Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Information Security and Communication Technology

Øyvind Jensen

The Cyber Threat Landscape on Blacklisted Malicious Domains

Master's thesis in Information Security
Supervisor: Assoc. Prof. Dr. Geir Olav Dyrkolbotn
June 2019

NTNU
Norwegian University of Science and Technology

NTNU
Norwegian University of
Science and Technology

The Cyber Threat Landscape on Blacklisted Malicious Domains

Øyvind Jensen

01-06-2019

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Information Security and Communication Technology
Norwegian University of Science and Technology,

Supervisor: Assoc. Prof. Dr. Geir Olav Dyrkolbotn
Co-Supervisor: Dr. Andrii Shalaginov

Acknowledgments

With this thesis I finish my two-year specialization into computer security, an important and fascinating leaf on the tree that computer science is. This master program has been challenging, interesting, but not at least rewarding and I would like to thank the Gjøvik campus for delivering such an experience. I would like to thank my supervisor, Dr. Andrii Shalaginov, for his detailed, precise feedback and guidance during the pre-project and actual master thesis.
Finally, I would also like to thank my mother, for invaluable feedback on the different small aspects of a master thesis; its form and contents. Additionally, my father for invaluable feedback on the importance of taking breaks. I would not have finished my thesis without the help I have received, and I am grateful for it, but had I taken all the recommended breaks I would not have finished the thesis on time.

Øyvind Jensen, May 27th 2019.

Abstract

The internet is a dangerous place, filled with lots of different malware. That is why blacklists have been utilized for a long time to block known infection and delivery sources. By blocking domains, we do not have to bother with them anymore and our system is protected against being infected by these domains. However, by blacklisting the domains and forgetting about what is behind them, we are leaving a landscape of threats to be unknown and forgotten. In this thesis, we have found that the cyber threat landscape on blacklisted domains is like that of the general cyber threat landscape. Furthermore, we have shown that the focus on internet and its applications by malicious actors are nothing new and has been going on since 2006. Even with an internet that is as dangerous as it is, internet users have never had a safer foundation in their operating systems, browsers and applications than what we currently have. Secure developmental methodology through security in compilers to randomized memory layouts are amongst some fundamental security pieces that have been standardized in the last decade. By shedding light on this part of the cyber threat landscape we have increased the information security field's holistic understanding of the landscape we are working with. Understanding that updates are the simplest and most efficient way to secure your system against any exploitation should be good news for users given that most, if not all, applications has been equipped with automatic updates.

Contents

Acknowledgments . . . . **i**
Abstract . . . . **ii**
Contents . . . . **iii**
List of Figures . . . . **vi**
List of Tables . . . . **viii**
Listings . . . . **ix**
Glossary . . . . **xi**
Acronyms . . . . **xii**
1 Introduction **1**
1.1 Topic covered by the project 1
1.2 Keywords 2
1.3 Problem description 2
1.4 Justification, motivation and benefits 2
1.5 Research questions 2
1.6 Planned contributions 3
1.7 Thesis outline 4
2 Cyber threat landscape evolution **5**
2.1 Malware taxonomy 5
2.2 Exploit kits 6
2.3 Malware evolution 7
2.3.1 Top 5 malware categories ’06-’18 8
2.3.2 Noteworthy takeaways from industry reports 10
2.4 The cybercrime black markets 14
3 Microsoft Windows operating system and system security measures evolution **17**
3.1 Microsoft Windows operating system security measures 17
3.2 Browser security 21
3.2.1 Most used browsers 21
3.2.2 Security measures 23
4 State of the Art **25**
4.1 Social Engineering in cyberspace 25
4.2 Cyber threats from malicious websites 27
5 Choice of methods **29**
5.1 Literature Review 29
5.2 The overall procedure 32

