FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

HET VERZAMELEN VAN CYBERTHREATS IN HET DARKWEB

Simon Backx

Promotor: prof. dr. ir. Bjorn De Sutter
Begeleiders: dhr. Matthias Stala (Ernst & Young Advisory Services)
dhr. Dieter Vandenbroeck (Ernst & Young Advisory Services)

Masterproef ingediend tot het behalen van de academische graad van:
Master of Science in de industriële wetenschappen: informatica

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. Rik Van de Walle
Academiejaar 2016-2017

UNIVERSITEIT
GENT

Voorwoord

Bij de keuze van mijn masterproefonderwerp sprong het onderwerp in samenwerking met Ernst & Young Advisory Services Diegem meteen in het oog. Hoewel het darkweb en threat intelligence toen nog nobele onbekenden waren, sprak het me sterk aan. Ik heb er dan ook enorm veel uit geleerd.

Alvorens u los te laten in de wondere wereld van het darkweb wil ik graag nog enkele personen die deze masterproef mogelijk maakten, bedanken. Allereerst mijn promotor, prof. Bjorn De Sutter voor de waardevolle feedback en de begeleiding.

Ik dank mijn begeleiders Matthias Stala en Frederik Moonen van EY voor het gidsen van de masterproef in de juiste richting. Daarnaast wil ik in het bijzonder het bedrijf EY Advisory Services bedanken om mij de mogelijkheid te hebben gegeven dit boeiende onderwerp te onderzoeken.

Ik dank Michel Raes – de informatieveiligheidsconsulent van de UGent – voor zijn opmerkingen op de risicoanalyse, om mij in contact te brengen met een jurist, en voor de tijd die hij daar in stak.

Tot slot wil ik mijn ouders, broers en vriend bedanken, voor hun steun en helpende hand tijdens niet alleen mijn masterproef maar ook tijdens de studies die eraan vooraf gingen.

Simon Backx, juni 2017

Toelating tot bruikleen

"De auteur geeft de toelating deze masterproef voor consultatie beschikbaar te stellen en delen van de masterproef te kopiëren voor persoonlijk gebruik.
Elk ander gebruik valt onder de bepalingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit deze masterproef."

"The author gives permission to make this master dissertation available for consultation and to copy parts of this master dissertation for personal use.
In the case of any other use, the copyright terms have to be respected, in particular with regard to the obligation to state expressly the source when quoting results from this master dissertation."

Juni 2017

Abstract

Het verhinderen van cybercrime wordt steeds belangrijker voor bedrijven. Via *cyber threat intelligence* wordt hier informatie over verzameld. Deze masterproef onderzocht het darkweb als een mogelijke informatiebron. Via de ontwikkeling van een crawler werd het darkweb zoekbaar gemaakt voor bedrijven. Deze crawler werd vervolgens gedurende een week uitgetest. Daarbij kon worden vastgesteld dat er in een darkweb heel veel persoonsgegevens zoals wachtwoorden en creditcardgegevens te vinden zijn. Die gegevens maakten dikwijls deel uit van korte gratis uittreksels uit grotere te betalen lijsten. Deze uittreksels worden gedeeld om de betrouwbaarheid van hackers te promoten zodat zij hun volledige lijsten kunnen verkopen. De uittreksels zijn van mindere kwaliteit maar geven inzicht in welke bedrijven hier precies door getroffen worden. Dit zijn voornamelijk providers (o.a. Gmail, Telenet), banken, overheidsdiensten (o.a. FBI, Nederlands Ministerie van Algemene Zaken) en internetdiensten (Netflix, Steam...). Al deze organisaties of bedrijven worden besproken in de masterproef. De ontwikkelde applicatie kan de basis vormen van een waardevolle bron voor *cyber threat intelligence*.

Kernwoorden: *darkweb, Tor, cyberthreats, cybercrime, crawler*

Cyber threat collection in the dark web

Simon Backx

Supervisor(s): Bjorn De Sutter, Dieter Vandenbroeck, Matthias Stala

Abstract— **Preventing cyber crimes becomes more important for businesses. Information about this is collected through *cyber threat intelligence*. This master's thesis investigated the dark web as a possible source of information. The dark web was made searchable for businesses through the development of a crawler. Afterwards, the crawler was tested for one week, in which the developed application found a lot of personal information such as passwords and credit cards across the dark web. Most of this information is outdated because it is easily accessible by everyone and is already acted upon as a result. However, it provides insight in the companies that need to deal with these leaks: providers (e.g. Gmail, Telenet), banks, governments (e.g. FBI, Dutch Ministry of General Affairs) and Internet companies (e.g. Netflix, Steam).**

Keywords—**dark web, Tor, cyber threats, cyber crime, crawler**

I. Introduction

The number of cyber crimes continues to increase every year. The total cost of cyber attacks was calculated to be $ 450 billion in 2016 [1]. More businesses digitize their products and services, resulting in more opportunities for hackers. Businesses need to gain insight in known attack strategies, future targets, and security vulnerabilities in order to protect their systems. Most insights are shared by businesses, governments, and security researchers. Usually these insights are not public [2]

An alternative source was investigated: the dark web. The dark web allows hackers to communicate anonymously. Which makes it perfect to sell stolen data or discovered security breaches. We want to provide insights that are useful for a predefined group of companies and applications while discarding all unrelated threats.

In this abstract, we will discuss the development of a tool that allows businesses to search the dark web based on regular expressions and word matching. We will also examine the risks involved. The tool was tested on a range of possible applications for one week. These results will also be looked into.

II. Risk analysis

To be able to search the dark web, one needs to develop a crawler that visits pages and recursively follows the links on those pages. The dark web is for the most part a criminal environment, which the crawler should be adapted to. We will highlight three risks of using a crawler on the dark web and place them in a risk matrix (Figure 1). However, regular risks that normal crawlers should attend (e.g. malware infection and legal issues) will not get addressed. We will assume the crawler only operates on the Tor dark web.

A. Loss of invisibility

Loss of invisibility means that a hidden service (a website in the Tor dark web) can discover the existence of a crawler, not knowing it's purpose or identity.

The activity of a crawler is detectable if it exhibits unusual behaviour. Unusual behaviour is when the requests of the crawler will stand out in the log files of any server. Some examples:

1. Sending too many requests in a short amount of time.
2. Not downloading resource files (styles, images, JavaScript...).
3. Sending HTTP headers different from those of the Tor browser.
4. Automatically submitting forms with invalid information (e.g. login or CAPTCHA forms).
5. Getting caught by a honeypot (a trap created by the hidden service owner).

A crawler that is detected does not pose a threat to the researchers because the hidden service owner does not know the objectives and owners of the crawler. However, the hidden service can defend it against such crawlers by adding CAPTCHAs or requiring authentication.

B. Server identification

Server identification occurs when a hidden service is able to link the activity of a crawler to the IP address of its server.

To explain the Tor dark web briefly and simplified: traffic in Tor is sent over nodes (the circuit) – which are running by volunteers – before it arrives at the destination [3] . This ensures the source and destination remain anonymous. The first node in this circuit is called the entry node or entry guard and is the only node that knows the IP address of the source. The last node in this circuit is called the exit node, which is the only node (except from the source) that knows the IP address of the destination.

There are techniques that allow attackers to bypass the anonymity of Tor by correlating network packets at the entry and exit node in a Tor circuit. These techniques are called confirmation attacks and require the attacker to control the entry and exit node in the same circuit. We differentiate between two types of attacks: passive confirmation attacks and active confirmation attacks.

Passive confirmation attacks do not modify packets. By using a circuit-linking algorithm they are able to link packets at the entry node with the corresponding packets at the exit node, and

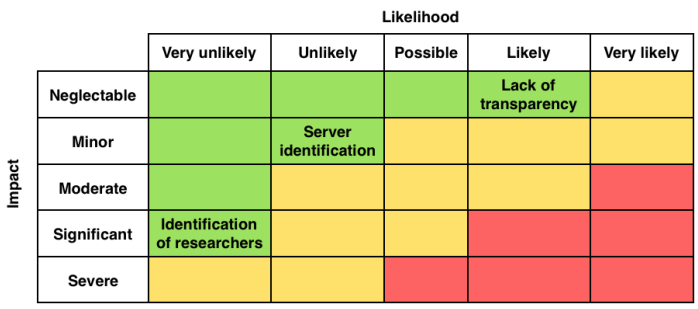


Fig. 1. Risk matrix

subsequently the IP addresses of the two communicating computers [4].

A tagging attack is an active confirmation attack which modifies the packets at one side and detects the changes at the other side of the Tor circuit [5].

The problem with these attacks is that they can not be prevented by the crawler. Müller describes some techniques to fix the attacks, but only the Tor application can implement these protocol changes [4].

To increase anonymity and protect against the above mentioned attacks, one can connect over a Virtual Private Network (VPN) with Tor.

C. Identification of researchers

The identification of researchers is the risk with the most severe consequences. Once a hidden service is able to identify the server of the crawler he can try to get more details about the owners of the server. The most obvious way to do this is by hacking into the server and checking the log files.

Given the IP-address of the server a hacker can also identify the hosting provider and try to break into their servers. If the public IP address is not shared between multiple instances then a hacker can find the corresponding customer for a given server. This is more difficult (if not impossible) if a server does not have a public static IP address.

A server should always be properly secured, this is not different for a dark web crawler. One should use a strong firewall and update software regularly. Best practices should be applied. When possible do not assign a static public IP address to the server.

III. Long term queries

While collecting cyber threats on the dark web, multiple search queries need to run for a long time. E.g.: to search for a certain email address. These queries do not change often and are not very numerous. Indexing every downloaded page – which requires a lot of resources – does not make sense in this scenario because the crawler can execute them when new pages are downloaded. Queries are thus saved and performed on every new page the crawler downloads. The results are saved in a database.

A. Advantages

This method has some advantages:

1. Not generating page indexes saves disk space and reduces costs.
2. The implementation is less complicated.
3. Queries can be more advanced with the ability to combine text search, regular expressions, sentiment analysis, and text classification.
4. New results can trigger notifications to the user.

B. Implementation

To adjust the crawler to this method, it is important that the crawler revisits the main pages of a hidden service in order to detect new pages fast. Eventually the crawler needs to revisit every page. However, new pages should have priority over pages that are already visited.

To accomplish this, the crawler uses multiple queues with pages for every hidden service. These queues keep track of the pages the crawler should visit next and their priority. Pages with a low depth (the homepage and its subpages) have priority over all other pages and are placed in the priority queue. New pages have priority in the remaining pages.

We make the assumption that most new pages are added to the homepages and their subpages just like newspapers, forums, and shops do. To find new pages within a given hidden service, we put the homepages back in the priority queue. When revisited, its subpages (including new pages) will be added back to the queues. The pages with low depth will be added to the priority queue to make sure they are revisited before older pages. Pages with a higher depth are added to one of the remaining queues, giving priority to the new pages. In the end, new pages are found and downloaded fast. This makes this method so efficient.

To keep track of the depth of pages within a given hidden service, we define the introduction points, a list of pages with depth equal to zero. These are the pages that other hidden services link to. The depth increases for every link needed to reach a certain page coming from the introduction points. The crawler keeps track of this depth and can both increase and decrease the depth of a page. When a link to a page moves to a deeper page, the depth will increase. To enable this behaviour, the crawler keeps track of the cycle, a number increased once every time the introduction points (homepages) are added back to the priority queue. The cycle when a link to a page was found will get saved along with every page, making it possible to detect a removed link.

IV. RESTful API

A microservice is responsible for the storage of the queries and results. The communication with the crawler and web application is enabled by a RESTful API. It provides authentication and communicates with the MongoDB database. It also serves the web application on its main URL.

V. Web application

The web application displays the available queries and results. It allows to build advanced queries by joining text searches and regular expressions with operators (and, or) using drag and drop in a tree based editor (Figure 2).

With this application users can also view the results of their queries; mark and delete results; and safely view the HTML code of the pages that triggered a query.

VI. Results

To test the developed application, the crawler was held online for one week. We configured some queries to investigate possibly interesting data. Despite the short duration, the crawler was able to find some interesting results. These queries and their results are discussed in this section.

A. Details about the dark web

In one week the crawler found 4,854 available hidden services (Figure I). At least one page needs to be downloaded successfully in order to be classified as available. The crawler also discovered 16,162 hidden services that were not available. These

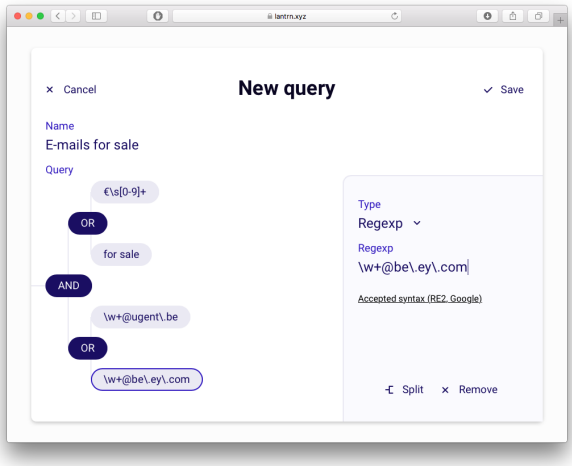


Fig. 2. Creating a query in the web application.

do not include 746,979 generated domains that were found: apparently some dark web page(s) link to generated, inexistent, domain names. They stand out because they were used as a counter, e.g.: zzzzzzzzzzzzzzqh.onion, zzzzzzzzzzzzzzqi.onion, zzzzzzzzzzzzzzqj.onion, zzzzzzzzzzzzzzqk.onion, ...

These generated domain names heavily affected the speed of the crawler because they had to be checked on availability. Their existence was only detected after inspecting the list of unavailable domains. The crawler should detect these generated domains automatically in the future.

Available hidden services	4,854
Not available	16,162
Not tested (unknown availability)	335

TABLE I

AVAILABILITY OF THE DISCOVERED HIDDEN SERVICES.

B. *Passwords and email addresses*

One query searched for lists of email addresses and discovered a lot of pages with pieces of leaked databases or cracked accounts. Table II lists the most frequently found email addresses grouped by provider. On a national level, Table II lists the same for Belgian and Dutch providers. This table illustrates that not only global internet companies (e.g. Google) are found, but also regional companies like Telenet (Belgium), albeit less numerous.

Altogether 7,373 email addresses were found in the processed results. These email addresses and passwords are often login credentials for online services, e.g. Steam or Netflix. For ethical and legal reasons, the passwords weren't tested. Odds are that a lot of these passwords are outdated. Often a person on a dark web forum published these small lists for free in order to gain trust and customers for his/her complete lists.

# email addresses	Provider
2033	gmail.com
1283	hotmail.com
936	yahoo.com
203	aol.com
160	mail.ru
135	hotmail.co.uk
130	live.nl
82	msn.com
68	ic.fbi.gov
66	comcast.net
55	yahoo.co.uk
48	sbcglobal.net
41	live.co.uk
41	btinternet.com
36	mac.com

TABLE II

TOP 15 DISCOVERED EMAIL ADDRESSES GROUPED BY PROVIDER.

# email addresses	Provider
130	live.nl
18	hotmail.nl
16	telenet.be
4	quicknet.nl
3	planet.nl
3	kpnmail.nl
2	zonnet.nl
2	ziggo.nl
2	xs4all.nl
2	upcmail.nl
2	hva.nl
2	chello.nl
1	vestingfinance.nl
1	tele2.nl
1	skynet.be
1	partyclick.nl
1	minaz.nl
1	mail.be
1	live.be
1	home.nl

TABLE III

TOP 20 DISCOVERED BELGIAN AND DUTCH EMAIL ADDRESSES GROUPED BY PROVIDER.

C. *Credit and debit cards*

One query searched for mentions of Belgian banks and found some sites that sell credit and debit cards.

D. *Other*

Queries to gain information about the WannaCry ransomware and exploits (some unpatched) used by the NSA did not result in interesting information [6]. Also a query to search for information about nuclear facilities in Belgium did not find anything.

These queries probably needed more time.

A query that searched for forged passports resulted in some marketplaces.

VII. Conclusion

The tool primarily found personal information during the short test period. Particularly free samples of bigger paid lists. These parts are outdated because they are easily accessible by everyone. However, this information provides insight in which companies have to deal with this type of leaks. Not only big international companies like Google, Netflix, or Spotify, but also smaller local telecom providers and governments are exposed.

This application could be a valuable source for threat intelligence when further developed and used without interruption.

References

[1] Hamilton Place Strategies. (2016, February 2). *"Cybercrime Costs More Than You Think"* [Online]. Available: https://www.hamiltonplacestrategies.com/insights/cybercrime-costs-more-you-think/

[2] Gartner. (n.d.). *"Threat Intelligence: What is it, and How Can it Protect You from Todays Advanced Cyber-Attacks?"* [Online]. Available: https://www.gartner.com/imagesrv/media-products/pdf/webroot/issue1_webroot.pdf [2017, June 9]

[3] The Tor Project, Inc. (n.d.).. *Tor path selection specification*. Available: https://gitweb.torproject.org/torspec.git/tree/path-spec.txt

[4] K. Bauer, D. McCoy, D. Grunwald, T. Kohno and D. Sicker. (2007). *"Low-Resource Routing Attacks Against Tor"* [Online]. Available: https://www.freehaven.net/anonbib/cache/bauer:wpes2007.pdf

[5] Arma. (2009, February 18). *"One cell is enough to break Tor's anonymity"* [Online]. Available: https://blog.torproject.org/blog/one-cell-enough

[6] L.V. Lonkhuyzen. (2017, May 14). *"WannaCry werd mede mogelijk gemaakt door lek bij de NSA"* [Online]. Available: https://www.nrc.nl/nieuws/2017/05/14/wannacry-kwam-bij-de-nsa-vandaan-9075218-a1558565

Inhoudsopgave

Voorwoord **I**

Toelating tot bruikleen **II**

Abstract **III**

Extended abstract **IV**

Inhoudsopgave **VIII**

Gebruikte afkortingen **XII**

1 Inleiding **1**

1.1 Onderzoeksvragen 2

1.2 Structuur 3

2 Achtergrond **4**

2.1 Introductie tot het darkweb 4

2.2 The Onion Router (Tor) 5

2.3 Crawlen van het darkweb 9

2.3.1 OnionScan 9

2.4 Gerelateerd werk 10

2.4.1 Darknet and Deepnet Mining for Proactive Cybersecurity Threat Intelligence 10

3 Opstelling **12**

3.1 Serveropstelling 12

3.2 Go 14

3.3 MongoDB 15

4 Risicoanalyse **16**

4.1 Transparantieverlies 17

4.1.1 Oorzaken 17

4.1.2 Gevolgen 18

4.1.3 Aanpak 18

4.1.4 Conclusie 19

4.2 Serveridentificatie 19

4.2.1 Oorzaken 19

4.2.2 Gevolgen 20

4.2.3 Aanpak 20

4.2.4 Conclusie 21

4.3 Identificatie van de onderzoekers 21

4.3.1 Oorzaken 21

4.3.2 Gevolgen 22

4.3.3 Aanpak 22

4.3.4 Conclusie 23

4.4 Juridische vraagstukken 23

5 Implemenatie van de crawler **24**

5.1 Langdurige zoekopdrachten 24

5.1.1 Voordelen 25

5.2 De crawler 26

5.3 De worker 27

5.3.1 Wachtrijen 28

5.3.2 Downloaden van pagina's 30

5.3.3 Het verwerken van gedownloade pagina's 32

5.3.4 Crawlcyclus 34

5.3.5 Het verwerken van een link (NewReference methode) 35

5.3.6 Geheugenbeheer 38

5.3.7 Data opslag 38

5.3.8 Onbereikbare hidden services 39

5.3.9 Zoekopdrachten . . . 39
5.4 Testen . . . 40
5.5 Online plaatsen . . . 41
5.6 Activiteiten . . . 41

6 Implementatie van de RESTful API 42
6.1 Datastructuur . . . 42
6.1.1 Zoekopdracht (Query) . . . 43
6.1.2 Resultaat (Result) . . . 43
6.1.3 Gegroepeerde resultaten (AggregatedResult) . . . 45
6.2 Beschikbare endpoints . . . 46
6.3 Authenticatie en beveiliging . . . 47
6.4 Online plaatsen . . . 49
6.5 Activiteiten . . . 49

7 De front-end webapplicatie 50
7.1 Ontwerp . . . 50
7.2 JavaScript . . . 52
7.2.1 Nieuwe zoekopdracht samenstellen . . . 52
7.2.2 Weergeven van darkweb pagina's . . . 53
7.3 Activiteiten . . . 56

8 Resultaten 57
8.1 Informatie over het darkweb . . . 57
8.2 Persoonsgegevens . . . 59
8.3 Vervalste identiteitskaarten . . . 63
8.4 Vermeldingen van Belgische bedrijven . . . 63
8.5 Vermeldingen van de UGent of KULeuven . . . 63
8.6 Informatie over Belgische kerncentrales . . . 63
8.7 WannaCry ransomware . . . 64

9 Conclusie **65**

10 Toekomstig werk **67**

Bibliografie **69**

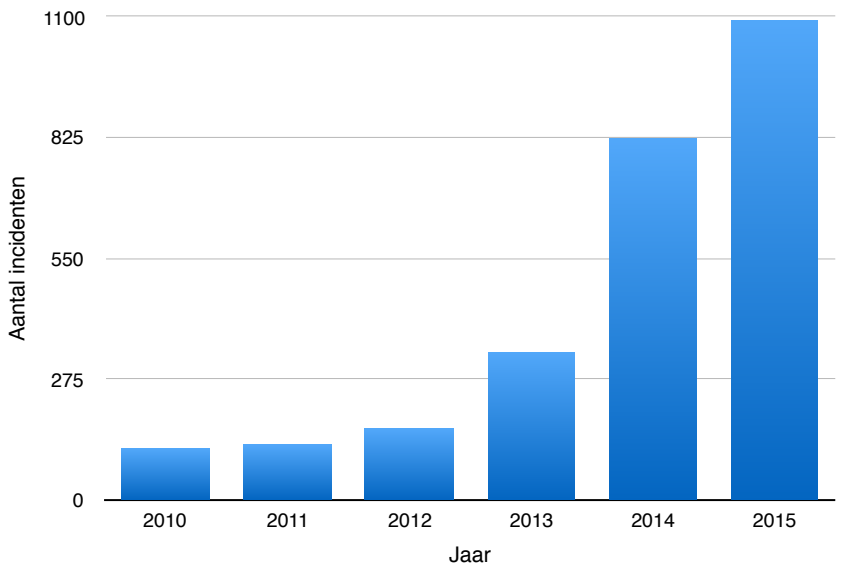
Gebruikte afkortingen

AWS	Amazon Web Services
BSON	Binary JSON
DDoS	Distributed denial-of-service-aanval
DHT	Distributed hash table
DNS	Domain Name System
DOM	Document Object Model
DoS	Denial-of-service-aanval
HSD	Hidden service discriptor
I2P	Invisible Internet Project
JSON	JavaScript Object Notation
TLD	Topleveldomein
Tor	The Onion Router
VPC	Virtual private cloud
VPN	Virtual Private Network
WWW	World wide web

Hoofdstuk 1

Inleiding

Bedrijven en banken digitaliseren massaal. Cybercriminelen krijgen hierdoor steeds meer potentiële doelwitten. Gevoelige informatie zoals financiële en medische gegevens kan in grote hoeveelheden in één keer worden bemachtigd. Systemen zoals nutsvoorzieningen, toegangsdeuren en zelfs bankrekeningen kunnen vanop afstand worden gemanipuleerd. Het verbaast dan ook niet dat het aantal cyberincidenten jaar na jaar stijgt (Figuur 1.1).



