAT kits found under the malware category label. The table includes the known name of the malware and the affected systems.

Table A.3 includes a summary of the botnet software found in the dataset. The table includes the operative system affected by the software, and when the botnet was first reported.

Table A.4 summarises the ransomware found in the dataset. The table includes the operative system affected by the ransomware, and the date when was first reported.

RAT Name	Systems Affected	First seen
Ares	Windows, Linux	2016
AutoRAT	Android	2014
BetterBackdoor	Windows	2019
BlackShades	Windows	2014
Dendroid	Android	2014
DroidJack	Android	2015
njRat	Windows	2013
ORCUS	Windows	2016
RedLine	Windows	2020
Venom	Windows	-
Warzone	Windows	2018

Table A.2: List of RAT kits found in the dataset

Botnet Name	Systems Affected	First seen
BlackNET	Windows	2020
LokiBot	Windows	2017
Oreo QBOT	Android	2016

Table A.3: List of Botnets found in the dataset

Ransomware Name	Systems Affected	First seen
Andr0id	Android	-
BasicLocker	Windows	2020
Chaos (Ryuk)	Windows	2021
CryptoLocker	Windows	2013
EDA2	Windows	2017
Gemcutter	Windows	2015
GrandCrab	Windows	2018
HiddenTear	Windows	2016
Jigsaw	Windows	2016
MyLittleRansomware	Windows	2016
NxRansomware	Windows	2017
Petya	Windows	2017
Shark	Windows	2021
Stampedo	Windows	2016
WannaCry	Windows	2017