5.3 Data collection . . . 32
5.3.1 Crawler . . . 34
5.3.2 Data storage structure . . . 38
5.3.3 Domain data collection . . . 39
5.3.4 Cuckoo Sandbox . . . 42
5.4 Data analysis . . . 43
5.4.1 Identification of possible malware files . . . 43
5.4.2 Preliminary automated analysis . . . 44
5.4.3 Manual domain analysis . . . 44
5.4.4 Content analysis . . . 44
5.4.5 Results analysis . . . 45
5.5 Tools . . . 45
5.5.1 Pandas DataFrame . . . 45
5.5.2 VMware Workstation . . . 46
5.5.3 Clonezilla . . . 47
5.5.4 Github . . . 47
6 Experiments and results . . . 48
6.1 Environment setup . . . 48
6.1.1 Host machine(s) for VMs . . . 48
6.1.2 Linux VMs used for crawling and analysis . . . 49
6.1.3 Windows VMs used for sandboxing analysis . . . 49
6.1.4 Network diagram . . . 52
6.1.5 Software . . . 53
6.2 Collection phase . . . 55
6.2.1 Crawler . . . 55
6.2.2 Domain data collection . . . 61
6.2.3 Cuckoo . . . 64
6.3 Analysis phase . . . 66
6.3.1 File and link analysis . . . 66
6.3.2 Text content analysis . . . 69
6.3.3 GeoIP analysis . . . 72
6.3.4 WHOIS analysis . . . 73
6.3.5 URL Abuse analysis . . . 73
6.3.6 VirusTotal analysis . . . 74
6.3.7 Cuckoo analysis . . . 75
6.3.8 Manual analysis . . . 76
7 Discussion . . . 81
7.1 Theoretical implications . . . 84
7.2 Practical implications . . . 86
8 Summary and future work . . . 88

Bibliography **90**
Appendices **97**
Appendix A Detailed module listing for Crawler VM **98**
Appendix B Detailed module listing for Cuckoo Host VM **102**
Appendix C Save procedures for the crawler **105**
Appendix D Post-duplication removal word description **107**

List of Figures

1 Browser statistics from NetMarketShare [1] 22
2 Illustration of multi-processor components in [2], based on Figure 5 and section 3.1, created in Visio Professional 24
3 Download button example from a file-sharing website 26
4 Illustration of the thesis workflow, created in Visio Professional 33
5 Illustration of a traditional website, created in Visio Professional 35
6 Flowchart illustrating the overall flow of the crawler, created in Visio Professional . . 37
7 Flowchart illustrating the usage of domain data collection resources, created in Visio Professional 39
8 Example output from GeoIP from a malicious domain lookup 40
9 Flowchart illustrating analysis stages and progression, created in Visio Professional . 43
10 Illustration of the virtualization layer on the Workstation host machine, created in Visio Professional 50
11 Illustration of the virtualization layer on the Dell Precision host machine, created in Visio Professional 51
12 A network diagram of the internet-connected systems in this thesis, created in Visio Professional 54
13 Flowchart of the crawler implementation, created in Visio Professional 57
14 Illustration of the external gatherer scripts flow, created in Visio Professional 62
15 Iframe illustration, created in Visio Professional 68
16 ClamAV classification distribution of all downloaded files from the crawler 69
17 ClamAV threat categories 69
18 Topic modeling output from modeling website titles 73
19 Detailed Cuckoo scores for the 5 most common scores per analysis machine 76
20 The runtime of the Cuckoo analysis VMs 77
21 The network tab in the Firefox web console 77
22 The inspector tab in the Firefox web console 78
23 The types of domains visited during the analysis 78
24 Google Safe Browsing warnings shown during analysis 79
25 A snippet of the VirusTotal website, advanced domain information link is highlighted in yellow 79
26 Location of files shown when analyzing domains on VirusTotal 80
27 Threat categories found by using VirusTotal's advanced domain information 80
28 New detection signatures created each year, 2002 to 2009 — number from [3, 4] . . 83