Figuur 1.1: Gemiddeld aantal gemelde cyberincidenten per maand bij CERT.be [4]

Het geschatte waardeverlies in Nederland door cybercrime wordt door Deloitte geschat op 10 miljard euro per jaar [2]. De grootste slachtoffers zijn de publieke sector (2,4 miljard), de technologiebedrijven (1,1 miljard), defensie (415 miljoen) en banken (360 miljoen).

Wereldwijd wordt de totale kost geschat op $450 miljard in 2016, een verdubbeling in 5 jaar tijd [3].

Vooral grote bedrijven en banken hebben er baat bij dat hun computersystemen afgesloten blijven. Een inbraak in een bank of zelfs een kerncentrale kan reusachtige gevolgen hebben voor de maatschappij. De onlineverkoop van inloggegevens van werknemers kan hier aan de basis liggen.

Beveiligingslekken zijn bijna onvermijdelijk. Bedrijven kunnen bijvoorbeeld gebruik maken van software met beveiligingslekken die nog niet bekend zijn bij de ontwikkelaars (ook *zero-day exploits* genoemd). Het is dan belangrijk dat bedrijven zo snel mogelijk op de hoogte worden gebracht van deze gevaren.

Cyber threat intelligence is de kennis van bestaande en toekomstige gevaren voor computersystemen. Deze kennis omvat o.a. de oorzaken, gevolgen, herkenningspunten, werking, en advies voor beslissingen die moeten worden genomen als reactie op deze gevaren [1]. Dit laat toe zich beter voor te bereiden tegen mogelijke aanvallen. Om deze kennis te voeden is op voorhand informatie nodig over gebruikte aanvalsstrategieën, potentiële veiligheidsproblemen, bestaande malware, toekomstige doelwitten en doelwitten uit het verleden.

De bronnen om aan deze inlichtingen te geraken zijn enorm divers. Vaak wordt deze informatie gedeeld door bedrijven, overheden en experts. Deze worden *threat intelligence service providers* genoemd. Zij maken hun kennis meestal niet publiekelijk beschikbaar [1].

Er bestaan ook vrije gegevensbanken met *threat intelligence*. Slatman lijst in 'awesome-threat-intelligence' enkele bronnen op [5]. Volgens Metivier zijn deze bronnen echter zo talrijk dat er een te veel aan informatie ontstaat voor bedrijven [6].

1.1 Onderzoeksvragen

Dit onderzoek heeft zich gefocust op een mogelijke informatiebron: het darkweb. Het darkweb staat hackers toe om anoniem met elkaar te communiceren, wat het ideaal maakt om gestolen persoonsgegevens en veiligheidslekken te delen.

Deze masterproef bespreekt de ontwikkeling van een hulpmiddel dat het mogelijk maakt om gerichte opzoekingen te doen in het darkweb. De volgende vragen zullen daarbij beantwoord worden:

1. Wat zijn de risico's verbonden aan het doorzoeken van het darkweb?
2. Welke bruikbare informatie kan er gevonden worden in het darkweb? Waar bevindt zich deze informatie?
3. Hoe waardevol zijn deze inlichtingen, wat is de kwaliteit en kwantiteit?

1.2 Structuur

Allereerst geven we in Hoofdstuk 2 wat achtergrondinformatie over het darkweb dat onderzocht zal worden. Aangezien het onderzoek in een crimineel milieu plaatsvindt, wordt ook een risicoanalyse uitgevoerd. Hiervoor is het belangrijk om eerst een idee te hebben van de serveropstelling, die wordt eerst in Hoofdstuk 3 behandeld. De risicoanalyse zal hierna worden voorgesteld in Hoofdstuk 4.

Eens de risico's in beeld waren gebracht, kon op een veilige manier een crawler ontwikkeld worden die doorheen het darkweb zoekt. De werking van deze crawler zal stap voor stap besproken worden in Hoofdstuk 5. Daarna volgt Hoofdstuk 6. Dit bespreekt de microservice die instaat voor de dataopslag van de zoekopdrachten en resultaten. De webapplicatie waarmee gebruikers zoekopdrachten kunnen aanmaken en de resultaten ervan kunnen bekijken wordt besproken in Hoofdstuk 7.

Aan de hand van enkele mogelijke toepassingen wordt de uitgewerkte applicatie geëvalueerd in Hoofdstuk 8. In de conclusie zullen de onderzoeksvragen worden beantwoord aan de hand van deze resultaten. Tot slot wordt het onderzoek kritisch geëvalueerd en worden er tips voor volgend onderzoek opgelijst in Hoofdstuk 10.

Hoofdstuk 2

Achtergrond

2.1 Introductie tot het darkweb

Het darkweb is een onderdeel van het internet dat enkel bereikbaar is met speciale software. Het wordt vaak verward met het deepweb, dat is het vanaf zoekmachines onbereikbare deel van het internet. Dat kan bijvoorbeeld gaan over pagina's die beschermd zijn met inloggegevens, zoals bijvoorbeeld een Facebook-profiel of vertrouwelijke documenten. Het deel van het internet dat iedereen kan bereiken vanaf een gewone browser heet het clearweb.

De term darkweb klinkt erg mysterieus en gevaarlijk. Het wordt dan ook vaak in verband gebracht met kinderporno, terrorisme, wapen- en mensenhandel, drugs, hacken en andere criminele activiteiten. Het grootste deel van het darkweb bestaat inderdaad uit die zaken, maar het is er zeker niet tot beperkt. Men kan er ook blogs van ondergedoken activisten terugvinden. Journalisten en klokkenluiders kunnen er op een veilige manier communiceren [7]. Het darkweb is dus zeker niet alleen een plek voor criminelen.

Eén van de kenmerken van een darkweb is dat zowel de bezoeker als de website er anoniem kunnen communiceren. Dit kan alleen als de IP-adressen van beide partijen verborgen blijven. Het *Domain Name System* (DNS) kan bijgevolg niet gebruikt worden voor het opzoeken van domeinnamen. Websites in het darkweb bestaan niet uit een domeinnaam gevolgd door een bekend topleveldomein (TLD) zoals .be of .com. Een darkweb moet hier dus een alternatief voor aanbieden.

Er bestaan verschillende softwareoplossingen die elk toegang bieden tot een afzonderlijk darkweb. Gebruikers hebben bijgevolg met de ene oplossing geen toegang tot het darkweb van een andere oplossing. Enkele voorbeelden zijn Tor, Invisible Internet Project (I2P) en Freenet [7, 9, 10]. Deze masterproef beperkt zich tot Tor.

2.2 The Onion Router (Tor)

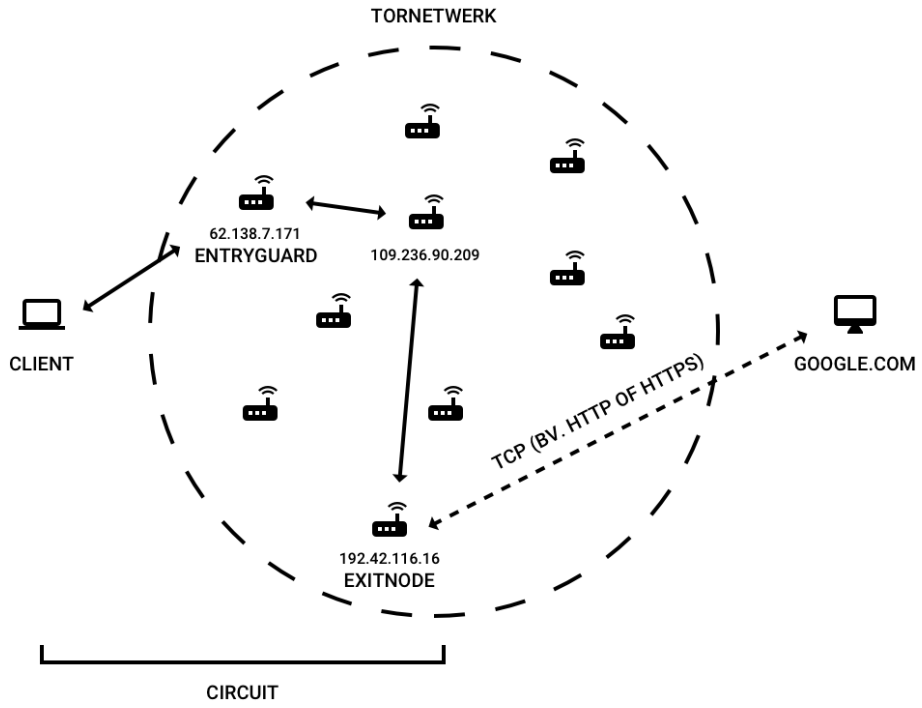
Met Tor, afkorting voor *The Onion Router*, is het ook mogelijk om het clearweb te bezoeken [7]. Via Tor kan men dus naar eender welke website surfen. Daarbij blijft de bezoeker – niet de website – anoniem. Het anoniem houden van de bezoeker – hierna client genoemd – is dan ook het eerste dat zal worden besproken. Dit vormt ook de basis van het darkweb. De informatie in deze sectie is gebaseerd op de specificaties en documentatie van Tor zelf [11–14].

Tor beschikt over een groot netwerk van servers en computers die door vrijwilligers ter beschikking gesteld zijn. Deze worden nodes, routers of relays genoemd. Al deze nodes zijn opgenomen in een lijst, die beschikbaar is via enkele *directory servers* [12].

Figuur 2.1 illustreert hoe een anonieme verbinding met een server tot stand komt. Hiervoor bouwt de client stap voor stap een pad van geëncrypteerde verbindingen tussen verschillende nodes in het Tor netwerk, dit heet een circuit [13]. Hierbij kennen alle nodes op het pad enkel de twee nodes waarmee ze rechtstreeks communiceren, en dus niet het volledige circuit. De nodes die rechtstreeks in verbinding staan met de eindbestemming (de server) of de client heten respectievelijk de *exitnode* en de *entryguard* (ook *entrynode*).

Het is mogelijk om *Transmission Control Protocol* (TCP) verkeer door een circuit te sturen. Tor zorgt enkel voor de encryptie binnenin een circuit. Dat wil zeggen dat de exitnode het ongeëncrypteerde TCP-verkeer kan bekijken en wijzigen [14]. Om dit te voorkomen kan het onderliggende TCP-verkeer via een end-to-end geëncrypteerd protocol zoals HTTPS (*HyperText Transfer Protocol Secure*) verstuurd worden.

Voor de server lijkt het alsof al het verkeer afkomstig is vanaf de exitnode, zodat de anonimiteit van de client gegarandeerd is zolang niet alle nodes in het circuit gecompromitteerd zijn. Een circuit bestaat doorgaans uit drie nodes.



Figuur 2.1: Opbouw van een Tor-circuit [13].

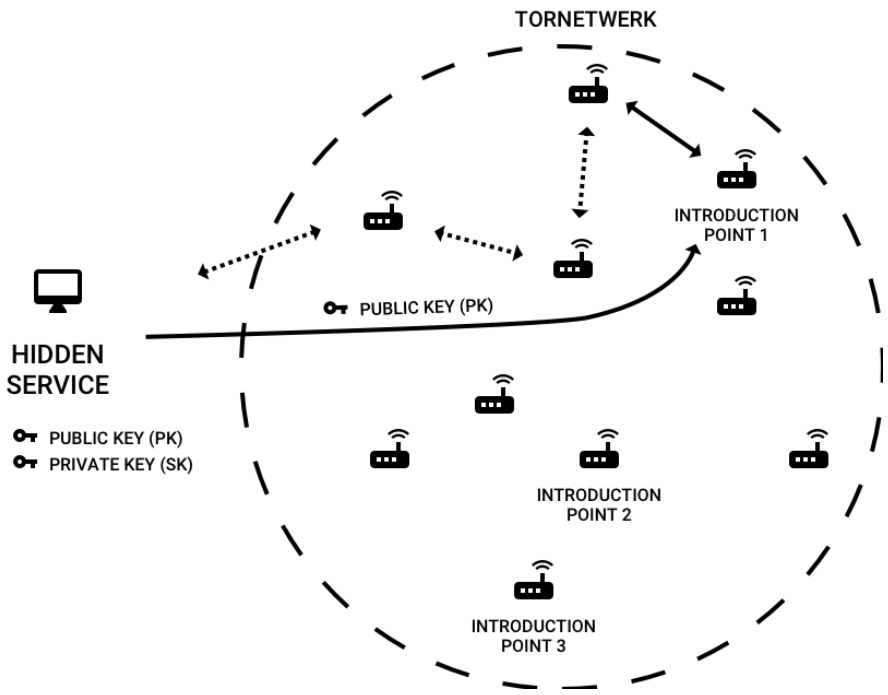
Zodra het circuit is opgesteld kan de client er TCP-data over verzenden. Uit performantieoverwegingen behoudt Tor hetzelfde circuit voor nieuwe verbindingen gedurende 10 minuten. Pas daarna wordt een nieuw circuit opgesteld.

Om een echt darkweb te kunnen vormen is het noodzakelijk dat ook de servers die websites aanbieden anoniem kunnen blijven. In Tor heten deze *hidden services*. Een hidden service moet zich eerst registreren en kenbaar maken voordat het gevonden kan worden.

1. Hiertoe genereert de hidden service een publieke en daarbij horende private sleutel. Het adres waarop de server bereikbaar is, wordt gevormd door de base32-codering van de 80 eerste bits van de SHA1-hash van de publieke sleutel (160 bits) [11]. Dit adres bestaat hierdoor uit 16 karakters gevolgd door .onion en is dus niet vrij te kiezen. Via het herhaaldelijk opnieuw genereren van de sleutels kan deze wel gekozen worden zodat het bij de website past (de bruteforce methode). Facebook maakt zichzelf bijvoorbeeld toegankelijk via facebookcorewwwi.onion in het Tor darkweb [8].

2. Vervolgens kiest de hidden service enkele Tor-nodes als *introduction points*. Elk *introduction point* ontvangt de publieke sleutel als identificatie (Figuur 2.2). Deze

communicatie gebeurt over verschillende circuits (in stippellijn op de figuur) zodat het IP-adres van de hidden service verborgen blijft voor de introduction points. De introduction points houden de publieke sleutel bij, samen met het circuit waarop die ontvangen werd.

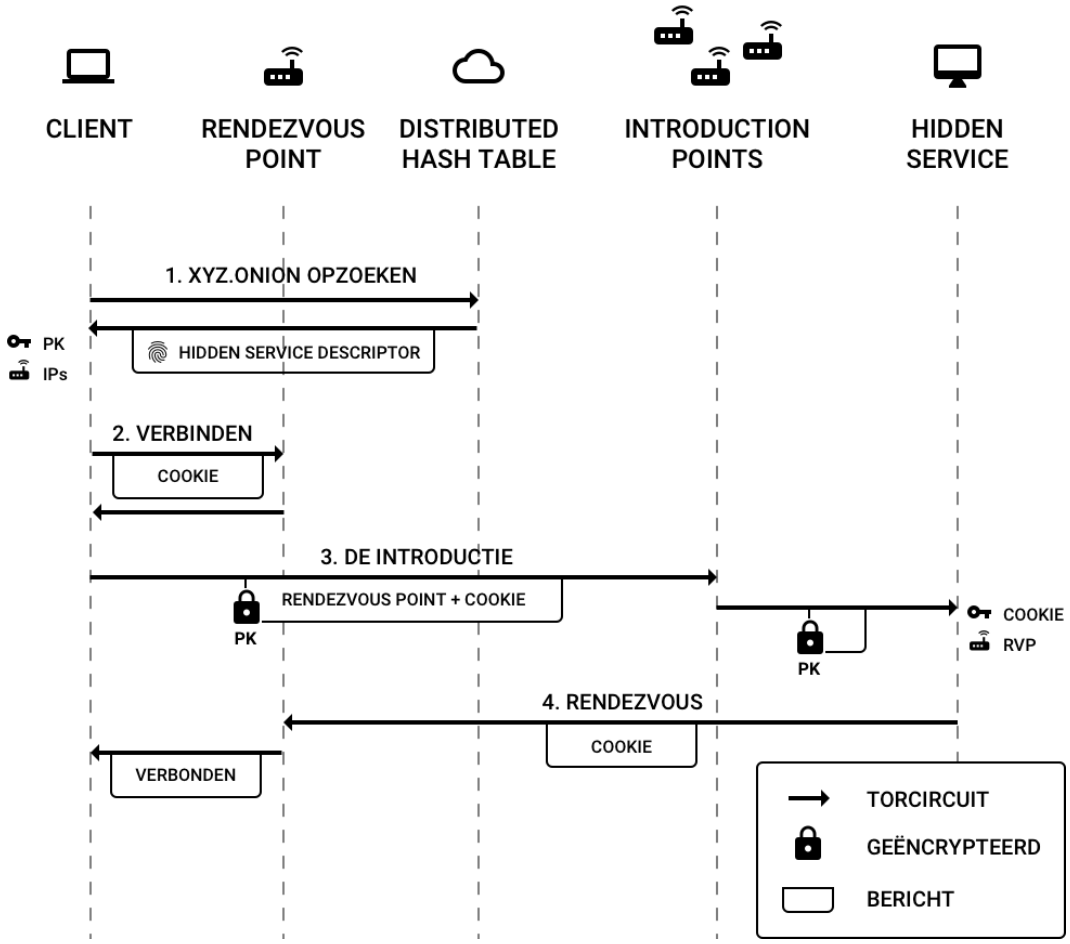


Figuur 2.2: Kiezen van introduction points [11].

3. De *hidden service* maakt een *hidden service discriptor* (HSD), die bevat zijn publieke sleutel en een beschrijving van elk *introduction point*. Het bericht wordt ondertekend met de private sleutel van de hidden service. De hidden service publiceert deze vervolgens in de *distributed hash table* (DHT) van het Tor-netwerk. Dit is een gedistribueerde hashtabel die bewaard wordt door verschillende Tor-nodes die hiervoor aangeduid zijn door de beheerders van het Tor-netwerk.

De hidden service is nu geregistreerd en is bereikbaar voor clients. Een client kan een circuit met een hidden service opzetten door het volgen van de onderstaande stappen. Zie hiervoor ook Figuur 2.3.

1. De client zoekt aan de hand van het adres van de hidden service de HSD op in de DHT van het Tor-netwerk. Hieruit haalt de client de publieke sleutel en de *introduction points* van de hidden service.



Figuur 2.3: Schematisch overzicht van hoe het circuit tussen client en hidden service tot stand wordt gebracht [11].