Table A.4: List of Ransomware found in the dataset

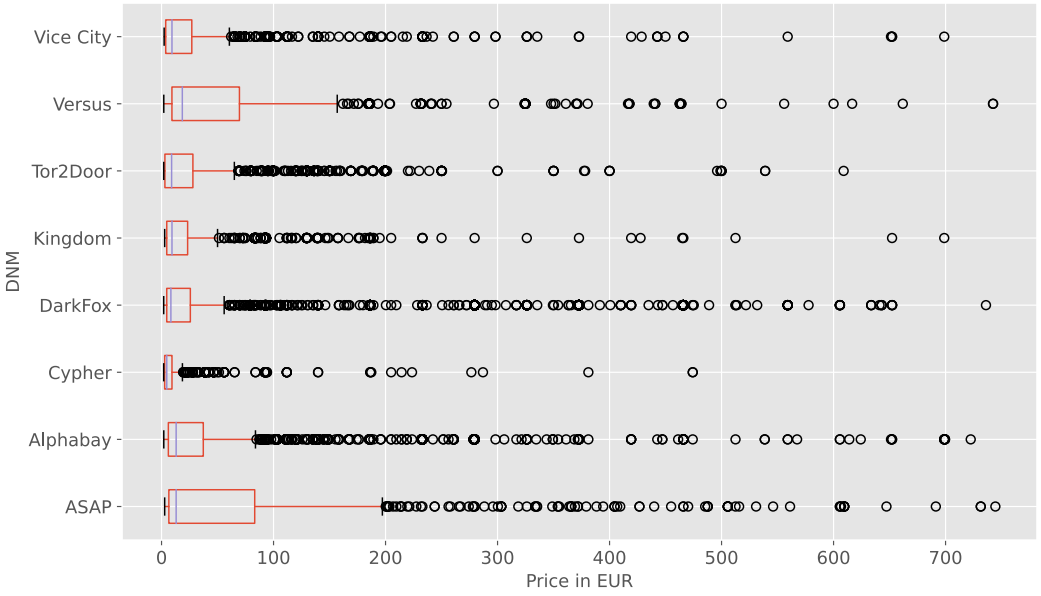

Figure A.1: Price boxplot grouped by DNM without outliers

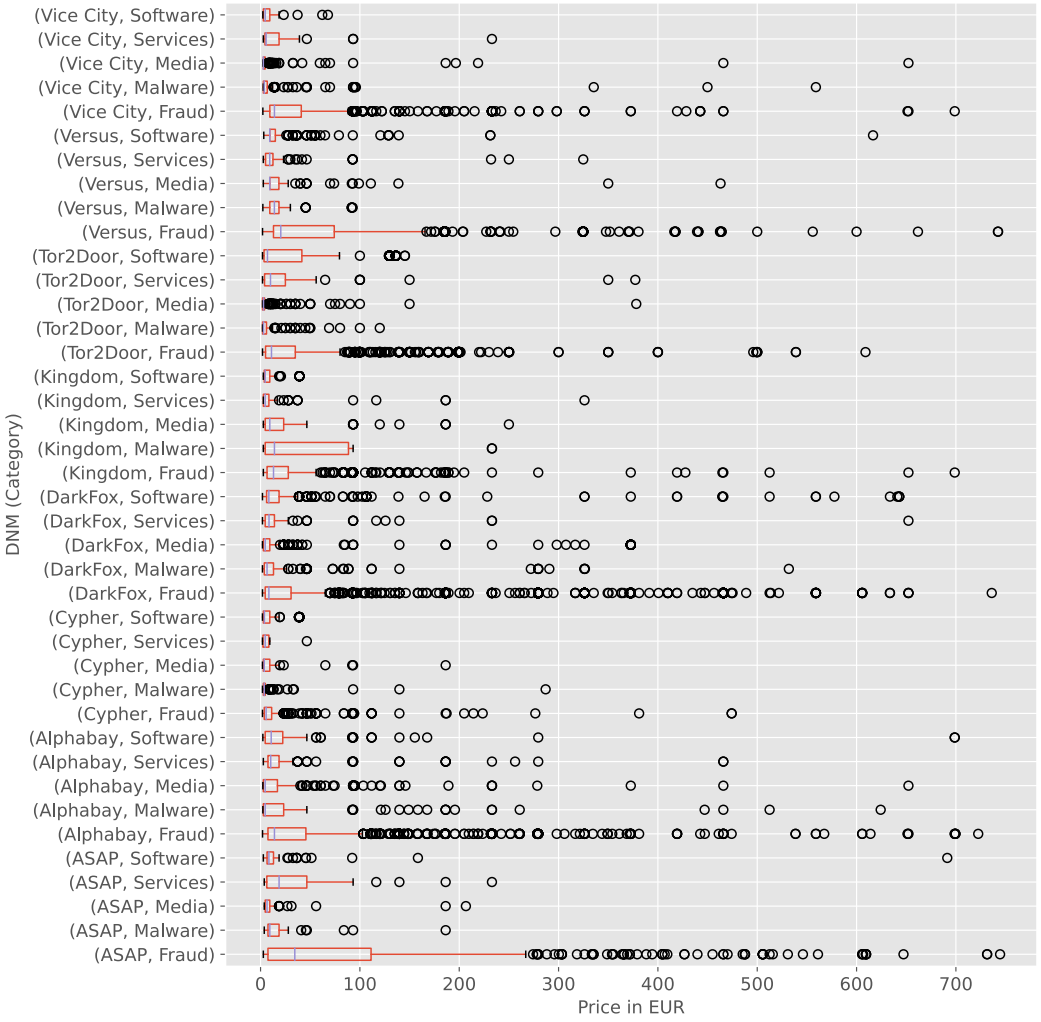

Figure A.2: Price boxplot grouped by DNM and category without outliers

Appendix B

Crawling Experiments

DNM	Max. Delay (in seconds)	Max. Connections
Versus	5	5
ASAP	5	5
Alphabay	5	10
DarkFox	10	3
Tor2Door	5	5
Vice City	10	10
Kingdom	5	10
Cypher	10	3

Table B.1: Crawler stable setup for the different DNMs

Appendix C

Risk Assessment

Table C.1 includes a brief risk analysis of each category subgroup. The metrics utilised were success likelihood, as the probability of being useful immediately after bought; and impact, as the potential damage the product could cause.

Category	Group	Success Likelihood	Impact
Fraud	Carding	Low	Medium
	Hacked Databases	High	High
	Miscellaneous Accounts	Low	Low
Malware	Botnets	Low	Medium
	Crypto-Miners	Medium	High
	Ransomware	High	High
Media	Background and PII Checks	Medium	High
	Carding Methods	Medium	High
	Hacking Tutorials	Low	Low
Service	Various	High	High
Software	Hacking Tools	Low	Medium
	Hacked Software	High	High

Table C.1: Risk assessment of the categories found in the dataset