29 New detection signatures created each year, 2010 to 2018 — numbers from [5, 6, 7, 8] 84

List of Tables

1 Colored after appearing consecutively for *n* amount of years 8
2 Top 5 malware categories years 2006-2007 8
3 Top 5 malware categories years 2008-2010 9
4 Top 5 malware categories years 2011-2014 9
5 Top 5 malware categories years 2015-2017 10
6 Top 5 malware categories for the year 2018 10
7 Noteworthy takeaways years 2006-2007 11
8 Noteworthy takeaways years 2008-2010 12
9 Noteworthy takeaways years 2011-2014 13
10 Noteworthy takeaways years 2015-2017 14
11 Noteworthy takeaways for the year 2018 14
12 Windows operating system security measures evolution 18
13 Literature topics and thesis sections 30
14 Type distribution in selection from blacklist 32
15 The top 5 scans ranked on their number of positive AV engine results 75

Listings

5.1 Example output from using tree command on the download folder of a website with little content such as no external JS, CSS, etc. 38
5.2 Example output from using tree command on the download folder of a website with external content 38
5.3 Example output from doing a WHOIS lookup on a malicious domain 39
5.4 Example of URL Abuse output from a malicious domain lookup 41
5.5 Example of VirusTotal detailed domain information 41
5.6 Example of data extracted from a VirusTotal public API JSON response 42
5.7 CSV-file example 45
5.8 Column-wise DataFrame usage 46
5.9 DataFrame value counts usage 46
6.1 Excerpt from the main crawling function 56
6.2 Utility function, hash_url, used by save_file function 59
6.3 Concurrency source code from visit_site function 61
6.4 For-loop illustration 62
6.5 WHOIS implementation and usage 63
6.6 GeoIP usage 63
6.7 URL Abuse usage 64
6.8 VirusTotal usage 64
6.9 Cuckoo configuration (cuckoo.conf) 64
6.10 VMware configuration (vmware.conf) 65
6.11 Routing configuration (routing.conf) 65
6.12 Modification to Internet Explorer Cuckoo module 65
6.13 Excerpt from Cuckoo analysis insertion script 66
6.14 Top 5 resolved domains with their respective counts 67
6.15 Example filetype output from file command 67
6.16 File types of all downloaded files by crawler 67
6.17 Words description on domains without JavaScript rendering enabled 69
6.18 Top 10 most common words and their count on domains without JavaScript rendering enabled 70
6.19 Word description on domains with JavaScript rendering enabled 70
6.20 Top 10 most common words and their count on domains with JavaScript rendering enabled 70
6.21 LDA model creation in Python 71

6.22 10 topics found by our topic model together with their weights without JavaScript enabled, words occurring 100 or more times . . . 71
6.23 10 topics found by our topic model together with their weights with JavaScript enabled, words occurring 100 or more times . . . 71
6.24 10 topics found by our topic model together with their weights without JavaScript enabled, duplicates removed, filtered words . . . 72
6.25 10 topics found by our topic model together with their weights with JavaScript enabled, duplicates removed, filtered words . . . 72
6.26 Top 10 most represented countries . . . 72
6.27 Top 5 names from WHOIS . . . 73
6.28 BGP Ranking description generated from dataframe with Pandas . . . 73
6.29 Top 10 DNS Servers in BGP Ranking dataframe . . . 74
6.30 Classification of blacklisted domains in VirusTotal with their respective count . . . 74
6.31 Top 10 antivirus engines . . . 74
6.32 Cuckoo analyses scores for the Windows XP VM . . . 75
6.33 Cuckoo analyses scores for the Windows 7 VM . . . 75
6.34 Cuckoo analyses scores for the Windows 10 VM . . . 75
C.1 Save procedure . . . 105
C.2 Load procedure . . . 106
D.1 Words description on domains without JavaScript rendering enabled, post-duplication removal . . . 107
D.2 Word description on domains with JavaScript rendering enabled, post-duplication removal . . . 107