2. De client bouwt een circuit naar een willekeurige Tor-node, en verstuurt een bericht met daarin een gegenereerde waarde van 20 bytes, de *rendezvous cookie*. De Tor-node bevestigt dit met een antwoordbericht. Vanaf nu heet deze het *rendezvous point*. Het *rendezvous point* houdt het bestaande circuit bij en associeert het met deze cookie.

3. De client maakt een bericht met daarin de beschrijving van zijn *rendezvous point* en de cookie. Het bericht wordt versleuteld met de publieke sleutel van de hidden service en samen met de identificatie van de publieke sleutel verstuurd naar één van de *introduction points* van de hidden service. De *introduction point* geeft het daarna door aan de hidden service via het reeds bestaande circuit dat hij herkent aan de hand van de publieke sleutel.

4. De hidden service decodeert het bericht met zijn private sleutel en maakt een circuit aan naar het *rendezvous point* van de client. Het wachtwoord uit het bericht wordt gebruikt als identificatie.

Er wordt een circuit vanaf de client naar de server aangemaakt als een .onion-domeinnaam wordt aangevraagd. Deze bestaat uit 6 nodes: drie voor het circuit vanaf de client tot en met het *rendezvous point* en drie voor het circuit tussen de server en het *rendezvous point*. Tor-aanvragen naar gewone websites uit het clearweb verlopen gewoon via het eenvoudige circuit dat eerder werd besproken.

2.3 Crawlen van het darkweb

Crawlen is het automatisch doorzoeken van het internet door software. Google moet dit bijvoorbeeld doen om zijn zoekdiensten mogelijk te maken [16]. Een (web)crawler (ook wel scraper, spider of bot genoemd) downloadt enkele webpagina's en herhaalt dat vervolgens voor alle links op die pagina's. Reeds gedownloade pagina's worden genegeerd en pas na verloop van tijd opnieuw bezocht. Het world wide web (WWW) kan zo volledig worden geïndexeerd als de crawler bij voldoende webpagina's begint. Bijgevolg zijn websites (of groepen van websites) zonder inkomende links onvindbaar op een zoekmachine. Deze horen daarom bij het deepweb.

Het is geen slecht idee om op voorhand stil te staan bij hoe het darkweb verschilt van het clearweb. Op het clearweb zijn alle websites namelijk verbonden via vele interne links waardoor het mogelijk is voor een crawler om het te indexeren. Maar hoe groot is het darkweb, en hoe is het gesteld met de interne links tussen hidden services?

2.3.1 OnionScan

OnionScan is een open source hulpmiddel dat het darkweb onderzoekt om onderzoekers en operators van hidden services te helpen. Geregeld brengt het een rapport naar buiten met informatie over het darkweb. In maart 2017 had OnionScan weet van 30.000 verschillende .onion-domeinnamen. Daarvan waren er ongeveer 4.000 via HTTP en 400 via een ander protocol bereikbaar (bv. SSH) [20]. De resterende waren helemaal onbereikbaar. Tussen

die hidden services bevinden zich ook domeinen die dezelfde website tonen, vaak dienen deze voor load balancing en redundantie zodat aanvragen kunnen worden verspreid over meerdere servers [18].

OnionScan voerde ook een studie uit naar de darkweb market *Hansa Market*. Daaruit blijkt dat in 15 maanden tijd meer dan 3 miljoen dollar werd verhandeld [21]. Drugs is één van de meest verkochte producten. Digitale producten nemen maar een klein percentage van de total omzet in beslag.

Eén van de rapporten bespreekt de connectiviteit van hidden services. Daaruit blijkt dat er een erg goede connectiviteit is tussen clusters van websites die naar elkaar linken [19].

2.4 Gerelateerd werk

2.4.1 Darknet and Deepnet Mining for Proactive Cybersecurity Threat Intelligence

De paper *Darknet and Deepnet Mining for Proactive Cybersecurity Threat Intelligence* van Nunes et al. bespreekt een uitgewerkt systeem dat op het tijdstip van publicatie elke week gemiddeld 305 cyberthreat-waarschuwingen verzameld had [22]. Dit werd verwezenlijkt door alle producten en berichten op enkele gekozen markets en forums in het darkweb te verzamelen. Men liet een deel van deze producten en berichten door experts beoordelen als betrouwbaar of onbetrouwbaar. Hierna werden enkele *machine learning* technieken gebruikt om hetzelfde te doen voor de overige en toekomstige data. De paper vergelijkt hierbij verschillende *supervised machine learning* technieken: *naive bayes* (NB), *random forest* (RF), *support vector machine* (SVM) en *logistic regression* (LOG-REG), maar ook *semi-supervised* technieken zoals *label propagation* (LP) en *co-training* (CT).

Bij deze aanpak moet de crawler op voorhand per website (dus per hidden service) aangeleerd krijgen waar bepaalde kenmerken in de HTML-broncode kunnen worden gevonden. Kenmerken zoals prijs, titel, beschrijving en datum kunnen zo automatisch gevonden worden voor de HTML-structuren van aangeleerde websites. Om verder te kunnen bouwen op dit onderzoek heb ik nagekeken of het niet mogelijk was om de plaats van deze zaken automatisch te herkennen voor alle markets en forums, zonder dat deze manueel moeten

worden aangeleerd. Hiervoor werden de HTML-broncodes van pagina's van verschillende markets gedownload en opgeslagen. Een zelfgeschreven programma bepaalde aan de hand van statistische kenmerken, zoals de lengte van de tekst, aantal speciale symbolen, gebruikte HTML-tags, afstand tussen tags en gebruik van bepaalde woorden, waar deze kenmerken gevonden konden worden in de gedownloade broncodes. Na lang nauwkeurig afstellen bleek het moeilijk om de kenmerken zonder al te veel fouten te lokaliseren.

De noodzaak om dit aan te leren is bijgevolg een groot nadeel van deze aanpak. Daarom zal deze masterproef geen onderscheid maken tussen markets, forums of andere hidden services. Op die manier kan vrij worden gecrawld op het darkweb en blijft de implementatie onveranderd als de HTML-structuur van webpagina's wijzigt. Hoofdstuk 5 formuleert de concrete werkwijze.

Hoofdstuk 3

Opstelling

Dit hoofdstuk beschrijft de software- en hardwareopstelling die in de masterproef gebruikt wordt.

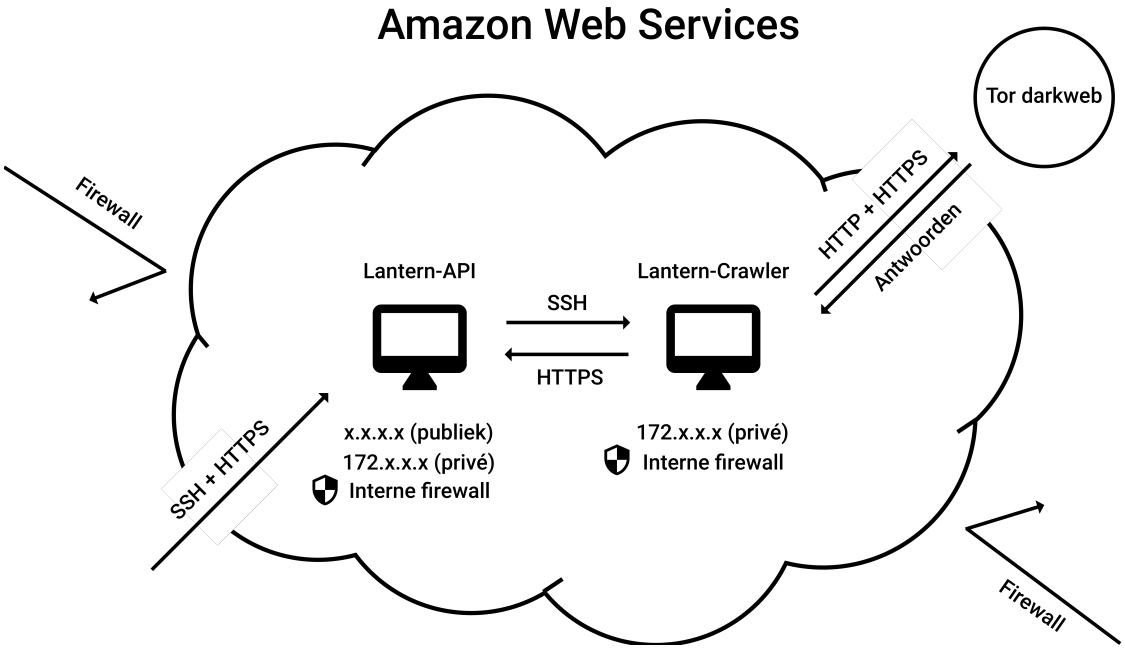
3.1 Serveropstelling

Het budget zal zoveel mogelijk beperkt worden. De hosting van de servers is hierbij de enige kost. Een zicht op het aantal te crawlen pagina's is noodzakelijk om de capaciteit van het systeem te berekenen.

Niet alle gedownloade pagina's zullen worden opgeslagen. Het systeem zal zo worden geïmplementeerd dat de zoekcriteria op voorhand bekend zijn (zie Hoofdstuk 5). Een pagina wordt opgeslagen als ze aan één van de criteria voldoet. Als het 7 dagen duurt om alle relevante websites te crawlen, zal het bijgevolg ook 7 dagen duren voor alle zoekresultaten van een zoekopdracht gevonden zijn.

Er zijn ongeveer 4.000 hidden services bereikbaar via HTTP volgens de OnionScan rapporten (zie Sectie 2.3.1). Het is niet meteen duidelijk hoeveel verschillende pagina's elke hidden service bevat. Een voorlopige schatting geeft aan dat elke hidden service ongeveer 1.000 nieuwe pagina's bevat elke week. Dat zijn 4 miljoen pagina's in totaal. Als al deze pagina's in 7 dagen tijd moeten worden verwerkt, dan moeten er minimum 6,7 pagina's per seconde verwerkt worden.

Aangezien dit aantal relatief laag is, zal het voor dit onderzoek voldoende zijn om met slechts één crawlserver te werken. Dit moet vooral veel tijd besparen en ruimte bieden om belangrijkere zaken te onderzoeken.



Figuur 3.1: Serveropstelling

Er worden in totaal twee servers gebruikt (zie Figuur 3.1). Beide servers worden gehost binnen Amazon Web Services (AWS). Het voordeel van AWS is dat er naast de vaste kosten, enkel kosten worden aangerekend voor uitgaand netwerkverkeer, en dat is bij een crawler lager dan het inkomende netwerkverkeer. Een crawler downloadt pagina's in tegenstelling tot een webserver die pagina's uploadt naar gebruikers. Bovendien voorziet AWS een configureerbare firewall rondom de virtual private cloud (VPC), dat is de verzameling van het netwerk waar alle AWS-servers in een bepaalde regio zijn op aangesloten [23].

Eén server is verantwoordelijk voor het crawlen van alle pagina's. Deze server krijgt de naam Lantern-Crawler. 'Lantern' is de gekozen projectnaam, beeldspraak voor het doorzoeken van het darkweb met een lantaarn. De reden dat een projectnaam werd gekozen is om een duidelijke naam te kunnen voorzien voor de bestanden, repositories, servers en de domeinnaam van het project.

De andere server is verantwoordelijk voor de opslag van gegevens en het beschikbaar stellen van de front-end applicatie. Deze server krijgt de naam Lantern-API. Enkel deze server staat netwerkverkeer toe van buiten de VPC. Deze server is ook de enige server met een statisch publiek IP-adres. De Lantern-Crawler krijgt een tijdelijk publiek IP-adres dat niet verbonden wordt aan het AWS-account. De private IP-adressen zijn niet bereikbaar vanaf het internet [23]. Beide servers hebben een streng ingestelde firewall met behulp van iptables. Enkel uitgaand HTTP, HTTPS en DNS verkeer en de antwoorden daarop zijn toegestaan omwille van de betere bescherming tegen malware en hackers.

SSH-verkeer naar de Lantern-Crawler server is enkel toegestaan vanaf het IP-adres van de Lantern-API server. SSH- en HTTPS-verkeer is wel toegestaan van buiten AWS naar de Lantern-API server. Hierdoor kan iemand met slechte bedoelingen uit het darkweb niet zomaar connecteren met het (tijdelijke) publieke IP-adres van de Lantern-Crawler, aangezien dat wordt blootgesteld aan het darkweb. SSH-toegang tot die server blijft echter wel toegankelijk vanaf de Lantern-API server. Het is dus mogelijk om eerst een SSH-verbinding op te zetten met de Lantern-API server en vanaf daar een nieuwe SSH-verbinding te starten met de Lantern-Crawler server.

3.2 Go

Go, vaak aangeduid als Golang, is een programmeertaal ontwikkeld door Google [24]. De taal is open source. Go werd als programmeertaal voor dit onderzoek gekozen om een aantal redenen:

1. Go gebruikt goroutines, een lichtgewicht alternatief voor threads. Hierdoor is de programmeertaal erg geschikt voor serverapplicaties zoals crawlers. Goroutines zijn bovendien eenvoudig om mee te werken.

2. Go biedt oplossingen voor synchronisatieproblemen out-of-the-box d.m.v. channels en *mutual exclusion* (mutex).

3. Gecompileerde programma's zijn uitermate efficiënt.

4. Go bevat uitgebreide standaardbibliotheken, bijvoorbeeld voor het verwerken van HTML of het downloaden via HTTP via een proxy.

5. Er zijn veel open source bibliotheken voor Go beschikbaar.

3.3 MongoDB

Vanwege de hoge flexibiliteit van MongoDB database werd deze gekozen voor de opslag van data [25]. MongoDB is geen relationele database (NoSQL), hierdoor is het mogelijk om snel nieuwe velden toe te voegen aan bestaande collecties of documenten. MongoDB zal het later ook eenvoudiger maken om de recursieve structuur van zoekopdrachten op te slaan, zonder dat hiervoor verschillende tabellen moeten worden aangemaakt.

Hoofdstuk 4

Risicoanalyse

Alvorens een risicoanalyse wordt opgesteld is het belangrijk om stil te staan bij wat er met de gevonden data zal worden gedaan. Zo is het niet de bedoeling om actief op zoek te gaan naar bijvoorbeeld wapenhandel of mensenhandel. Dan zouden de gevolgen van de hierna besproken risico's aanzienlijk vergroot worden. Wat voor deze masterproef wel interessant is, zijn de software gerelateerde zaken: exploits, inloggegevens, malware, ...

In dit hoofdstuk zal dieper worden ingegaan op volgende risicofactoren:

1. Transparantieverlies: Een hidden service ontdekt dat een crawler actief is. Het IP-adres, de identiteit of locatie van de onderzoeksserver zijn onbekend.
2. Serveridentificatie: Het IP-adres van de onderzoeksserver kan worden gelinkt aan crawlactiviteiten in het darkweb. De intenties van de crawler zijn niet bekend.
3. Identificatie van de onderzoekers: De identiteit van onderzoekers wordt gelinkt aan crawlactiviteiten in het darkweb.

Mijn begeleiders bij EY raadden aan om de risico's voor te stellen aan de hand van een risicomatrix. Een risicomatrix zorgt ervoor dat belangrijke risico's in één oogopslag kunnen worden herkend (zie Figuur 4.1). Hiervoor wordt elk risico onderverdeeld op basis van zijn kans en impact op een schaal van 1 tot 5. Om dat te bepalen worden de oorzaken en gevolgen van elk risico besproken. Dit wordt vervolgens gekoppeld aan een concrete aanpak die het risico moet verkleinen.

Impact \ Kans	Zeer onwaarschijnlijk	Onwaarschijnlijk	Mogelijk	Kansrijk	Zeer kansrijk
Minimaal				Transparantie-verlies	
Matig		Server-identificatie			
Ernstig					
Groot	Identificatie van de onderzoekers				
Catastrofaal					

Figuur 4.1: Risicomatrix

4.1 Transparantieverlies

Bij transparantieverlies ontdekt een hidden service dat onze crawler actief is. Het IP-adres, de identiteit of locatie van de onderzoeksserver blijven onbekend.

4.1.1 Oorzaken

Een crawler kan herkend worden als hij abnormaal gedrag vertoont. Het valt snel op in de logbestanden van een server als een crawler op een snel tempo alle pagina's downloadt. Bovendien zal een crawler vaak geen bronbestanden zoals afbeeldingen of stijlbestanden downloaden omdat deze van weinig belang zijn voor een computer. Een bezoeker die deze bestanden niet downloadt, kan dus al snel worden gedetecteerd als een crawler. Uit eigen onderzoek blijkt dat sommige hidden services gebruik maken van captcha's en inlogmuren, hoogstwaarschijnlijk om crawlers tegen te houden.

Om een crawler te detecteren moet de eigenaar van een hidden service actief op zoek gaan. Het kan natuurlijk zijn dat de crawler door een fout de hidden service gaat overbelasten. In dat geval zal de eigenaar waarschijnlijk op de hoogte worden gebracht. In het andere geval kan dit aan de hand van een analyse van de logbestanden. Een andere mogelijkheid om crawlers te detecteren zijn honeypots, dit zijn valstrikken waarbij de crawler bewust in een val wordt gelokt zodat hij kan worden ontmaskerd. Een voorbeeld van een honeypot

voor crawlers is een speciale link die verborgen wordt voor gewone gebruikers. Een crawler zal deze link gewoon volgen en valt zo door de mand.

4.1.2 Gevolgen

Zodra een hidden service ontdekt dat er een crawler actief is, kan deze in principe niet veel doen om de crawler tegen te houden. En darkweb zorgt er namelijk voor dat de bezoeker anoniem is, wat het bijzonder lastig maakt om het verkeer van één bezoeker te blokkeren zonder die van andere te verstoren. Bij Tor is het zo dat meerdere gebruikers via hetzelfde IP-adres (van een exitnode) connecteren met de hidden service.

4.1.3 Aanpak

In het darkweb werken heeft een groot voordeel: het is ook moeilijker om de logbestanden te analyseren doordat het IP-adres van de crawlserver geheim blijft voor de hidden service, tenzij wordt gebruik gemaakt van het zojuist beschreven identificatie systeem in URL's. Het enige wat aanvragen onderscheidt zijn de HTTP-headers (onder andere HTTP_USER_AGENT) en cookies. Het is dus belangrijk om die gegevens niet te laten opvallen, want abnormale waarden zouden de aanvragen doen opvallen. Aangezien er een populaire browser wordt meegeleverd met Tor, is het beter om dezelfde HTTP-headers te gebruiken.

Er kan beter een ruime tijd worden gewacht tussen aanvragen naar dezelfde hidden service. Op die manier vallen de aanvragen minder op in de logbestanden van een hidden service.

Veel online tutorials, zoals die van *Darkwebnews* (https://darkwebnews.com), raden het gebruik van JavaScript af [26]. Hierdoor kunnen captcha's nooit erg geavanceerd zijn in het darkweb. De sterke captcha's van Google, reCAPTCHA, zijn hierdoor bijvoorbeeld niet bruikbaar [27].

Om op afbeelding gebaseerde captcha's te ontcijferen bestaan er tegenwoordig speciale online diensten. Een voorbeeld hiervan is 2Captcha, daar kunnen captcha's naar medewerkers worden gestuurd die de tekst in de afbeelding ontcijferen. Daarna wordt de

ontcijferde captcha terug gestuurd [28]. Het foutpercentage is hierdoor enorm laag. Voor ongeveer 1,00 USD kunnen er 1000 captcha's ontcijferd worden [28].

4.1.4 Conclusie

Doordat een crawler doorgaans geen bronbestanden downloadt, en omdat het door honeypots kan ontdekt worden, is de kans groot dat de crawler zal worden gedetecteerd. Daarom wordt deze in de risicomatrix op kansrijk geplaatst. De gevolgen bij een detectie zijn – zoals eerder besproken – minimaal.

4.2 Serveridentificatie

Bij serveridentificatie kan een hidden service het IP-adres van een onderzoeksserver koppelen aan crawlactiviteiten in het darkweb. De intenties van de crawler zijn echter niet bekend voor de hidden service.

4.2.1 Oorzaken

Identificatie van de server is mogelijk als een hidden service actief op zoek gaat naar de crawlserver. Tor kan de anonimiteit van een client niet garanderen want er zijn verschillende soorten aanvallen op Tor bekend. Een voorbeeld hiervan is de *confirmation attack*. Bij deze aanval heeft de aanvaller controle nodig over een entry- en exitnode op hetzelfde circuit. Deze aanvallen zijn echter duur omdat ze veel servers en bandbreedte vereisen [29].

Passive confirmation attack

Aan de hand van een circuit-linking-algoritme kunnen aanvragen langs de entrynode gelinkt worden aan de antwoorden bij de exitnode, waardoor de anonimiteit van een Tor circuit verloren gaat [29]. Dit maakt o.a. gebruik van de tijdstippen van de verschillende aanvragen.

Tagging attack

Bij een *tagging attack* (een actieve variant van de *confirmation attack*) wordt de datastroom langs één kant aangepast zodat het aan de andere kant kan worden gedetecteerd. Op die manier kan de link tussen de servers die communiceren worden gelegd [30].

4.2.2 Gevolgen

Aan de hand van het IP-adres kan de internetprovider en soms de locatie van de crawlserver bepaald worden [31]. Daarom is het belangrijk dat geen server gebruikt wordt die op het eigen netwerk gehost wordt. Een externe hosting provider zoals Amazon Web Services (AWS) is een betere optie. In dat geval blijven de gevolgen beperkt tot een *distributed denial-of-service-aanval* (DDoS) of pogingen tot hacken van de server. Het is dus belangrijk dat ook de crawlserver hiertegen bestand is.

4.2.3 Aanpak

Verdediging tegen serveridentificatie is lastig. Men kan ten eerste proberen om veel overbodig verkeer door te sturen om ervoor te zorgen dat het patroon niet meer kan worden onderscheiden [32]. Een andere oplossing is *defensive dropping*, hierbij verstuurt de client extra lege berichten die door nodes tussen de entry- en exitnode verwijderd worden [32]. In Tor kan dit door de *time-to-live* (TTL) waarde in de IP-headers laag in te stellen. Hierdoor ontvangt de entrynode meer berichten dan de exitnode en wordt het moeilijker om patronen te herkennen. *Adaptive padding* werkt op de omgekeerde manier door nodes af en toe extra berichten te laten introduceren tussen de entry- en exitnodes [32]. Beide technieken zijn nog niet geïmplementeerd in Tor.

Een andere mogelijkheid is om het IP-adres van de server te verbergen met behulp van een *Virtual Private Network* (VPN). Dit zorgt echter voor een extra kost.

4.2.4 Conclusie

Het is onwaarschijnlijk dat een hidden service op zoek zal gaan naar het IP-adres van een crawler omdat deze aanval veel tijd en geld in beslag zal nemen. De hidden service kan er achteraf ook geen winst uithalen.

Indien het IP-adres toch wordt achterhaald, kan er nog geen identiteit aan worden verbonden. In dit project krijgt de crawlserver bovendien geen statisch publiek IP-adres toegewezen (Sectie 3.1). Het tijdelijke publieke IP-adres is gedeeld en niet gekoppeld aan een AWS account. De gevolgen zijn matig en de kans is onwaarschijnlijk op de risicomatrix.

4.3 Identificatie van de onderzoekers

Een hidden service kan de identiteit van de onderzoekers koppelen aan crawlactiviteiten in het darkweb.

4.3.1 Oorzaken

Eens een persoon met slechte bedoelingen het IP-adres van de crawlserver bezit, wordt het mogelijk om de identiteit van de achterliggende bedrijven of personen te bemachtigen. Hiervoor zijn er twee opties. Ofwel wordt ingebroken op de servers van de hostingprovider om zo de persoonlijke gegevens die horen bij het IP-adres te bemachtigen, ofwel wordt ingebroken op de crawlserver in de hoop daar persoonlijke informatie te achterhalen.

Inbraak bij hostingprovider

Een hostingprovider houdt allerlei persoonlijke gegevens bij van zijn klanten. Een hacker kan de gegevens die bij een bepaalde server horen proberen te bemachtigen. De moeilijkheidsgraad is dan sterk afhankelijk van de gekozen hostingprovider.

Hacken van de crawlserver

Een hacker kan proberen toegang te krijgen tot de crawlserver. Hiervoor zou hij speciale malware kunnen ontwerpen die de crawler kan besmetten om zo meer informatie over de server te bemachtigen. In dat geval is het niet noodzakelijk dat het IP-adres gekend is. Een hacker zou ook kunnen proberen om via een veiligheidslek toegang te krijgen tot de server als het IP-adres gekend is.

4.3.2 Gevolgen

Eens een hidden service de identiteit van de onderzoekers kan achterhalen, kan hij die onder druk zetten om de crawler af te sluiten. Dat kan gebeuren door bijvoorbeeld een *distributed denial-of-service-aanval* (DDoS) uit te voeren op belangrijke servers van de onderzoekers. De onderzoekers kunnen ook het slachtoffer worden van hacking, diefstal, chantage, fysieke bedreigingen of in het meest extreme geval zelfs moord.

4.3.3 Aanpak

De crawlserver moet goed beveiligd worden en de firewall mag op een paar uitzonderingen na geen inkomend verkeer toestaan. In dit project staat de Lantern-Crawler server geen inkomend verkeer toe met uitzondering van SSH-verkeer afkomstig van de Lantern-API server.

Malware besmetting kan voorkomen worden door ongebruikte software te verwijderen. Software die toch gebruikt wordt moet altijd up-to-date gehouden worden. Bovendien is het veiliger dat de crawler geen JavaScript interpreteert en geen gedownloade bestanden opslaat.

Er mag geen persoonlijke informatie (zoals IP-adressen) op de crawlserver aanwezig zijn. Op die manier zal geen persoonlijke informatie worden vrijgegeven als de crawlserver wordt gehackt. In dit project staat de crawlserver enkel SSH-verkeer toe vanaf de Lantern-API server, wat ervoor zorgt dat er geen persoonlijke IP-adressen aanwezig zijn in de logbestanden.

Daarnaast is het belangrijk dat gebruikelijke veiligheidsmaatregelen gerespecteerd worden. Enkele voorbeelden:

- Inloggen via SSH wordt enkel toegestaan met *key-based authentication*. Rechtstreeks inloggen op het rootaccount is niet toegestaan.
- De standaard SSH-poortnummer (22) kan worden vervangen door een willekeurig poortnummer. Dit kan doorgaans omzeild worden door middel van *port scanning*.
- De server kan software bevatten die een malware-infectie kan detecteren.
- Alle software op de crawler indien mogelijk compileren met alle protecties aan (o.a. *stack protection* en *buffer overflow protection*).

4.3.4 Conclusie

Hoewel het gevaar zeker niet geminimaliseerd mag worden, kan wel degelijk worden gesteld dat het onwaarschijnlijk is dat de persoonsgegevens zullen worden bemachtigd. Er werd eerder al vastgesteld dat het onwaarschijnlijk is dat het IP-adres van de server wordt gevonden. En net dat is noodzakelijk om verdere informatie over de crawler te bemachtigen.

Op de risicomatrix is het zeer onwaarschijnlijk dat de onderzoekers worden geïdentificeerd door een hidden service, de gevolgen zijn groot in dat geval.

4.4 Juridische vraagstukken

Aan het begin van het project waren er ook enkele juridische vraagstukken:

- De crawler is getuige van wapenhandel, mensenhandel, dealen van drugs, hacking... Is het strafbaar om dit waar te nemen en er een kopie van op te slaan?
- Wat met het auteursrecht? Is deze van toepassing op crawlers?

Michel Raes (informatieveiligheidsconsulent van de UGent) bracht mij in contact met een jurist aan de UGent. Doordat er geen tijd kon worden vrijgemaakt op de planning, is het juridische aspect niet verder onderzocht.

Hoofdstuk 5

Implemenatie van de crawler

Dit hoofdstuk bespreekt hoe de crawler werd geïmplementeerd.

5.1 Langdurige zoekopdrachten

Een conventionele zoekmachine – zoals Google – slaat de broncode op van webpagina's die de crawler tegenkomt. Daarnaast bouwt het een gigantische index op van alle woorden die het tegenkomt en de locatie ervan op elke pagina [33]. Samen met geavanceerde technieken kan het hiermee de meest geschikte webpagina's berekenen voor elke zoekopdracht. Dit resultaat moet zeer snel worden berekend omdat de gebruiker erop moet wachten. Een intern experiment bij Google toonde aan dat een vertraging van een halve second bij zoekopdrachten zorgde voor een daling van 20% in het zoekverkeer [34].

Stel dat er wordt gekozen om op soortgelijke manier te werken. Bij *threat intelligence* zullen de gebruikte zoekopdrachten niet vaak wijzigen en telkens voor een lange periode gebruikt worden. Als we bijvoorbeeld op zoek gaan naar vermeldingen van Universiteit Gent in het darkweb, dan zullen we hiervoor een zoekopdracht samenstellen en deze één keer uitvoeren. We willen echter op de hoogte worden gebracht als er nieuwe resultaten zijn. We gaan de zoekopdracht daarom elke dag herhalen. Daarbij wordt dezelfde zoekopdracht elke keer op de volledige index uitgevoerd. Dit is zinloos omdat de zoekopdrachten enkel uitgevoerd moeten worden op de nieuwe en gewijzigde pagina's, het resultaat van de andere pagina's kennen we namelijk al.

Een pagina kan enkel wijzigen in de index als de crawler een nieuwe versie van de pagina ophaalt. De crawler kan de zoekopdracht dus meteen uitvoeren na het downloaden van de pagina. Hierdoor is het niet meer nodig om een index bij te houden.

We maken dus gebruik van langdurige zoekopdrachten die worden opgeslagen. Zodra een zoekopdracht wordt toegevoegd, zal deze uitgevoerd worden op elke pagina die hierna gedownload wordt. Als de pagina voldoet, wordt deze pagina bij de resultaten opgeslagen. Het zijn dus de zoekopdrachten en de resultaten die opgeslagen worden, niet de index van alle pagina's.

Hierdoor zal een nieuwe zoekopdracht in het begin geen resultaten bevatten, zelfs al heeft de crawler ooit een pagina gedownload die aan de zoekopdracht zou voldoen. Dit is het enige grote nadeel.

5.1.1 Voordelen

Deze methode heeft ook enkele belangrijke voordelen:

- Er moet geen index van alle gedownloade pagina's bijgehouden worden. Dit bespaart veel opslagruimte en verlaagt de kostprijs.
- De implementatie is eenvoudiger omdat er geen gebruik moet worden gemaakt van de complexe technieken die het zoeken in de index in korte tijd mogelijk maken.
- De zoekopdrachten kunnen veel geavanceerder zijn zonder het geheel complexer te maken. Dit is mogelijk omdat de volledige broncode beschikbaar is bij het uitvoeren van de zoekopdrachten. Naast tekstopzoekingen kan er gebruik gemaakt worden van reguliere expressies en machine learning voor *sentiment analysis* en *text classification*.
- Een zoekopdracht kan voor lange periodes gebruikt worden. Hierbij kan een notificatie worden verzonden zodra een nieuw zoekresultaat is gevonden. Zo kan een bedrijf bijvoorbeeld op de hoogte worden gebracht als er een e-mailadres van een werknemer vermeld wordt.

5.2 De crawler

Voor er kan worden ingezoomd op de details van de crawler, is het belangrijk dat de algemene werking kort wordt beschreven zodat alle individuele onderdelen aan elkaar gelinkt kunnen worden.

De crawler bestaat zoals eerder vermeld uit verschillende goroutines. Goroutines zijn – zoals in Sectie 3.2 aangehaald – vergelijkbaar met threads, maar ze zijn minder belastend voor het systeem. Een goroutine voert een functie asynchroon uit tot deze is voltooid. Een goroutine in dit hoofdstuk zal dikwijls een oneindige lus uitvoeren waarin het wacht op bepaalde signalen. Hierna zal de term 'activeren' worden gebruikt om het opstarten van een nieuwe goroutine aan te duiden. De term 'afsluiten' zal worden gebruikt om het beëindigen van de lus binnenin de goroutine aan te duiden.

De belangrijkste klasse uit het systeem is de Crawler. Deze orkestreert het hele gebeuren en activeert de goroutines van de workers. Elke worker krijgt één hidden service aangewezen en zal enkel pagina's van die hidden service downloaden binnenin de toegewezen goroutine (maximaal één). Daarbij wacht elke worker een willekeurig aantal seconden tussen het downloaden van elk pagina. Op die manier zijn we zeker dat de crawler geen *denial-of-service-aanval* (DoS-aanval) uitvoert op servers aangezien er voor elke hidden service maximaal één worker actief kan zijn: alle aanvragen naar die website zullen dus altijd afgewisseld worden met een pauze.

Elke worker houdt wachtrijen bij van pagina's die hij nog moet downloaden. Als de wachtrijen van een worker leeg zijn, dan sluit de worker zijn goroutine af. Ook als een worker een maximaal aantal pagina's heeft gecrawld zal deze zichzelf afsluiten nog voor de wachtrijen leeg zijn. Als een worker zichzelf afsluit stuurt deze een signaal naar de crawler goroutine.

Het maximaal aantal pagina's die de goroutine van een worker mag downloaden wordt bepaald tijdens het activeren van de goroutine. Dit aantal is gerandomiseerd maar ligt binnen een opgelegd bereik en is niet constant omdat dit ondetecteerbaar moet zijn voor de hidden service. Een te hoge waarde zorgt ervoor dat het lang duurt voor andere workers aan de beurt komen. Een te lage waarde zorgt voor veel overhead vanwege het talrijk

activeren en afsluiten van goroutines. In de standaardconfiguratie ligt dit bereik op 10 tot 60 pagina's. Beide aantallen ontstonden uit het experimenteren en vergelijken van de snelheid.

De crawler goroutine voert continu een lus uit, daarin wacht hij op een signaal van een worker. Als een worker zichzelf afsluit kan de crawler een andere worker activeren. Hierbij wordt het aantal actieve goroutines aangepast naargelang het beschikbare geheugen op de server. Workers die geen lege wachtrijen hebben en dus nog pagina's te downloaden hebben, krijgen de status 'sleeping' en komen in de 'SleepingCrawlers' wachtrij van de crawler terecht. Zodra er plaatsen beschikbaar zijn activeert de crawler de eerste workers uit deze wachtrij.

5.3 De worker

Een worker houdt alle informatie bij in verband met één hidden service. Als we een worker activeren starten we een goroutine die de Run methode van deze worker uitvoert. Deze run methode is een lus die eerst luistert of de crawler goroutine geen signalen (via een threadsafe channel) naar de worker heeft gestuurd. Als dat niet zo is haalt de worker de volgende pagina die gedownload moet worden van de wachtrijen en downloadt die synchroon. Daarna wacht het een willekeurig aantal milliseconden (2000 - 6000) voor de lus opnieuw begint.

Doordat alle informatie slechts vanuit één goroutine wordt gebruikt, hebben we geen last van race conditions. De crawler weet op elk moment of de goroutine van een worker actief is omdat hij deze zelf opstart en een signaal krijgt als ze afgesloten worden. Tussen het afsluiten en het signaal van afsluiten is het nog mogelijk dat de crawler signalen naar de worker verstuurt (op dat moment denkt de crawler dat de worker actief is) terwijl hij hier niet meer naar luistert. Daarom handelt ook de crawler de nog niet verwerkte signalen naar de worker af zodra het afsluitsignaal bij de crawler binnen komt (terwijl hij die signalen zelf had verzonden).

De crawler weet dus wanneer het rechtstreeks informatie uit de worker mag adresseren en wijzigen of wanneer dit moet gebeuren via signalen die de worker in de goroutine

opvangt. Aangezien de crawler vanuit slechts één goroutine uitgevoerd wordt en enkel de crawler een worker kan starten is er ook dan geen gevaar voor race conditions. Naast het gebruik van channels voor de signalen tussen de goroutines maken we dus geen gebruik van synchronisatiehulpmiddelen.

5.3.1 Wachtrijen

Het is noodzakelijk dat we pagina's op een georganiseerde manier crawlen zodat nieuwe pagina's snel worden ontdekt en achteraf relatief snel worden gedownload. Om die reden gebruiken we meerdere wachtrijen. Bovendien houden we de introduction points bij, dat is een lijst met pagina's van de hidden service van de worker waarnaar gerefereerd wordt vanaf een andere hidden service. Deze pagina's wijzen we een diepte toe van nul. Alle andere pagina's van de worker krijgen een diepte gelijk aan het minimum aantal links die moeten worden aangeklikt om aan de pagina te geraken vanaf de introduction points.

De worker houdt 4 wachtrijen bij:

- **PriorityQueue**: een lijst met alle pagina's met diepte strikt kleiner dan drie.
- **Queue**: een lijst met pagina's die nog nooit gedownload zijn met een diepte hoger dan of gelijk aan drie.
- **LowPriorityQueue**: een lijst met pagina's die al eens gedownload werden met een diepte hoger dan of gelijk aan drie.
- **FailedQueue**: een lijst met pagina's waarvan de laatste downloadpoging is mislukt. De pagina's staan gesorteerd op het aantal (achtereenvolgende) mislukte downloadpogingen, van laag naar hoog. Deze lijst is niet geïmplementeerd met een prioriteitswachtrij maar bestaat uit een tabel van 10 wachtrijen. Als deze wachtrijen na elkaar worden geplaatst verkrijgen we de gesorteerde wachtrij. Een pagina waarvan het downloaden n keer is mislukt, bevindt zich in de n-de wachtrij. Er zijn ten slotte maar 10 verschillende waarden waarop gesorteerd kan worden: zodra een pagina meer dan 10 keer achtereenvolgens niet gedownload kan worden, wordt deze voorgoed genegeerd. Bij toevoegen moet de juiste wachtrij opgehaald worden uit de tabel. Vervolgens kan daaraan worden toegevoegd. Toevoegen heeft bijgevolg een

performantie van O(1). Bij verwijderen moeten maximaal 10 wachtrijen op het al dan niet leeg zijn worden gecontroleerd. Daarna wordt van de overige – niet lege – wachtrij verwijderd. Dit zorgt ook voor een performantie van O(1).

Pagina's worden opgeslagen in CrawlItem instanties en houden volgende informatie bij:

- De relatieve URL (zonder domeinnaam en protocol), bv. */hallo/pagina?id=23* of */hallo/pagina/43/*, merk hierbij op de de laatste schuine streep behouden blijft in tegenstelling tot de genormaliseerde URL in de hashmap.
- De diepte van de pagina
- **Cycle**: een getal dat helpt bij het berekenen van de diepte. Meer info volgt in Sectie 5.3.4.
- **Ignore**: boolean die bijhoudt of deze pagina moet worden genegeerd. Het bestandstype (bv. HTML, PDF, PNG en JPEG) van een URL kan pas na de eerste download gedetecteerd worden. Als het gedownloade bestand geen HTML-bestand is, moet de worker dat onthouden om volgende downloads te voorkomen. Andere bestandstypes worden namelijk niet ondersteund waardoor het downloaden ervan nutteloos is.
- **FailCount**: het aantal keer dat het downloaden van deze pagina achtereenvolgens is mislukt. Bij een succesvolle download wordt dit terug gelijkgesteld aan 0.
- **LastDownloadStarted**: bevat het tijdstip waarop de laatste download van deze URL is gestart. Bij een succesvolle download wordt deze waarde gewist (dit is een pointer) om opslagruimte te sparen (zie Sectie 5.3.6).
- **LastDownload**: het tijdstip van de laatste succesvolle download.
- **Queue**: de queue waarin dit item zich momenteel bevindt.
- **Subdomain**: referentie naar het subdomein binnen deze hidden service.

Daarnaast houdt de worker ook een hashmap bij van alle CrawlItems (pagina's) van die hidden service, geïndexeerd op hun genormaliseerde URL: de URL zonder de domeinnaam. Als het pad van de genormaliseerde URL op een schuine streep eindigt, wordt dit laatste

karakter ook verwijderd (URL's met of zonder schuine streep zijn daarna gelijk). Merk op dat de genormaliseerde URL enkel de string is die als sleutel wordt opgeslagen in de hashmap, deze URL zal nooit in een aanvraag worden verstuurd. Zo kan de worker voor een gegeven URL het reeds aangemaakte CrawlItem vinden. Als de URL niet kan worden gevonden is het een pagina die nog nooit werd gevonden en wordt een nieuw CrawlItem aangemaakt. De worker houdt hiervoor een hashmap bij per subdomein van de hidden service, deze heten de AlreadyFound hashmaps.

Als de worker een pagina wil downloaden geeft hij voorrang aan items uit de FailedQueue met een FailCount kleiner dan twee. Daarna volgt de PriorityQueue, de Queue en de LowPriorityQueue. Als die allemaal leeg zijn worden de overige items (FailCount > 1) uit de FailedQueue nogmaals geprobeerd. Op die manier krijgen pagina's waarvan het downloaden één keer mislukt is nog voorrang op alle andere pagina's.

De FailedQueue voegt hierbij nog extra logica toe. Items worden pas vanaf de FailedQueue vrijgegeven als de tijd tussen de vorige mislukte poging groot genoeg is. Hiervoor maakt de FailedQueue gebruik van het LastDownloadStarted-kenmerk van een item. Aangezien items altijd chronologisch (op het tijdstip van het mislukken) aan de FailedQueue worden toegevoegd, moet enkel het eerste item van elke wachtrij (de FailedQueue bevat er 10, zoals eerder vermeld) gecontroleerd worden. Als dat item niet voldoet wordt die wachtrij – niet de FailedQueue zelf – als leeg beschouwd en wordt de volgende wachtrij gecontroleerd.

5.3.2 Downloaden van pagina's

Eens de volgende URL uit de wachtrij is gehaald zal die worden klaargemaakt om gedownload te worden. Hierbij worden exact dezelfde HTTP-headers doorgestuurd als de Tor browser. Op die manier valt de crawler niet op tussen het andere verkeer van de hidden service. Alle Tor gebruikers sturen namelijk dezelfde headers door. Het niet verzenden van dezelfde headers zou de crawler meteen doen opvallen en onderscheidbaar maken. Dan zou een hidden service makkelijk het gedrag van de crawler kunnen analyseren.