Glossary

Word	Definition
Threat	Something which can happen. E.g. a threat could be a person holding a hard disk drive out of the window while being on the top floor of a skyscraper.
Vulnerability	A weakness or error in e.g. software that is not intentionally present. This can then be used to attack said software.
Exploit	Is using a vulnerability to e.g. break into software. Thus, you can exploit a vulnerability.
Headless browser	A browser that does not have a graphical user interface.
Domain	A domain is a name that is resolving to an IP address when looking it up via DNS [9]. This means that http://exampledomain.example can be resolved to e.g. 156.123.21.23.
DNS	DNS is used when browsing the internet. When your computer tries to access http://exampledomain.example it is asking your DNS server what IP address this domain resolves to.
URL	A uniform resource locator is an identifier for a page that consists of three parts [9]; the protocol (e.g. HTTP), the DNS name and the unique path of the page. For our example we could be visiting http://exampledomain.example/style.css. Our example URL is using the HTTP protocol, it is on the domain exampledomain.example and the page we are request is the style.css which is the site's style sheet if it has one. Later, in the thesis when we are talking about a *link*, it is interchangeable with a URL.
DNS-BH	A blacklist that is provided by Risk-Analytics. For a throughout explanation, see section 5.2.
Browser	Is used interchangeably with web browser in the thesis.
Fingerprint	Environmental parameters that is used in e.g. exploit kits to detect exploitable machines. A fingerprint could be your operating system and browser. Details are explained in section 2.2.

Acronyms

IE Internet Explorer

API Application Programming Interface

SSD Solid State Drive

OEM Original Equipment Manufacturer

VM Virtual Machine

Win Windows

VPN Virtual Private Network

DDoS Distributed Denial of Service

DNS Dynamic Name System

OS Operating System

HTTP Hypertext Transfer Protocol

CSS Cascading Style Sheets

JS JavaScript

VT VirusTotal

AV Antivirus

DF DataFrame

APT Advanced Persistent Threat

HTML Hypertext Markup Language

Edge Microsoft Edge

MS Microsoft

1 Introduction

1.1 Topic covered by the project

[1] The world wide web has exploded in popularity the last two decades, the last numbers showing that there are 4.38 billion internet users[2]. With so many users there are lots of opportunity for profit, both legitimate and illegitimate. Internet companies are growing large, Google which started out as a search engine is now one of the largest companies by stock valuation in the world[3], valued at 806.9 billion dollars. Facebook, the largest social network in the world with over 2.3 billion users[4] is valued at 528.9 billion dollars[5] on the stock market.

With this many users and potential for profit there are many entrepreneurs, new websites and applications are being created all the time. Since the use of the internet is so widespread now the level of vigilance is not as high as when only specialized users used the internet. This makes the internet a good hunting ground for criminals that want to earn easy profits. The number and variations of social engineering attacks are many as can be seen in [10].

In this thesis we will be looking at blacklisted malicious domains. Specifically, from the point of view of how a user without security measures will experience visiting these websites. Blacklists are a useful, albeit old-fashioned and a static defense mechanism that has the limitations that it will not update itself, but the website addresses that are on the list will stay blocked for the users that employ the blacklist. The website addresses that are on the blacklists are on the lists because someone reported them as being malicious or spreading malware, or both. We are going to be analyzing these websites by looking at their content, what is running on them when you visit them, software that is both being automatically downloaded and which you can download from them, visiting links they have linked to, what servers they are using, etc. When we have acquired everything that we can get we will start analyzing what we have and see if the content and downloaded files can give us insights into the threats from these websites. To get this insight we will discover topics with topic modeling, identifying features that can be applicable for machine learning, gather intelligence from various sources (explained further in section 5) and use these parts to create a holistic picture of the cyber threat landscape on blacklisted malicious domains.