Om het verkeer over Tor te sturen wordt gebruik gemaakt van de SOCKS5-proxy die de Tor applicatie zelf ter beschikking stelt [15]. Na eigen tests bleek het efficiënter om meerdere Tor instanties gelijktijdig te starten. Elke worker krijgt bij het activeren cyclisch

een bepaalde proxy-instantie toegekend. Meerdere workers kunnen gebruik maken van dezelfde proxy, maar een worker maakt niet noodzakelijk elke keer gebruik van dezelfde proxy.

Eens de HTTP(S)-verbinding met de hidden service voor een bepaalde URL tot stand is gebracht kijken we de statuscode na. Bij ongebruikelijke statuscodes (buiten 200 - 299) wordt het downloaden meteen afgebroken. Afhankelijk van de status code wordt de pagina voorgoed genegeerd (bv. bij 404 not found) of gemarkeerd als mislukt. Redirects worden automatisch gevolgd door de standaard HTTP-bibliotheek van Go. Als het aantal redirects groter is dan 10 zal de verbinding worden afgebroken en wordt de URL gemarkeerd als te negeren.

Hierna controleren we of de *Content-Length* antwoordheader niet groter is dan 2 MB. In dat geval annuleren we het downloaden meteen en markeren we de URL als te negeren. Het is echter mogelijk dat deze header niet is gegeven of foutief is, daarom beperken we het lezen tot 2 MB. Indien ergens in de verwerking deze grens wordt overschreden, wordt het downloaden geannuleerd en de URL gemarkeerd als te negeren.

De reden dat voor 2 MB wordt gekozen is omdat de bestandsgrootte van de meeste darkweb pagina's onder die grens ligt (uit tests blijkt dit in de buurt van 50 KB te liggen). Deze grens mag bovendien niet te hoog liggen omdat grote bestanden het systeem belasten. Het opslaan van de broncode van te grote HTML-pagina's (bij het vinden van een resultaat) is bovendien onmogelijk in MongoDB omdat deze een maximale documentgrootte van 16 MB hanteert [35]. Ik ga hier uit van de veronderstelling dat pagina's groter dan 2 MB niet waardevol genoeg zijn om het systeem te belasten.

Om te beginnen worden de eerste 512 bytes van de pagina gelezen. Aan de hand van die eerste bytes wordt bepaald of de pagina een HTML-bestand is. Hiervoor wordt gebruik gemaakt van de standaard HTTP-bibliotheken van Go. Deze baseert zich hiervoor op de *MIME sniffing* standaard [36]. Deze bibliotheek gebruikt maximaal 512 bytes om het bestandstype te detecteren. Alle niet HTML-pagina's worden meteen gemarkeerd als te negeren en het downloaden wordt in dat geval onderbroken.

Daarna wordt de rest van het bestand ingelezen. Het is mogelijk dat hierbij een timeout optreedt omdat het downloaden te lang duurt. Dat kan het geval zijn als het netwerk of

de server wordt overbelast. Deze timeouts registreren we en deze bepalen mee het aantal workers dat tegelijk actief kan zijn.

5.3.3 Het verwerken van gedownloade pagina's

Als het downloaden van de pagina met diepte gelijk aan nul slaagt, wordt deze toegevoegd aan de IntroductionPoints (een lijst van links waarnaar verwezen wordt vanaf andere hidden services) op voorwaarde dat het aantal IntroductionPoints niet groter is dan 9. Anders wordt enkel de hoofdpagina met relatieve URL '/' toegelaten als 10e IntroductionPoint. Deze maximale limiet wordt opgelegd om het aantal introduction points te beperken. Een URL kan onmogelijk dubbel in de IntroductionPoints terecht komen, aangezien een pagina na een succesvolle download nooit meer in de wachtrijen kan terecht komen voordat de IntroductionPoints worden leeg gemaakt (hierover later meer).

Het HTML-bestand wordt verwerkt zonder de volledige *Document Object Model* (DOM) op te bouwen. De Go standaard bibliotheek voorziet hiervoor methodes die ervoor zorgen dat we niet de hele DOM in het geheugen moeten houden. Events worden geregistreerd tijdens het lezen. Hieruit halen we alle <a> tags. Bovendien lezen we de <title> tag in. Alle tekst die we tegenkomen in de <body> tag, en niet in <script>, <noscript> of <style> tags, wordt eerst omgezet in kleine letters en weggeschreven naar een lijst van bytes.

Op deze lijst worden de zoekopdrachten uitgevoerd. Aangezien het aantal zoekopdrachten groot is, wordt een suffixtabel aangemaakt zodra één van de zoekopdrachten gebruik maakt van tekstopzoekingen (reguliere expressies maken enkel gebruik van de suffixtabel als die al bestaat en als de reguliere expressie met een vaste reeks karakters begint). Dit versnelt het opzoeken aanzienlijk. Indien er resultaten gevonden worden, worden deze doorgegeven via de RESTful API naar de MongoDB database, dit wordt besproken in Hoofdstuk 6. Hierbij wordt o.a. de titel, URL, broncode en een kort uittreksel (snippet) van wat de zoekopdrachten precies hebben gevonden, verzonden.

Alle geldige URL's worden uit het 'href' attribuut van de <a> tags gehaald. Het fragmentgedeelte van de URL's (deel na #) wordt verwijderd omdat die niet wordt meegestuurd

in HTTP-aanvragen, zoals opgelegd door de HTTP-specificaties [37]. Het is bijgevolg zinloos om deze bij te houden als het resultaat van de aanvragen er niet van afhangt.

Relatieve URL's worden eerst samengevoegd met de URL van de gedownloade pagina tot een absolute URL. Hierna worden volgende zaken voor de URL's gecontroleerd. URL's die niet voldoen aan één van de voorwaarden worden genegeerd.

- Het protocol van de URL moet gelijk zijn aan HTTP of HTTPS.
- De domeinnaam van de URL moet uit minstens één punt bestaan. Hierdoor kan er onderscheid worden gemaakt tussen een top-level-domeinnaam (TLD), een second-level-domeinnaam en eventueel bijkomende subdomeinnamen. Een domeinnaam in Tor bestaat namelijk altijd uit een second-level-domeinnaam gevolgd door de 'onion' TLD.
- De TLD van de URL moet gelijk zijn aan 'onion'.
- De lengte van de second-level-domeinnaam moet gelijk zijn aan 16 (de base32-codering van de eerste 80 bits van de SHA1 hash van de publieke sleutel) karakters. Ik heb gemerkt dat een aantal websites in het darkweb verwijzen naar andere hidden services door gebruik te maken van domeinnamen met een lengte groter dan 16 karakters. Hierbij zijn ongeldige karakters zoals het liggend streepje toegevoegd. Het klikken op deze URL's in de Tor-browser, zorgt ervoor dat de browser deze ongeldige karakters eerst verwijdert. Om dit gedrag te simuleren verwijdert de worker alle karakters niet in het het bereik van een base32-codering (A-Z of 2-7). De lengte van het second-level-domeinnaam moet nu gelijk zijn aan 16.

Hierna volgen twee speciale gevallen: ofwel verwijst de URL naar een pagina binnen de hidden service van de worker (interne link) ofwel verwijst deze naar een andere hidden service (externe link).

De situatie voor interne links is erg eenvoudig. Aangezien een actieve worker een alleenrecht heeft op haar wachtrijen (threadsafe), kan deze rechtstreeks haar NewReference methode uitvoeren voor elke gevonden interne link. Deze methode maakt indien nodig de CrawlItem instantie (met informatie over de pagina) aan en voegt deze in de juiste wachtrij toe (hierover later meer in Sectie 5.3.5).

Externe links worden door de worker doorgestuurd naar de crawler over een channel. Hierbij worden ze eerst samengevoegd in een lijst. Zodra de crawler naar het betreffende channel kan luisteren, verwerkt hij al deze URL's in één keer. Het kan echter voorkomen dat een andere worker reeds externe links naar de crawler had verzonden via die channel zonder dat de crawler tijd kreeg om deze in te lezen en te verwerken. In die situatie zou het schrijven naar de channel ervoor zorgen dat de goroutine van waaruit geschreven wordt, tijdelijk wordt geblokkeerd tot de channel terug leeg is.

Om dat te voorkomen probeert de worker eerst de channel in te lezen (zonder te blokkeren), als dat lukt voegen we de reeds aanwezige lijst samen. Daarna probeert de worker de eventueel samengevoegde lijst te schrijven naar de channel (zonder te blokkeren). Dit wordt herhaaldelijk toegepast tot schrijven lukt zonder te blokkeren.

Zodra de crawler naar de betreffende channel kan luisteren, verwerkt hij al deze URL's in één keer. De crawler vindt de bijhorende worker van elke URL aan de hand van een hashmap en het domein van de gegeven URL. De crawler maakt eventueel een nieuwe worker aan als de hidden service nog niet gekend is. Dan voegt hij deze worker ook toe aan de hashmap.

Als de worker in kwestie actief is, verstuurt de crawler de URL via een channel. Als de worker niet actief is, kan de crawler rechtstreeks de NewReference methode uitvoeren van de worker, want dan is dit threadsafe. De NewReference methode kan de wachtrijen van de worker aanpassen, daarom is het mogelijk dat de worker nu bereid is om terug actief te worden als hij dat eerder niet was. De voorwaarden worden terug gecontroleerd en eventueel wordt de worker op de SleepingWorkers wachtrij geplaatst. Zodra alle URL's verwerkt zijn, worden de eerste workers uit SleepingWorkers geactiveerd als het aantal actieve workers laag genoeg is. Indien er geen workers kunnen worden geactiveerd zal dit enkel gebeuren nadat één van de actieve workers zichzelf afsluit of als het maximum aantal actieve workers wordt verhoogd.

5.3.4 Crawlcyclus

Het is belangrijk dat na verloop van tijd de reeds gecrawlde URL's opnieuw worden gecrawld. Op die manier worden nieuwe URL's en aangepaste inhoud ontdekt. Daarom

worden de IntroductionPoints, een lijst van de hoofdpagina's van een hidden service, 12 uur nadat de eerste erin werd geplaatst, terug in de PriorityQueue van de worker geplaatst. Hierbij verhoogt de worker het cyclusgetal dat bijhoudt hoeveel keer dit proces werd gestart. Zodra het downloaden van één van de introduction points lukt, wordt deze terug in de IntroductionPoints lijst toegevoegd zodat het proces elke 12 uur kan herhaald worden. IntroductionPoints die niet meer geldig zijn worden op die manier ook verwijderd omdat het downloaden niet zal slagen.

De IntroductionPoints krijgen meteen voorrang bij het downloaden omdat ze in de PriorityQueue staan. Alle URL's tot op diepte twee worden hierna ook nog toegevoegd aan de PriorityQueue. Alle nieuwe URL's met hogere diepte komen terecht in de Queue. URL's die al eens werden gevonden met hogere diepte komen terecht in de LowPriotityQueue.

Op deze manier worden nieuwe URL's op de pagina's met lage diepte snel ontdekt en gedownload. Pagina's met hogere diepte worden op termijn ook herhaaldelijk gedownload, maar deze krijgen geen voorrang.

Meestal is het zo dat nieuwe URL's op de startpagina van een website komen te staan, of in ieder geval in één van de pagina's die bereikbaar zijn vanaf de startpagina. Een webshop zal de laatste nieuwe producten op de startpagina plaatsen, een nieuwswebsite, forum of blog doet hetzelfde voor nieuwe berichten. Dit maakt de methode erg efficiënt, ook in het darkweb.

5.3.5 Het verwerken van een link (NewReference methode)

Bij het verwerken van een link is er sprake van een bron en een bestemming. De bron is de pagina die verwees naar een andere pagina, de bestemming.

De worker zal voor de URL van de bestemming, eerst het domein en de relatieve URL van elkaar afsplitsen. Het domein zal gebruikt worden om het subdomein van de hidden service te bepalen. Een overbodig vraagteken wordt op het einde van de relatieve URL verwijderd omdat deze geen invloed heeft op het resultaat aangezien de querystring dan leeg is. De relatieve URL wordt vervolgens genormaliseerd zoals eerder besproken.

Aan de hand van het subdomein en de genormaliseerde URL zoekt de worker in de AlreadyFound hashmap van het subdomein op of deze al gevonden is. De reeds aangemaakte CrawlItem is beschikbaar als deze al gevonden was. In het andere geval maken we de CrawlItem aan en voegen we deze toe aan de AlreadyFound hashmap met de genormaliseerde URL als sleutel.

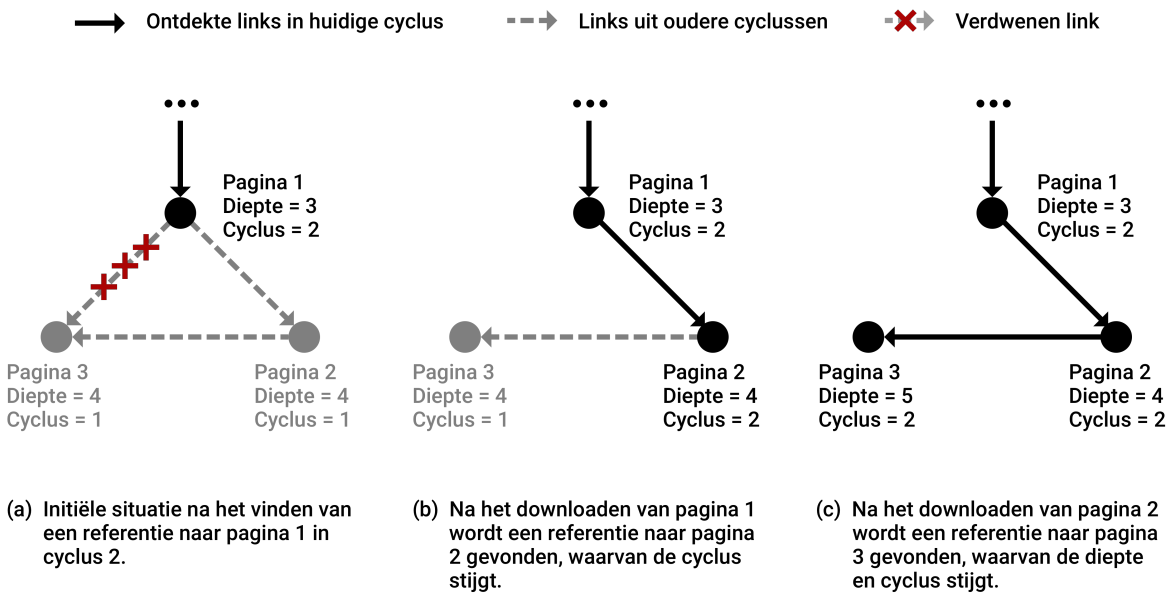
Als de gevonden CrawlItem gemarkeerd werd als te negeren, breken we de methode af zonder de wachtrijen aan te passen.

Hierna start de berekening van de paginadiepte. Elke nieuwe cyclus herberekent de dieptes van de pagina's vanaf de IntroductionPoints. De diepte die in oudere cyclussen berekend werd, wordt genegeerd. De diepte begint bij nul voor de IntroductionPoints. Voor alle pagina's die vanaf dan worden gevonden stijgt de diepte telkens per tussenliggende pagina. Alle links met een bron van een ander domein, krijgen sowieso een diepte gelijk aan nul (deze kan worden toegevoegd aan de IntroductionPoints als het downloaden later slaagt). Anderzijds kan een interne verwijzing worden onderverdeeld in vier speciale gevallen:

- De bestemming werd nooit eerder gevonden (nieuw CrawlItem): de diepte van de bestemming wordt gelijkgesteld aan die van de bron plus één. Hierna wordt het cyclusgetal van de bestemming gelijkgesteld aan die van de bron.
- De bestemming werd al eens gevonden en heeft:
  - **een lager cyclusgetal dan de bron**: de diepte van de bestemming wordt gelijkgesteld aan die van de bron plus één. Hierna wordt het cyclusgetal van de bestemming gelijkgesteld aan die van de bron. De diepte van de bestemming wordt hier genegeerd omdat de diepte enkel per cyclusgetal wordt berekend, hierbij geven we voorrang aan hogere cyclusgetallen.
  - **hetzelfde cyclusgetal dan de bron**: de diepte van de bestemming wordt gelijkgesteld aan die van de bron plus één **als dit kleiner is dan de diepte van de bestemming**.
  - **een hoger cyclusgetal dan de bron**: geen aanpassingen.

De eerder ingestelde diepte wordt genegeerd bij verwijzingen vanaf een nieuwe cyclus. Hierdoor kan de diepte van een pagina stijgen als een link naar een pagina verdwijnt. Dit maakt de diepteberekening krachtig maar eenvoudig.

Figuur 5.1 illustreert hoe de diepte van een pagina wordt gecorrigeerd na het verdwijnen van een link. Dit verloopt in drie stappen waarbij telkens een pagina gedownload wordt.



Figuur 5.1: Aanpassing van diepte bij verdwenen links.

Als een pagina tijdelijk niet beschikbaar is, is het mogelijk dat hierdoor de diepte van verschillende pagina's tijdelijk verhoogt. Zodra de pagina terug beschikbaar is en gedownload wordt, wordt de diepte op termijn gecorrigeerd.

Als de worker een nieuwe crawlcyclus start, zullen alle URL's geleidelijk aan terug naar die hogere cyclus overgaan zolang er een nieuwe link naar elke pagina wordt gevonden.

Nadat de diepte van de pagina werd aangepast in de NewReference methode, moet de pagina in de juiste wachtrij worden geplaatst. Hierbij kijken we eerst na of de pagina niet al in een wachtrij aanwezig was. Bij het wijzigen van de diepte is het ook mogelijk dat een pagina van de ene wachtrij naar de andere moet overschakelen. Om die reden ondersteunen de wachtrijen ook verwijderoperaties op specifieke pagina's.

5.3.6 Geheugenbeheer

Het opslaan van de AlreadyVisited hashmaps en de CrawlItems in het geheugen kan ervoor zorgen dat er al snel te weinig geheugen beschikbaar is en dat de crawler wordt afgesloten. Om dat te voorkomen zullen enkel actieve workers in het geheugen worden gehouden. Alle andere workers worden weggeschreven naar een bestand.

Inkomende links naar workers die niet in het geheugen staan, zullen tijdelijk worden opgeslagen in een lijst per worker. Als die lijst te groot wordt, zal de worker in de SleepingWorkers wachtrij geplaatst worden als dat nog niet zo was. Zodra de worker dan terug geactiveerd wordt en in het geheugen terecht komt verwerken we al de inkomende links.

Elke minuut wordt bovendien het aantal beschikbare actieve workers verminderd als er te veel geheugen gebruikt wordt. Het omgekeerde geldt ook. Daarnaast voorzien we swapruimte op de server om tijdelijke pieken te overbruggen.

5.3.7 Data opslag

Als de crawler stopt is het noodzakelijk dat alle gekende hidden services en hun URL's behouden blijven tot de crawler opnieuw start. Bij het afsluiten van de crawler schrijft de crawler alle workers weg naar een bestand als die nog in het geheugen zaten. Alle workers die niet actief zijn, zijn al weggeschreven naar een bestand (zie Sectie 5.3.6). De niet actieve workers kunnen wel een lijst bevatten met nog niet verwerkte links. In dat geval wordt de worker in kwestie terug in het geheugen geladen waarna deze lijst kan worden verwerkt (met de NewReference methode). Daarna worden de bijgewerkte workers opnieuw naar een bestand weggeschreven.

Bij het opstarten van de crawler worden alle wachtrijen en CrawlItems uit de bestanden gelezen en kan er weer verder gegaan worden.

5.3.8 Onbereikbare hidden services

In het darkweb is het gebruikelijk dat hidden services niet 24 op 24 uur bereikbaar zijn. Vaak zijn hidden service slechts voor korte tijd bereikbaar en verdwijnen ze daarna voorgoed. Daarom is het belangrijk dat hidden services worden genegeerd na een te groot aantal mislukte downloads.

Als 30 opeenvolgende downloads mislukken (pagina's die al eens mislukt zijn tellen niet mee) verhogen we de FailStreak van een worker. Daarna blijft de worker voor minimaal 6 uur inactief. Er werd voor het getal 40 gekozen omdat er veel timeouts ($\pm$ 1 per seconde) voorkomen voor beschikbare websites. Dit komt door de vele netwerkaanvragen die het netwerk overbelasten. Het aantal timeouts wordt elke minuut gecontroleerd zodat het maximaal aantal actieve workers daaraan kan worden aangepast. Door deze timeouts is er een redelijke kans dat het downloaden van pagina's een aantal keer na elkaar mislukt. Om zeker te zijn van de onbeschikbaarheid ligt dit getal (30) redelijk hoog.

Zodra een download slaagt, wordt de FailStreak terug op nul gezet. Als er weer 30 opeenvolgende downloads mislukken, wordt de FailStreak verhoogd. Deze keer wacht de worker 4 keer zo lang (24 uur). Dit gaat zo exponentieel verder, met een wachttijd gelijk aan $FailStreak^2 \times 6$ uur. Alle inkomende links worden tijdens die wachttijd genegeerd.

5.3.9 Zoekopdrachten

Er zijn 4 basisoperaties voor het zoeken. Geen enkele operatie is hoofdlettergevoelig.

- **Reguliere expressie**: Deze geeft true terug als de reguliere expressie ergens overeenkomt in de opgegeven tekst.
- **Tekst**: De 'tekst' basisoperatie bevat een string. Als deze string gevonden wordt in de tekst, en deze omvat enkel volledige woorden, geeft deze true terug. Zoeken op 'steen' zal dus niet overeenkomen met een tekst die 'schoorsteen' bevat.
- **Lijst**: De lijst basis operatie is hetzelfde als de 'tekst' basisoperatie, maar zoekt naar verschillende strings tegelijkertijd.

- **Operator**: De 'operator' basisoperatie bevat een linker en een rechter basisoperatie (en kan dus weer een 'operator' basisoperatie bevatten). Daarnaast bevat hij een operator die de waarden 'AND' of 'OR' kan aannemen. Voor 'AND' wordt true teruggegeven als beide basisoperatie true teruggeven. Voor 'OR' wordt true teruggegeven als minstens één van beide true terug geeft. Hierbij worden zo weinig mogelijk basisoperaties uitgevoerd om tot een resultaat te komen.

Elke zoekopdracht (query) houdt één van die basisoperaties bij. Door middel van de 'operator' basisoperatie kunnen meerdere operaties gecombineerd worden tot één complexe zoekopdracht. Deze kan voorgesteld worden als een boom, wat in de front-end gebruikt wordt om een zoekopdracht te bouwen (zie Hoofdstuk 7). Na het downloaden van een pagina zal deze worden toegepast op de tekst op die pagina.

Als een zoekopdracht een positief resultaat terug geeft, geeft deze ook een kort uittreksel van de tekst waarin het resultaat werd teruggevonden. Dit heet een snippet. Om dit te genereren geven de basisoperaties een lijst met de posities terug waarin ze hun resultaat vonden. De zoekopdracht voegt deze samen tot één snippet, net zoals andere zoekmachines.

5.4 Testen

Nog een voordeel van het gebruik van Go als programmeertaal is dat het standaard automatische tests ondersteunt. Hiervoor bevat het project bestanden van de vorm klassenaam_test.go. Elk van deze bestanden test de bijhorende klasse. Testen kan met het commando 'go test'. De 8 belangrijkste klassen en de datastructuren worden getest met deze bestanden. In totaal zijn er 22 klassen en datastructuren.

Naast de unit tests werd de crawler ook gewoon opgestart om deze te testen. Omdat het nogal onveilig is om rechtstreeks op het darkweb te testen bevat de crawler een configuratiebestand waarmee crawlen via Tor kan worden uitgeschakeld. De crawler werkt dan in het clearweb. De eerste tests op Tor werden uitgevoerd in een Docker container.

5.5 Online plaatsen

De crawler maakt gebruik van de Service bibliotheek (github.com/kardianos/service) voor Go. Deze zorgt ervoor dat de crawler kan worden geïnstalleerd als een daemon op verschillende besturingssystemen via hun opstart-daemons. Hierdoor start de crawler automatisch op bij het heropstarten van de server of bij het crashen van de crawler. Enkel de launchd (macOS) en systemd (Linux) implementaties werden getest en gebruikt door dit project.

Om de crawler op de server te plaatsen wordt gebruik gemaakt van Fabric, een open source Python bibliotheek die dit vereenvoudigt. In fabfile.py wordt de code specifiek voor het online plaatsen van de crawler geplaatst. Met behulp van het 'fab deploy' commando treedt deze code in werking. Tijdens dit proces worden de Go bestanden gecompileerd naar de AMD64 architectuur voor Linux, hiervoor hoeft Go niet geïnstalleerd te zijn op de server.

Als de crawler gecompileerd is wordt de crawler die actief is op de server uitgeschakeld. Daarna wordt het uitvoerbare bestand via het rsync-commando overgezet naar de /etc/lantern map. Hierna wordt de crawler daemon terug ingeschakeld.

De regels van de iptables firewall werden ingesteld met behulp van een shellscript.

5.6 Activiteiten

De ontwikkeling van de crawler nam ongeveer 9 weken in beslag. Dit is niet inclusief de week die daarin voorafging waarin geëxperimenteerd werd met verschillende mogelijke uitwerkingen (zoals eerder werd besproken in Sectie 2.4.1).

Enkele statistieken:

- 5.106 lijnen Go-code.
- ± 40 Go-klassen.
- 38 lijnen Python-code.

Hoofdstuk 6

Implementatie van de RESTful API

Er werd gekozen om een microservice te implementeren die de toegang tot de MongoDB database regelt via een RESTful API. Die MongoDB houdt alle zoekopdrachten bij en hun resultaten. De reden waarom voor een microservice gekozen werd is omdat ook de front-end applicatie toegang moet hebben tot de informatie in de databank, en dat kan enkel via HTTP. MondoDB voorziet geen RESTful api, hoewel er enkele open source alternatieven voor bestaan.

Doordat de microservice de toegang tot de databank bepaalt kan functionaliteit die enkel verband houdt met de dataopslag afgezonderd blijven van de andere applicaties. Een voorbeeld daarvan is dat een resultaat voor een gevonden zoekopdracht niet noodzakelijk moet leiden tot de opslag van een extra resultaat. Het is namelijk mogelijk dat er reeds een resultaat werd gevonden op die URL voor die zoekopdracht.

6.1 Datastructuur

MongoDB maakt gebruik van de Binary JavaScript Object Notation (Binary JSON, BSON) representatie bij het opslaan van documenten [38]. Op zich maakt dit niet veel uit aangezien de omzetting naar BSON volledig geautomatiseerd wordt door de open source 'bson' bibliotheek (gopkg.in/mgo.v2/bson), deze is een onderdeel van de 'mgo' bibliotheek (gopkg.in/mgo.v2), de MongoDB driver voor Go die gebruikt wordt door de microservice. Beide bibliotheken zijn open source.

Voor de communicatie tussen de RESTful API en de applicaties wordt de JavaScript Object Notation (JSON) representatie gebruikt. De belangrijkste reden hiervoor is dat BSON en JSON erg op elkaar lijken. Dat is eenvoudiger dan met XML en BSON te werken. BSON wordt niet gebruikt omdat browsers deze niet kunnen inlezen in JavaScript.

Het omzetten van Go instanties in JSON en omgekeerd verloopt bijna volledig automatisch met de standaard JSON-bibliotheek van Go. Enkel voor de zoekopdrachten was extra code noodzakelijk (zie Sectie 6.1.1).

Voor de beschikbare endpoints worden besproken, is het belangrijk om te kijken welke datastructuren er precies worden opgeslagen en wat hun belangrijkste eigenschappen zijn.

6.1.1 Zoekopdracht (Query)

Een zoekopdracht specificeert wanneer resultaten moeten worden gegenereerd. Dit is een complexe datastructuur om naar JSON en van JSON om te zetten. Dit komt omdat het Query-kenmerk de basisoperatie uit het vorige hoofdstuk bevat. Doordat dit een interface is weet de JSON bibliotheek niet om welke specifieke klasse het gaat. Hiervoor werden de Unmarshal-functies (omzetting van JSON naar objecten) overschreven en het 'type'-kenmerk toegevoegd aan de JSON representatie van de basisoperaties.

Een zoekopdracht bevat volgende eigenschappen:

- **Id**: de identificatie die MongoDB voor dit object heeft gegenereerd.
- **Name**: de gekozen naam voor deze zoekopdracht.
- **CreatedOn**: het tijdstip waarop deze zoekopdracht werd aangemaakt.
- **Query**: bevat één van de basisoperaties (zie Sectie 6.1.1).

6.1.2 Resultaat (Result)

De resultaten zijn één van de belangrijkste datastructuren. Er bestaat maximaal één resultaat voor de combinatie van een URL en een zoekopdracht.

Uit de praktijk blijkt dat dezelfde tekst vaak voorkomt op verschillende URL's binnenin één hidden service. Dat zorgt ervoor dat een zoekopdracht veel resultaten genereert voor hetzelfde stukje tekst. Zo herhaalt een webshop bijvoorbeeld op verschillende plaatsen de titel van een bepaald artikel. Zoeken op tekst die in die titel voorkomt zou veel verschillende overbodige resultaten opleveren. Daarom wordt bij het toevoegen eerst gecontroleerd of er al dan niet een resultaat bestaat met dezelfde host, zoekopdracht en snippet. Indien die al bestaat voegen we de URL gewoon doe aan de 'Urls'-eigenschap van het bestaande resultaat.

Omdat het aantal verschillende URL's met dezelfde snippet en host uit de hand kan lopen, wordt een limiet van 10 verschillende URL's gehandhaafd. Nieuwe URL's worden daarna genegeerd.

Beide aanpassingen zorgen ervoor dat de databank niet te groot wordt en dat dubbele resultaten als één worden weergegeven.

Een resultaat bevat o.a. volgende eigenschappen:

- **Id**: het ID dat door MongoDB werd gegenereerd voor dit resultaat.
- **QueryId**: het ID van de hierbij horende zoekopdracht.
- **Url**: de URL waarop dit resultaat het eerst werd gevonden voor deze zoekopdracht.
- **Body**: de HTML-code van de pagina op het moment dat de zoekopdracht het laatst werd gevonden op deze pagina.
- **Title**: de tekst uit de <title> tags van de pagina.
- **Host**: de host van de URL waarop het resultaat werd gevonden.
- **Snippet**: een klein uittreksel uit de pagina rond de plaats waar de zoekopdrachten 'matchten'.
- **Category**: een optionele string waarmee resultaten in verschillende categorieën kunnen worden gestoken om ze later sneller terug te vinden.
- **Urls**: een lijst met maximaal 10 verschillende URL's waarop hetzelfde resultaat gevonden werd voor deze zoekopdracht (de snippet was gelijk).

Doordat de volledige HTML-broncode van een pagina wordt opgeslagen bij het resultaat kan deze aan de gebruiker worden getoond. De gebruiker moet de URL dan niet handmatig opzoeken. Dit is belangrijk voor de veiligheid aangezien enkel de server van de crawler in contact mag komen met het darkweb. De risicoanalyse houdt namelijk rekening met de blootstelling van het IP-adres van de server, niet het IP-adres van een gebruiker (dan zijn de gevolgen ernstiger). Meer informatie over hoe er met deze broncode wordt omgegaan volgt in Hoofdstuk 7.

Eens een interessant resultaat wordt gevonden, moet dit kunnen worden gemarkeerd zodat het later sneller terug kan worden gevonden. Daarom werd het 'Category'-kenmerk toegevoegd. Hierdoor is het mogelijk om resultaten met een bepaalde categorie op te vragen. Resultaten met een categorie kunnen bovendien visueel worden aangeduid in de front-end applicatie.

6.1.3 Gegroepeerde resultaten (AggregatedResult)

Soms kan een bepaalde zoekopdracht veel verschillende resultaten (met een verschillende snippet) voor dezelfde host veroorzaken. Dit zorgt ervoor dat de resultaten onoverzichtelijk worden, de resultaten die behoren tot een andere host en in mindere mate voorkomen zijn bijna niet meer terug te vinden. Om dit op te lossen werden gegroepeerde resultaten geïntroduceerd. Dit is een tijdelijke onderverdeling die resultaten per host groepeert. De AggregatedResults voor een bepaalde zoekopdracht geven dus aan voor welke hosts er resultaten beschikbaar zijn.

Een AggregatedResult bevat o.a. volgende eigenschappen:

- Id: de host.
- Count: het aantal resultaten die opgeslagen zijn voor deze host en zoekopdracht.

Deze datastructuur wordt niet opgeslagen maar enkel gegenereerd uit de databank door MongoDB instructies. De resultaten die horen bij deze host en zoekopdracht worden zelf niet opgeslagen in een AggregatedResult omdat deze apart moeten worden opgevraagd.

6.2 Beschikbare endpoints

De RESTful API is beschikbaar op verschillende relatieve URL's op de domeinnaam van de API. De relatieve URL en de benodigde HTTP-methode die nodig zijn om een bepaalde functionaliteit te bekomen heet een endpoint.

Om de aanvragen te verwerken heeft ook Go een handige standaardbibliotheek (net/http). De microservice gebruikt deze in combinatie met de open source Gorilla bibliotheek (github.com/gorilla/mux). Gorilla verzorgt de routing van de API en zorgt ervoor dat een endpoint die aan bepaalde voorwaarden voldoet (bv. URL = /api/test en methode = POST) aan een functie kan worden gekoppeld. Bovendien kunnen we parameters in de URL vermelden door deze tussen accolades te plaatsen. De waarde van deze parameters kunnen worden ingelezen bij de verwerking van een aanvraag.

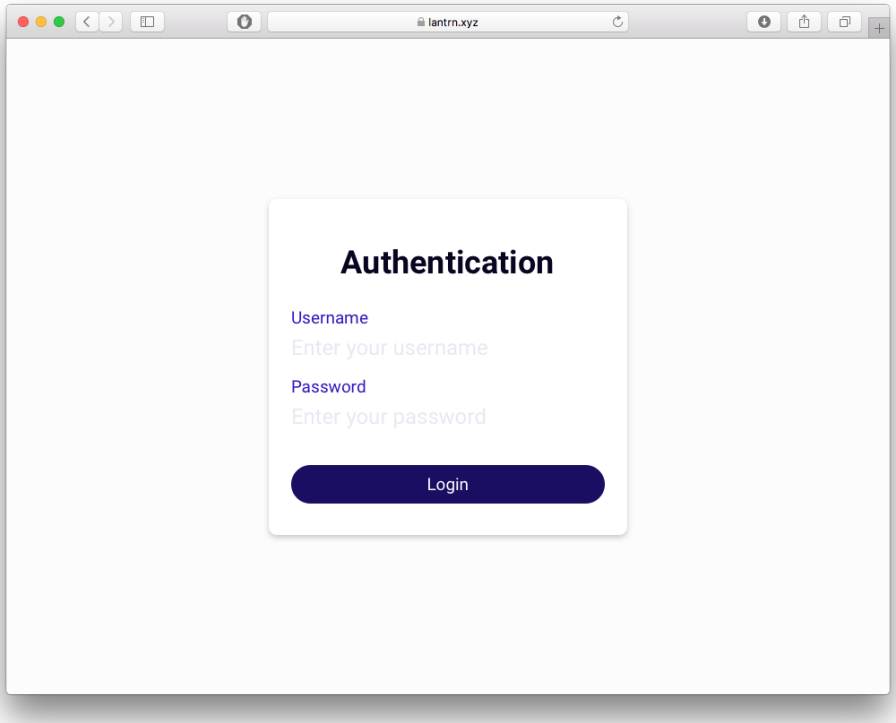
- **POST /api/result**: Deze aanvraag voegt een nieuw resultaat toe. Als al een resultaat bestaat met dezelfde URL en zoekopdracht, wordt die geüpdatet. Als er al een resultaat bestaat met dezelfde zoekopdracht, host en snippet, dan worden deze samengevoegd zoals beschreven in Sectie 6.1.
- **GET /api/result/{id}**: Vraag resultaat op met gekend ID.
- **POST /api/result/{id}**: Pas een resultaat aan (wordt niet gebruikt), hierbij worden alle velden overschreven.
- **POST /api/result/{id}/set-category**: Stel de categorie van het resultaat met het gegeven ID in op de body van het HTTP-bericht.
- **DELETE /api/result/{id}**: Verwijder het resultaat met het opgegeven ID.
- **GET /api/results/{queryId}[?host=...&category=...&nogrouping]**: Geef alle resultaten terug (maximum 200) horende bij de zoekopdracht met het opgegeven ID. Hierbij kan nog gefilterd worden op categorie en de host. De resultaten worden gegroepeerd per host tot AggregatedResults tenzij de nogrouping optie wordt gebruikt.

- **DELETE /api/results/{queryId}**[**?host=...**]: Verwijder alle resultaten voor de zoekopdracht met het opgegeven ID. Als de host optie wordt opgegeven in de querystring worden enkel de resultaten van die host verwijderd.
- **GET /api/queries**: Geef alle zoekopdrachten terug.
- **POST /api/query**: Deze endpoint voegt een nieuwe zoekopdracht toe als er geen ID aanwezig is in het JSON-object (dat werd meegegeven in de body van deze aanvraag), anders wordt de bestaande zoekopdracht aangepast.
- **DELETE /api/query/{queryId}**: Verwijder de zoekopdracht met het opgegeven ID.
- **POST /api/login**: Vraag een authenticatiesleutel (token) en gebruikersidentificatie (userID) aan door middel van een gebruikersnaam en wachtwoord.
- **POST /api/register**: Maak een nieuwe gebruiker aan met een bepaalde gebruikersnaam en wachtwoord.

Naast alle API endpoints op de URL's die beginnen met /api, heeft de microservice ook de taak om de bestanden van de front-end applicatie ter beschikking te stellen op de root URL (/). Surfen naar https://domein-microservice geeft toegang tot de applicatie die besproken wordt in Hoofdstuk 7. Hiervoor wordt gebruik gemaakt van de FileServer klasse uit de standaard net/http bibliotheek van Go.

6.3 Authenticatie en beveiliging

De microservice voorziet een eigen inlogsysteem. Daarbij worden de gebruikersnaam en wachtwoord in een JSON-object doorgestuurd in een aanvraag (/api/login). De applicatie moet deze zelf aanvragen bij de gebruiker, zoals in Figuur 6.1. Als die overeenkomen met een gebruiker opgeslagen in de MongoDB users collection, wordt een token gegenereerd en teruggegeven samen met het ID (opgeslagen in databank) van de gebruiker. Beide gegevens zorgen ervoor dat de gebruiker geauthenticeerde aanvragen kan doen in de RESTful API zolang hij deze meestuurt in de HTTP-headers X-API-USER en X-API-KEY. De inlog-endpoint is de enige endpoint waarvoor geen authenticatie vereist is.



Figuur 6.1: Inloggen in de front-end applicatie.

Deze token wordt ook opgeslagen in een hashmap die het ID van een gebruiker afbeeldt op de token. Als bij een aanvraag de ID en token van een gebruiker worden meegegeven, kan snel worden gecontroleerd of deze geldig zijn. Bovendien kan hierdoor slechts één toestel of browser voor elke gebruiker tegelijkertijd ingelogd zijn. Opnieuw inloggen zorgt ervoor dat de oude token wordt overschreven.

Een geauthenticeerde gebruiker kan elk moment nieuwe gebruikers aanmaken via de /api/register endpoint.

De wachtwoorden worden natuurlijk gehashet voor ze in de databank opgeslagen worden. Daarvoor worden de hash methodes gebruikt uit de bcrypt (golang.org/x/crypto/bcrypt) bibliotheek van Go.

Aangezien de wachtwoorden en tokens verstuurd worden in de aanvragen, is het belangrijk dat het verkeer tussen de microservice en de gebruiker beveiligd is tegen o.a. afluisteren en aanpassingen. Daarom beschikt de microservice over een domeinnaam en bijhorend erkend SSL/TLS certificaat. Dit is verkregen via de gratis Letsencrypt cer-

tificaatauthoriteit (letsencrypt.org) en wordt ondersteund door alle moderne browsers. Het certificaat heeft een geldigheid van 3 maanden en wordt indien nodig vernieuwd bij het opstarten van de microservice. Hiervoor wordt gebruik gemaakt van de autocert (golang.org/x/crypto/acme/autocert) standaard bibliotheek. Deze bibliotheek maakt gebruik van het *Automatic Certificate Management Environment* (ACME) protocol en vraagt automatisch een nieuw certificaat aan als het certificaat bijna gaat vervallen.

HTTP-verkeer wordt geblokkeerd door de firewall en genegeerd door de microservice.

6.4 Online plaatsen

De API microservice wordt op een gelijkaardige manier als de crawler op de server geplaatst. Hierbij wordt ook gebruik gemaakt van de Service bibliotheek en Fabric.

6.5 Activiteiten

De ontwikkeling van de microservice nam ongeveer 4 weken in beslag. Deze weken overlapten voor een deel met de ontwikkeling van de crawler en de webapplicatie.

Enkele statistieken:

- 1.144 lijnen Go-code.
- 33 lijnen Python-code.
- 8 Go-bestanden.

Hoofdstuk 7

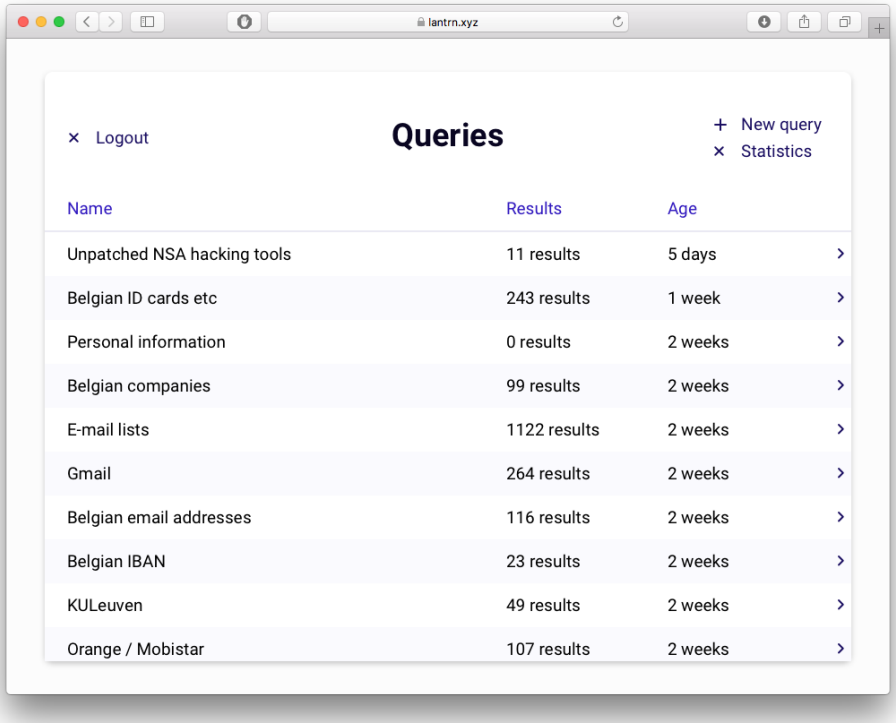
De front-end webapplicatie

7.1 Ontwerp

Het ontwerp van de webapplicatie (zie Figuur 7.1) werd getekend in het programma Sketch (enkel beschikbaar op MacOS). Vanaf Sketch werden ook alle icoontjes geëxporteerd naar PNG en SVG. Voorlopig maakt de applicatie enkel gebruik van de icoontjes met lage resolutie, maar als er later schermen met een hogere resolutie moeten worden ondersteund, kan dat gemakkelijk worden toegevoegd.

Daarna werd het ontwerp omgezet in HTML met bijhorende CSS. In plaats van rechtstreeks CSS te gebruiken werd gebruik gemaakt van SCSS (Sassy CSS) [39]. SCSS is een uitbreiding van CSS die gecompileerd wordt naar CSS d.m.v. een tool. Door onder andere deze handige toevoegingen wordt de code gemakkelijker onderhoudbaar:

- Verschillende SCSS-bestanden kunnen worden gecombineerd tot één CSS-bestand.
- Het gebruik van variabelen zorgt ervoor dat kleuren, afmetingen en lettergroottes op slechts één plaats moeten worden gedefinieerd.
- Nesting zorgt ervoor dat de opmaakregels (staan in CSS tussen accolades) die bij een CSS-selector horen, bijkomende CSS-selectors kunnen bevatten die overerven van de bovenliggende CSS-selector. Dit zorgt voor hiërarchische code die veel overzichtelijker en aanpasbaar is.
- Mixins zijn zelfgeschreven functies die CSS-code genereren op basis van enkele argumenten.



Figuur 7.1: Overzicht van de zoekopdrachten na het inloggen.

- De mogelijkheid tot overerving reduceert dubbele code.
- Het gebruik van operatoren zorgt ervoor dat berekeningen op variabelen mogelijk zijn.

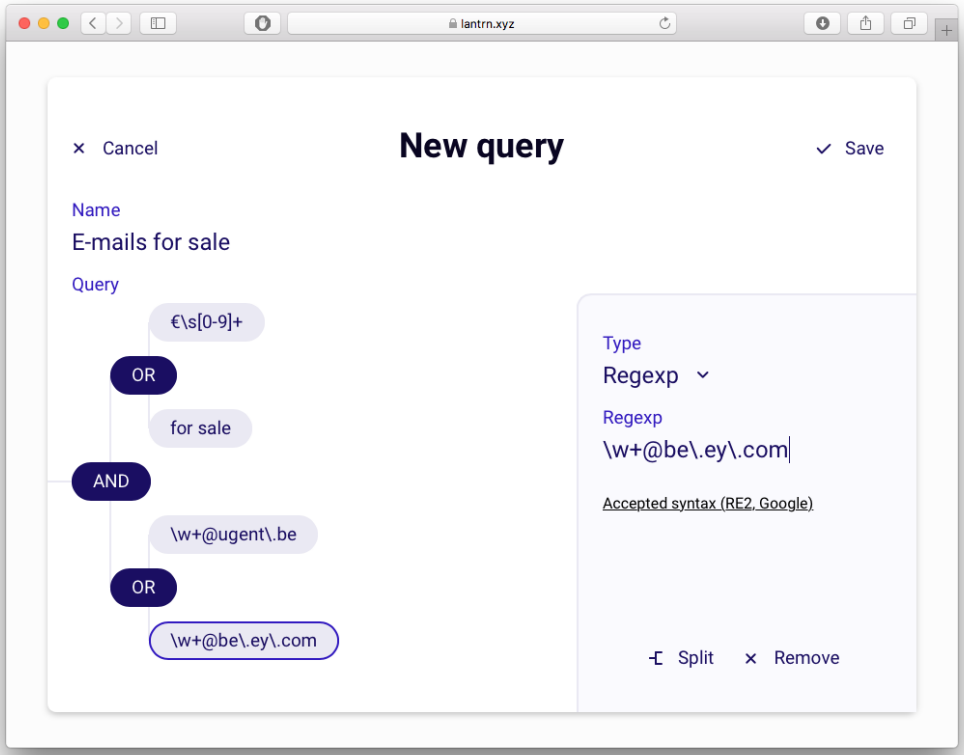
Er werd gekozen voor een responsive design. Het ontwerp past zich aan aan de schermgrootte van de gebruiker. Smartphones worden echter niet ondersteund omdat bepaalde componenten niet schaalbaar zijn naar de smartphone, zoals de opbouw van zoekopdrachten en het weergeven van darkweb pagina's.

Er wordt gebruik gemaakt van het Roboto lettertype, en dit wordt ingeladen via Google Fonts. Hier wordt later nog kort op ingegaan.

7.2 JavaScript

7.2.1 Nieuwe zoekopdracht samenstellen

Via de front-end kunnen gebruikers gemakkelijk nieuwe zoekopdrachten aanmaken. Hiervoor werd een 'QueryBuilder' ontworpen waarin de basisoperaties voor zoeken worden voorgesteld door tekstballonnen (zie Figuur 7.2). Deze ballonnen zijn verplaatsbaar met de muis. Het systeem werkt erg intuïtief en vereist dan ook geen uitleg om mee te werken.



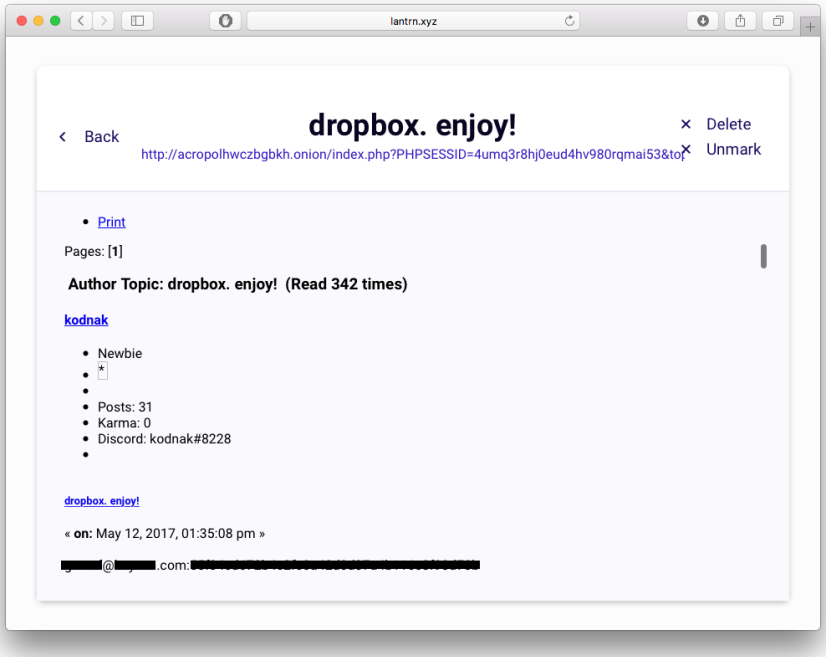
Figuur 7.2: Het opbouwen van een zoekopdracht.

Dit werd volledig geïmplementeerd in JavaScript zonder gebruik te maken van beschikbare bibliotheken. De builder bestaat eigenlijk uit een HTML-canvas (voor de lijnen tussen de ballonnen) met daarbovenop een container waarin de ballonnen als DOM-elementen met een absolute positie worden geplaatst. Deze posities worden berekend tijdens het slepen en het toevoegen van nieuwe basisoperaties. Om dit te verwezenlijken werd sterk op een object-georiënteerde manier geprogrammeerd. Dit zorgde voor een onderhoudbare code.

Voor het slepen wordt eerst bepaald welke ballon (en welke basisoperatie) aangeklikt wordt. Tijdens het slepen wordt berekend met welke ballon deze moet verwisseld worden. Dat wordt tijdelijk getoond via een animatie waarbij beide ballonnen van plaats verwisselen. Hierbij bewegen de ballonnen zoals een gedempte veer naar hun nieuwe positie, wat ervoor zorgt dat het slepen natuurlijk aanvoelt. Zodra de muis wordt losgelaten, worden de basisoperaties definitief van plaats verwisseld en de posities van alle ballonnen opnieuw berekend.

7.2.2 Weergeven van darkweb pagina's

Het is mogelijk om de momentopname van de HTML-inhoud van pagina's te bekijken toen deze werden gedownload door de crawler (Figuur 7.3). Dit moet echter doordacht gebeuren omdat dit veel veiligheidsproblemen kan veroorzaken. Het laden van externe bronbestanden kan namelijk het IP-adres van de gebruiker vrijgeven aan de hidden service. Bovendien kan JavaScript informatie verzamelen van de gebruiker.



Figuur 7.3: Een darweb pagina met Dropbox accounts en hun gehashte wachtwoorden.

Ten eerste is het noodzakelijk dat scripts in de HTML-pagina geblokkeerd worden. Daarom wordt de HTML-inhoud in een iframe tag geplaatst. Deze iframe krijgt de waarde 'allow-

same-origin' voor het sandbox attribuut en blokkeert op die manier o.a. JavaScript, Java en Flash [40].

Daarnaast mag het niet mogelijk zijn om eender welke externe bron te downloaden binnen de iframe. Als dit mogelijk zou zijn, is de identiteit (IP-adres) van de bezoeker door de darkweb pagina te achterhalen. Om dit te voorkomen wordt gebruik gemaakt van de Content-Security-Policy (CSP) HTTP-header [41]. Deze moet door de API microservice worden doorgestuurd voor alle aanvragen die betrekking hebben op de webapplicatie.

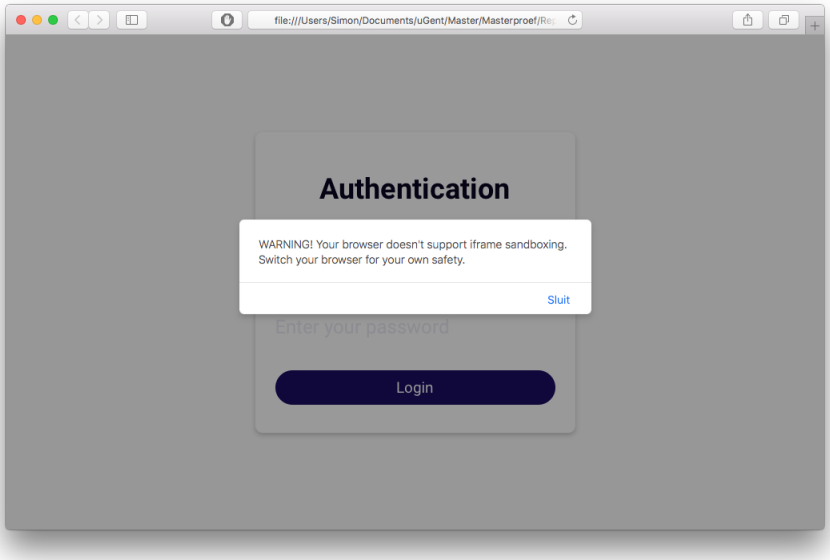
De CSP wordt zo ingesteld:

```
Content-Security-Policy: default-src 'self'; style-src 'self' 'unsafe-inline'
https://fonts.googleapis.com; font-src https://fonts.gstatic.com; script-src
'self' 'unsafe-inline'
```

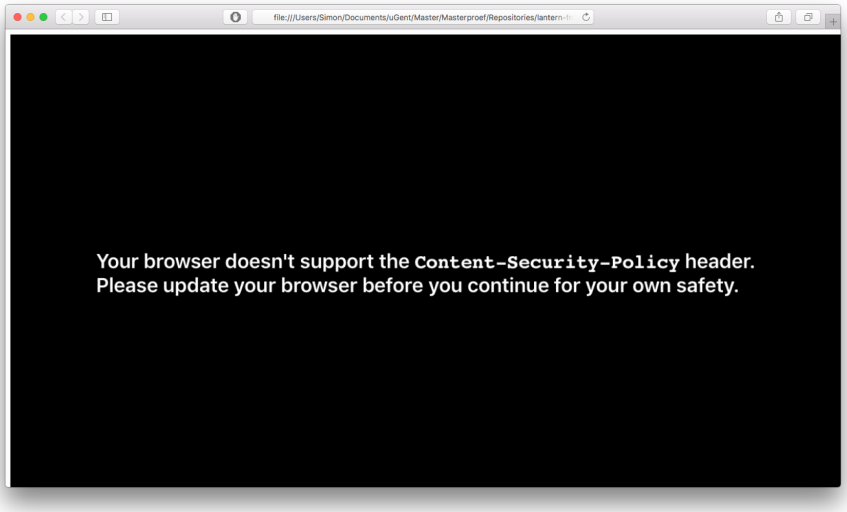
Deze beperking wordt opgelegd voor de volledige webapplicatie. De CSP zorgt ervoor dat de externe CSS enkel afkomstig mag zijn van de microservice of Google Fonts. Inline CSS en JavaScript is toegelaten in de applicatie, hoewel JavaScript wordt geblokkeerd in de iframe door het sandbox attribuut.

Niet elke browser ondersteunt de CSP-header of het sandbox-attribuut [40,41]. Bovendien is het mogelijk dat de CSP-header (tijdelijk) niet wordt doorgestuurd, bijvoorbeeld als de programmeercode van de API fouten bevat. Daarom worden twee beveiligingschecks ingebouwd:

- Bij het laden van de applicatie wordt een iframe toegevoegd met het sandbox-attribuut. Daarin wordt HTML-code geplaatst die een melding genereert via JavaScript. Als het sandbox-attribuut niet zou werken zou deze melding zichtbaar zijn voor de gebruiker. De melding brengt de gebruiker op de hoogte van het veiligheidsprobleem (zie Figuur 7.4).
- In dezelfde iframe wordt ook een afbeelding op twee verschillende manieren geplaatst. Eén wordt geplaatst als img-tag, de andere als achtergrondafbeelding via CSS. De afbeelding staat gehost op een ander domein. Als de CSP-header genegeerd wordt, zal deze afbeelding de volledige applicatie bedekken met daarop een duidelijke waarschuwing (zie Figuur 7.5).



Figuur 7.4: De veiligheidsmelding als het sandbox-attribuut niet ondersteund wordt.



Figuur 7.5: De veiligheidsmelding als de CSP-header niet verstuurd werd.

7.3 Activiteiten

De ontwikkeling van de webapplicatie nam ongeveer 7 dagen in beslag en werd voor een deel ontwikkeld voor de implementatie van de API. Op die manier kreeg ik een beeld van welke API-endpoints noodzakelijk waren.

Enkele statistieken:

- 2.297 lijnen JavaScript.
- 642 lijnen SCSS-code.
- 329 lijnen HTML-code.
- 19 JavaScript klassen.
- 9 SCSS-bestanden.
- 7 icoontjes.

Hoofdstuk 8

Resultaten

Het doel van deze masterproef was om op zoek te gaan naar gevaren voor bedrijven in het darkweb. Dit werd gerealiseerd door een zoekoplossing te creëren die de mogelijkheid heeft om erg gericht in het darkweb op zoek te gaan naar informatie.

Om de bruikbaarheid van de oplossing te testen werd de oplossing één week lang online gehouden met enkele ingestelde zoekopdrachten. Er werd gekozen om op zoek te gaan naar zaken waarvan ik vermoedde dat ze in het darkweb gevonden konden worden en naar zaken waarvan ik juist wou weten of ze in het darkweb te vinden zijn. Ondanks de korte testperiode konden enkele nuttige resultaten bekomen worden. Deze resultaten staan in dit hoofdstuk opgelijst, ongeacht of ze daadwerkelijk iets relevant opleverden.

8.1 Informatie over het darkweb

Tijdens het uittesten heeft de crawler ook meer informatie over het darkweb kunnen verzamelen. Hier werd niet specifiek naar gezocht, maar dit is interessant voor vervolgonderzoek.

Aantal beschikbare hidden services	4.854
Aantal niet beschikbare hidden services	763.141
Niet geprobeerd	4.431

Tabel 8.1: Bereikbaarheid van hidden services voor correctie.

In Tabel 8.1 staat een overzicht van het aantal hidden services die gevonden worden. Zodra één pagina van een hidden service kon worden gedownload beschouwen we deze als beschikbaar. Een hidden service beschouwen we pas als onbeschikbaar als geen enkele pagina kon worden gedownload en er minimaal één pagina geprobeerd werd. De overige hidden services plaatsen we in de categorie niet geprobeerd.

Wat meteen opvalt is het gigantische aantal onbereikbare hidden services tegenover het kleine aantal bereikbare hidden services. Tabel 8.2 bevat twee uittreksels uit de lijst van onbereikbare hidden services. We hebben een sterk vermoeden dat deze lijsten veel gegenereerde domeinnamen bevatten. Er moeten dus één of meerdere hidden services aanwezig zijn die verwijzingen bevatten naar grote hoeveelheden gegenereerde .onion domeinnamen. Waarom ze dit doen is niet duidelijk en vereist bijkomend onderzoek.

zzzzzzzzzzzzzzqh.onion	sctnkaduco5umxoz.onion
zzzzzzzzzzzzzzqi.onion	sctnkaduco5umxp2.onion
zzzzzzzzzzzzzzqj.onion	sctnkaduco5umxp3.onion
zzzzzzzzzzzzzzqk.onion	sctnkaduco5umxp4.onion
zzzzzzzzzzzzzzql.onion	sctnkaduco5umxp5.onion
zzzzzzzzzzzzzzqm.onion	sctnkaduco5umxp6.onion
zzzzzzzzzzzzzzqn.onion	sctnkaduco5umxp7.onion
zzzzzzzzzzzzzzqo.onion	sctnkaduco5umxpa.onion
zzzzzzzzzzzzzzqp.onion	sctnkaduco5umxpb.onion
zzzzzzzzzzzzzzqq.onion	sctnkaduco5umxpc.onion
zzzzzzzzzzzzzzqr.onion	sctnkaduco5umxpd.onion

Tabel 8.2: Twee uittreksels uit de lijst van onbereikbare hidden services.

Omdat deze domeinnamen de statistieken erg beïnvloedden werden ze automatisch geteld met behulp van een script die de perfect alfabetisch opeenvolgende namen telt in de lijst van niet geprobeerde en onbereikbare hidden services. Dit resulteerde in 751.075 gegenereerde domeinnamen, waarvan 4.096 domeinnamen in de categorie 'niet geprobeerd'. Een overzicht van de gecorrigeerde aantallen zijn terug te vinden in Tabel 8.3.

	Niet gegenereerd	Gegenereerd
Aantal beschikbare hidden services	4.854	0
Aantal niet beschikbare hidden services	16.162	746.979
Niet geprobeerd	335	4.096

Tabel 8.3: Bereikbaarheid van hidden services na correctie.

Er werden meer bereikbare hidden services teruggevonden dan het OnionScan onderzoek in de maand maart van 2017 vond (4.000, zie Sectie 2.3.1). Het rapport sprak toen van een sterke daling in het aantal bereikbare hidden services [20]. Dit aantal is ondertussen dus weer gestegen.

Het grote aantal gegenereerde domeinnamen verklaart ook iets wat tijdens het crawlen niet verklaard kon worden. De snelheid van de crawler bleef toen niet stabiel. Vaak werden snelle periodes (± 30 pagina's per seconde) van 10 minuten afgewisseld met 20 trage minuten (± 2 pagina's per seconde). Dit was zichtbaar in de statistieken van de crawler. De gegenereerde domeinnamen moeten namelijk eerst minimaal 40 keer getest worden voor deze een uur genegeerd worden (dit gaat exponentieel verder zoals besproken in Sectie 5.3.8). De crawler moet zich hier in de toekomst aan aanpassen zodat dit kan worden voorkomen.

Uiteindelijk werden in de beschikbare hidden services in totaal 1.941.523 unieke pagina's gedownload. 17.906.082 andere URL's van beschikbare hidden services staan in de wachtrijen maar zijn nog nooit gecrawld.

8.2 Persoonsgegevens

Een zoekopdracht ging via een reguliere expressie op zoek naar pagina's die meer dan 10 e-mailadressen vermelden. Getallen lager dan tien gaven te veel onrelevante resultaten terug. Met deze zoekopdracht was het de bedoeling om lijsten van e-mailadressen te vinden samen met hun gelekte wachtwoorden.

Dit resulteerde in de vondst van een groot aantal lijsten met e-mailadressen en wachtwoorden van allerlei diensten. Daarbij werden ook e-mailadressen van klanten van Telenet gevonden. Alle betrokken partijen werden op de hoogte gebracht.

Alle gevonden e-mailadressen werden opgelijst en in één bestand geplaatst. Vervolgens werden deze geordend per provider. Tabel 8.4 geeft een overzicht van de 15 meest voorkomende providers. De meest voorkomende providers zijn de gratis e-maildiensten (o.a. gmail, hotmail en yahoo). Dit is op zich niet zo speciaal. Wat opvalt zijn de 60 e-mailadressen van de *internet crime* afdeling van Amerikaanse federale politiedienst (FBI). Deze werden gevonden samen met de URL's van bijhorende inlogpagina's en persoonlijke gegevens (adres, geboortedatum) van de FBI-baas.

Aantal e-mailadressen	**Provider**
2033	gmail.com
1283	hotmail.com
936	yahoo.com
203	aol.com
160	mail.ru
135	hotmail.co.uk
130	live.nl
82	msn.com
68	ic.fbi.gov
66	comcast.net
55	yahoo.co.uk
48	sbcglobal.net
41	live.co.uk
41	btinternet.com
36	mac.com

Tabel 8.4: Aantal gevonden e-mailadressen met wachtwoorden per provider.

Tabel 8.5 geeft een overzicht van het aantal gevonden e-mailadressen voor enkele Belgische en Nederlandse providers, bedrijven en overheden. Hierbij staat ook een e-mailadres van de Nederlandse overheid (Ministerie van Algemene Zaken, minaz.nl) vermeld. Deze kwam

voor in een gehackte database van een Nederlandstalige website, met het bijbehorende gehashte wachtwoord. Aangezien gebruik werd gemaakt van het onveilige unsalted MD5-algoritme, was het eenvoudig om het overeenkomstige wachtwoord te vinden door de hash simpelweg te googlen [42].

In totaal gaat het om 7.373 e-mailadressen uit zoekresultaten die verwerkt werden. Deze e-mailadressen en wachtwoorden zijn niet altijd de inloggegevens voor het bekijken van e-mails. Soms zijn het de inloggegevens voor een bepaalde dienst. De dienst in kwestie staat dan vermeld op de pagina waarop de e-mailadressen gevonden werden. Zo werden o.a. inloggegevens van Steam en Netflix accounts teruggevonden.

Om ethische en juridische redenen werden de gevonden wachtwoorden nooit getest. De kans is groot dat veel van deze wachtwoorden niet meer geldig zijn en reeds zijn aangepast. Deze worden namelijk vaak gepubliceerd door hackers die een demo geven van wat ze te verkopen hebben op forums. Daarnaast werd ook veel informatie gevonden op verschillende darkweb versies van Pastebin (website waarop tekst kan worden geüpload).

Aantal e-mailadressen	Provider
130	live.nl
18	hotmail.nl
16	telenet.be
4	quicknet.nl
3	planet.nl
3	kpnmail.nl
2	zonnet.nl
2	ziggo.nl
2	xs4all.nl
2	upcmail.nl
2	hva.nl
2	chello.nl
1	vestingfinance.nl
1	tele2.nl
1	skynet.be
1	partyclick.nl
1	minaz.nl
1	mail.be
1	live.be
1	home.nl
1	eturn.nl
1	duycker.nl
1	djorlando.nl

Tabel 8.5: Aantal gevonden e-mailadressen met wachtwoord van Belgische en Nederlandse providers, overheden en bedrijven.

8.3 Vervalste identiteitskaarten

Via een zoekopdracht werd gezocht naar vermeldingen van Belgische identiteitskaarten of papieren. Daarbij werd een groot aantal pagina's gevonden van personen die vervalste identiteitsgegevens (o.a. identiteitskaarten, rijbewijzen en paspoorten) verkopen. Veel van deze pagina's zijn waarschijnlijk bedrog omdat ze een e-mailadres van Gmail gebruiken. Een legitieme hacker gebruikt waarschijnlijk enkel diensten binnen het darkweb voor communicatie.

8.4 Vermeldingen van Belgische bedrijven

Het blijkt behoorlijk moeilijk om op zoek te gaan naar relevante vermeldingen van bedrijven in het darkweb. Vaak wordt foutief een zoekresultaat gegenereerd omdat iemand een typfout maakt en zo een Belgische bedrijfsnaam vormt. Sommige bedrijfsnamen komen bovendien voor in een hele andere context. Het Belgische bedrijf Solvay wordt bijvoorbeeld vernoemd als het gaat over het gelijknamige chemische proces.

Voor de rest zijn het vaak politieke of alternatieve bewegingen die artikels schrijven over bedrijven. Een voorbeeld van een titel is: "Your Guide to the Latest Efforts to Hold Big Banks Accountable". Belgische banken worden ook vermeld door personen die creditcardgegevens verkopen.

8.5 Vermeldingen van de UGent of KULeuven

E-mailadressen van UGent of KULeuven werden enkel gevonden in verband met commits in open source software. De naam 'Ghent University' of 'KULeuven' werd meestal vermeld in historische context of in verband met publicaties van onderzoek.

8.6 Informatie over Belgische kerncentrales

In december 2016 ging nog een gerucht de ronde dat plannen van een Belgische nucleare installatie (Tihange) op het darkweb te vinden waren [43]. De Belgische veiligheidsdien-

sten zijn toen actief op zoek gegaan met o.a. privébedrijven maar konden de plannen in kwestie niet terug vinden.

Een zoekopdracht van deze applicatie ging ook op zoek naar vermeldingen van Belgische kerncentrales. Hierbij werd gezocht naar een combinatie van een plaatsnaam van een kerncentrale (o.a. Belgium, Tihange, Doel) en een woord dat verband houdt met een kerncentrale (o.a. nuclear, reactor).

Hier werd niets concreet teruggevonden. De gevonden pagina's bespreken meestal de geschiedenis van kernwapens.

8.7 WannaCry ransomware

De grote cyberaanval met de WannaCry ransomware was het gevolg van een veiligheidslek in het Server Message Block (SMB) protocol dat op Windows gebruikt werd. De Amerikaanse National Security Agency (NSA) had hiervoor een exploit ontworpen genaamd 'EternalBlue' [44].

Zoeken op deze termen leidde al snel tot het vinden van enkele andere veiligheidslekken die nog niet allemaal gedicht werden. De codenamen van deze exploits werden mee toegevoegd aan de zoekopdracht om misbruik te detecteren. Deze zoekopdracht bracht verder geen interessante informatie op.

Hoofdstuk 9

Conclusie

Het is mogelijk voor bedrijven om veiligheidsproblemen te vinden door het darkweb te onderzoeken, maar dit is niet nuttig voor elk bedrijf. Uit de resultaten blijkt dat vooral internetbedrijven, banken en overheden te maken krijgen met de publicatie van gegevens (inloggegevens of creditcardgegevens) van klanten. Voor die groep is het darkweb interessant omwille van die publicaties. Voor andere bedrijven vereist dit waardeoordeel meer onderzoek. Dit kan aan de hand van specifiekere zoekopdrachten.

De informatie die gevonden werd in het darkweb bevindt zich vooral in de categorie van persoonsgegevens. De kwaliteit van een groot deel van deze informatie is echter ondermaats. Het gaat namelijk om persoonsgegevens die hackers gratis weggeven om zichzelf te verkopen en klanten te overtuigen. Uit verder onderzoek blijkt dat een groot deel van deze informatie ook in het clearweb kan worden gevonden als de e-mailadressen uit deze lijsten op bijvoorbeeld Google worden opgezocht.

Met behulp van de gevonden lijsten is het mogelijk om in te schatten welke bedrijven te maken krijgen met de verkoop van inloggegevens. Zo komen enkele Belgische en Nederlandse providers voor in deze lijsten, wat het interessant maakt om een volledige lijst aan te kopen met hogere kwaliteit. De kans is groot dat daar meer gegevens van Belgen of Nederlanders terug te vinden zijn. Bedrijven zouden hiervoor kunnen samenwerken. Op die manier kunnen bedrijven voorkomen dat deze gegevens in de handen van criminelen terecht komen. Deze oplossing kan ethische bezwaren oproepen omdat de criminelen financieel worden beloond.

Bedrijven kijken beter na of e-mailadressen van hun medewerkers niet voorkomen in het darkweb, ook al zijn dat persoonlijke e-mailadressen. Men kan beter voorkomen dat bepaalde wachtwoorden in de handen van criminelen terechtkomen. Maar niet elk bedrijf moet daarom afzonderlijk een onderzoek voeren. Dit wordt beter worden uitbesteed aan een overkoepelend bedrijf of overheidsdienst.

Het kan voor de maatschappij interessant zijn om actief op zoek te gaan naar algemene gevaren in het darkweb. Een voorbeeld hiervan is documentatie van een kerncentrale of websites die de communicatie tussen terroristen mogelijk maken.

Hoofdstuk 10

Toekomstig werk

Uiteindelijk heeft de tool slechts één week werk geleverd. Het aantal gedownloade unieke pagina's kan gebruikt worden om een ruwe schatting te geven van de grootte van het onderzochte deel van het darkweb. Er werden 1.941.523 unieke URL's gedownload, 17.906.082 URL's staan in de wachtrij maar zijn nog nooit gedownload, dat is dus een bedekking van 9,78%. De werkelijk bedekking zal wellicht kleiner zijn omdat niet alle bestaande URL's gekend zijn.

De bedekking is nogal klein om uitgebreide conclusies te trekken. Het geeft wel een indicatie van wat mogelijk is, aangezien op korte termijn toch wel enkele resultaten zijn bereikt. Indien de oplossing verder uitgebreid wordt kan deze gebruikt worden als wakend oog over het darkweb in dienst van bedrijven of de maatschappij.

De huidige oplossing kan geen pagina's crawlen die beveiligd zijn met wachtwoorden of captcha's. De hoeveelheid pagina's die hiermee beveiligd zijn is niet bekend, maar uit eigen onderzoek bleek dat enkele grote darkweb markets hiervan gebruik maken. Dit is dus iets wat zeker kan toegevoegd worden in een vervolgonderzoek. Bovendien genereren de zoekopdrachten vaak veel overbodige resultaten. Door het toevoegen van extra basisoperaties moet het mogelijk zijn om het zoeken te verfijnen. Het gebruik van machine learning kan hier misschien bij helpen zoals onderzocht in de paper van Nunes et al. (besproken in Sectie 2.4.1). Het kan ook interessant zijn om een ander darkweb dan Tor te bestuderen.

Ook op gebied van snelheid moeten er aanpassingen gemaakt worden. Zo moet er een oplossing worden bedacht voor het grote aantal gegenereerde .onion adressen en hun invloed op de snelheid van de crawler (zie Sectie 8.1). Bovendien kan de toevoeging van meerdere servers die parallel crawlen de snelheid dramatisch verhogen.

Momenteel wordt een actieve worker telkens volledig in het geheugen gehouden door de crawler. Het geheugengebruik van grote websites kan daarom voor problemen zorgen. Deze problemen bleven in dit onderzoek beperkt, hoewel deze problemen zouden kunnen opduiken als de crawler langer actief was geweest. Om dit op te lossen zou het mogelijk moeten zijn om een worker slechts gedeeltelijk in het geheugen te houden. De CrawlItem's op het einde van lange wachtrijen zouden bijvoorbeeld tijdelijk naar een bestand kunnen worden geschreven. De AlreadyVisited hashmaps zouden ook vervangen kunnen worden door een alternatief dat minder geheugen in beslag neemt.

Bibliografie

[1] Gartner. (z.j.). *Threat Intelligence: What is it, and How Can it Protect You from Today's Advanced Cyber-Attacks?*. Geraadpleegd van https://www.gartner.com/imagesrv/media-products/pdf/webroot/issue1_webroot.pdf

[2] Deloitte. (2016, 4 april). *Cybercriminaliteit kost Nederlandse organisaties 10 miljard euro per jaar* [Persbericht]. Geraadpleegd van https://www2.deloitte.com/nl/nl/pages/over-deloitte/articles/cybercriminaliteit-kost-nederlandse-organisaties-10-miljard-euro-per-jaar.html

[3] Hamilton Place Strategies. (2016, 2 februari). *Cybercrime Costs More Than You Think*. Geraadpleegd van https://www.hamiltonplacestrategies.com/insights/cybercrime-costs-more-you-think/

[4] CERT.be. (2016). *Jaaroverzicht van CERT.be*. Geraadpleegd van https://www.cert.be/files/CERTbe_Jaaroverzicht2015.pdf

[5] H. Slatman. (z.j.). *A curated list of Awesome Threat Intelligence resources*. Geraadpleegd op 9 juni 2017, van https://github.com/hslatman/awesome-threat-intelligence

[6] B. Metivier. (2016, 12 juli). *A Guide to Cyber Threat Intelligence Sources*. Geraadpleegd van https://www.sagedatasecurity.com/blog/guide-to-cyber-threat-intelligence-sources

[7] The Tor Project, Inc. (z.j.). *Tor: Overview*. Geraadpleegd op 9 juni 2017, van https://www.torproject.org/about/overview.html.en

[8] Facebook Inc. (2014, 31 oktober). *Making Connections to Facebook more Secure*. Geraadpleegd van https://www.facebook.com/notes/protect-the-graph/making-connections-to-facebook-more-secure/1526085754298237

[9] I2P. (z.j.). *Het onzichtbare internet project*. Geraadpleegd op 9 juni 2017, van https://geti2p.net/nl/

[10] Freenet. (z.j.). *What is Freenet?*. Geraadpleegd op 9 juni 2017, van https://freenetproject.org/pages/about.html

[11] The Tor Project, Inc. (z.j.). *Tor Rendezvous Specification* [Handleiding]. Geraadpleegd op 30 maart 2017, van https://gitweb.torproject.org/torspec.git/tree/rend-spec.txt

[12] The Tor Project, Inc. (z.j.). *Tor version 3 directory server specification* [Handleiding]. Geraadpleegd op 9 juni 2017, van https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt

[13] The Tor Project, Inc. (z.j.). *Tor path selection specification* [Handleiding]. Geraadpleegd op 9 juni 2017, van https://gitweb.torproject.org/torspec.git/tree/path-spec.txt

[14] R. Dingledine, N. Mathewson & P. Syverson. (z.j.). *Tor: The Second-Generation Onion Router*. Geraadpleegd van https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf

[15] The Tor Project, Inc. (z.j.). *Tor's extensions to the SOCKS protocol* [Handleiding]. Geraadpleegd op 9 juni 2017, van https://gitweb.torproject.org/torspec.git/tree/socks-extensions.txt

[16] Google Inc. (z.j.). *How Search organizes information*. Geraadpleegd op 9 juni 2016, van https://www.google.com/search/howsearchworks/crawling-indexing/

[17] Tianjun Fu, Ahmed Abbasi & Hsinchun Chen. (z.j.). *A Focused Crawler for Dark Web Forums* (The University of Arizona). Geraadpleegd van http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.690.5833&rep=rep1&type=pdf

[18] Mascherari Press. (2016, 26 april). *OnionScan Report: April 2016 - The Tor Network: Security and Crime* [Blogpost]. Geraadpleegd van `https://mascherari.press/onionscan-report-april-2016-the-tor-network-security-and-crime/`

[19] Mascherari Press. (2016, 30 november). *OnionScan Report: This One Weird Trick Can Reveal Information from 25% of the Dark Web* [Blogpost]. Geraadpleegd van `https://mascherari.press/onionscan-report-this-one-weird-trick-can-reveal-information-from-25-of-the-dark-web-2/`

[20] Mascherari Press. (2017, 6 maart). *OnionScan Report: Freedom Hosting II, A New Map and a New Direction* [Blogpost]. Geraadpleegd van `https://mascherari.press/onionscan-report-fhii-a-new-map-and-the-future/`

[21] Mascherari Press. (2017, 15 januari). *OnionScan Report: Reconstructing the Finances of Darknet Markets through Reputation Systems* [Blogpost]. Geraadpleegd van `https://mascherari.press/onionscan-report-forensic-finances-dark-markets/`

[22] Eric Nunes, Ahmad Diab, Andrew Gunn, Ericsson Marin , Vineet Mishra, Vivin Paliath, John Robertson, Jana Shakarian, Amanda Thart & Paulo Shakarian. (2016). *Darknet and Deepnet Mining for Proactive Cybersecurity Threat Intelligence* (Arizona State University). Geraadpleegd van `https://arxiv.org/pdf/1607.08583.pdf`

[23] Amazon Web Services, Inc. (z.j.). *Amazon EC2 Instance IP Addressing* [Webpagina]. Geraadpleegd op 9 juni 2017, van `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-instance-addressing.html`

[24] Google Inc. (z.j.). *Frequently Asked Questions (FAQ)* [Webpagina]. Geraadpleegd op 9 juni 2017, van `https://golang.org/doc/faq`

[25] MongoDB, Inc. (z.j.). *What is MongoDB?* [Webpagina]. Geraadpleegd op 9 juni 2017, van `https://www.mongodb.com/what-is-mongodb`

[26] Tarquin. (2016, 20 september). *How To Access The Dark Web.* Geraadpleegd van `https://darkwebnews.com/help-advice/access-dark-web/`

[27] Google Inc. (2017, 9 juni). *reCAPTCHA V2* [Handleiding]. Geraadpleegd op 9 juni 2017, van `https://developers.google.com/recaptcha/docs/display`

[28] 2Captcha. (z.j.). *Human-powered CAPTCHA-solving service* [Webpagina]. Geraadpleegd op 9 juni 2017, van `https://2captcha.com`

[29] K. Bauer, D. McCoy, D. Grunwald, T. Kohno & D. Sicker. (2017). *Low-Resource Routing Attacks Against Tor* (University of Colorado & University of Washington) Geraadpleegd van `https://www.freehaven.net/anonbib/cache/bauer:wpes2007.pdf`

[30] Arma. (2009, 18 februari). *One cell is enough to break Tor's anonymity.* Geraadpleegd van `https://blog.torproject.org/blog/one-cell-enough`

[31] P. Hayes. (2017, 11 mei). *How can I get the exact location of an IP address?.* Geraadpleegd op 9 juni 2017, van `https://www.quora.com/How-can-I-get-the-exact-location-of-an-IP-address`

[32] K. Müller. (2015). *Defending End-to-End Confirmation Attacks against the Tor Network* (Gjøvik University College). Geraadpleegd van `https://brage.bibsys.no/xmlui/bitstream/id/355191/KMuller_2015.pdf`

[33] Google Inc. (z.j.). *Hoe Google Zoeken werkt.* Geraadpleegd op 9 juni 2017, van `https://support.google.com/webmasters/answer/70897`

[34] G. Linden. (2006, 9 november). *Marissa Mayer at Web 2.0.* Geraadpleegd op 9 juni 2017, van `http://glinden.blogspot.be/2006/11/marissa-mayer-at-web-20.html`

[35] MongoDB Inc. (z.j.). *MongoDB Limits and Thresholds.* Geraadpleegd op 9 juni 2017, van `https://docs.mongodb.com/manual/reference/limits/`

[36] G. P. Hemsley. (2017, 10 mei). *MIME Sniffing* [Standaard]. Geraadpleegd op 9 juni 2017, van `https://mimesniff.spec.whatwg.org`

[37] IETF. (2014, juni). *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content* [Standaard]. Geraadpleegd van `http://httpwg.org/specs/rfc7231.html`

[38] MongoDB Inc. (z.j.). *BSON Types* [Handleiding]. Geraadpleegd op 9 juni 2017, van `https://docs.mongodb.com/manual/reference/bson-types/`

[39] Sass. (z.j.). *Sass (Syntactically Awesome StyleSheets)* [Documentatie]. Geraadpleegd op 9 juni 2017, van `http://sass-lang.com/documentation/file.SASS_REFERENCE.html`

[40] MDN. (z.j.). *HTML <iframe> element* [Documentatie]. Geraadpleegd op 9 juni 2017, van `https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe`

[41] MDN. (z.j.). *HTTP Content-Security-Policy response header* [Documentatie]. Geraadpleegd op 9 juni 2017, van `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy`

[42] K. Brown. (2013, 15 november). *The Dangers of Weak Hashes*. Geraadpleegd van `https://www.sans.org/reading-room/whitepapers/authentication/dangers-weak-hashes-34412`

[43] Belga. (2016, 22 november). *Geen gevoelige plannen nucleaire sites op Darknet gevonden*. Geraadpleegd van `http://www.knack.be/nieuws/belgie/geen-gevoelige-plannen-nucleaire-sites-op-darknet-gevonden/article-belga-780041.html`

[44] L. V. Lonkhuyzen. (2017, 14 mei). *WannaCry werd mede mogelijk gemaakt door lek bij de NSA*. Geraadpleegd van `https://www.nrc.nl/nieuws/2017/05/14/wannacry-kwam-bij-de-nsa-vandaan-9075218-a1558565`