[1] Chapter, 1, except for section 1.7, is influenced and similar in some regard to the pre-project document created for IMT4205 with the title "The cyber threat landscape on publicly available websites labeled as malicious", authored by Øyvind Jensen, which was submitted December 2018.

[2] `https://www.internetworldstats.com/stats.htm`, retrieved 25.5.19

[3] `https://www.nasdaq.com/symbol/goog`, retrieved 20.5.19

[4] `https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/`, retrieved 25.5.19

[5] `https://www.nasdaq.com/symbol/fb`, retrieved 20.5.19

1.2 Keywords

Blacklisted domains, malware distribution, social engineering, cyber threat landscape, malicious websites

1.3 Problem description

Publicly available domains that are either legitimate compromised websites or websites created with a malicious purpose are a serious problem [11, 12, 13]. Given that these websites can be accessed by anyone means that people are at risk of being infected (by e.g. drive-by downloads) just by visiting websites in their (outdated) web browser. To combat this there have been developmental efforts towards a safer internet by building security features into operating systems, browsers and routers with e.g. certification, blacklists, sandboxing etc. Even with these developmental efforts towards a safer internet, users are still at risk if they are running old operating systems or by using old software that are missing protections against attacks. These older operating systems are typically not running the latest updates since Microsoft Windows XP, Vista, 7 and 8 are out of mainstream support[6]. This means that they will not get any more updates and other support and they can be severely outdated. Users that are vulnerable can be infected with malicious software which can then lead to these users being part of a botnet or cause other harm such as financial damages, privacy issues and other liability issues.

1.4 Justification, motivation and benefits

This thesis will ideally help hinder nefarious cyber criminals from being able to exploit and infect users through malicious websites, users that have systems without proper security updates. The work that will be presented in this thesis will help both the defenders and the users see what kind of cyber dangers that are present on the malicious internet. By exploring what is behind the blacklists we can raise awareness and knowledge of the threat landscape that is out there for users on malicious domains.

1.5 Research questions

This research can target multiple interesting facets of the overall field of publicly available websites that are blacklisted, but it is important to limit the scope and zoom in on an aspect that is very relevant for both users and the people that are defending users in cyberspace. An overall research question is the following:

1. What kind of cyber threats and content can be found on domains that are blacklisted and labeled as malicious by DNS-BH?

   This opens for multiple sub-questions which can be split into 5 overall categories; the content of the website, the software provided by the website, the domain infrastructure, the social

[6]https://support.microsoft.com/en-us/help/13853/windows-lifecycle-fact-sheet, retrieved 3.1.19.

engineering techniques utilized by the makers of the websites and automation possibilities to aid in the detection, defense and removal of such websites.

1. Website contents:
    a. What kind of malicious executables can be found from domains that are marked as malicious by DNS-BH?
    b. What kind of connections, if any, can be found between websites labeled as malicious by DNS-BH and their content?
2. Software from websites:
    a. What can the malware type distribution of software downloaded from websites labeled as malicious by DNS-BH tell us about the websites?
    b. Are there any groups or APTs that are running these sites, and if so, can the same entity be connected to multiple domains?
3. Domain:
    a. How can domain and website infrastructure information be used to detect compromised domains?
4. Social engineering
    a. Which social engineering techniques are used to entice users to visit, interact with, freely give private information to websites labeled as malicious by DNS-BH?
5. Automation
    a. Is it reasonable to rely on automated systems to detect and categorize malicious websites for end-users?

1.6 Planned contributions

The proposed solution(s):

- An analysis of the threats from publicly accessible blacklisted domains labeled as malicious. To analyze the threats, both an extensive literature study that covers malware evolution and the cyber threat landscape from 2006 to 2018 and an experiment will be used to possibly identify and map out the threats that are present.
- An analysis of the risks that a user takes by exposing themselves to malicious websites
- A machine learning model for classifying compromised websites based on their content and domain-information

When researching the initial research domain in late 2018, there were not any similar studies which tried to build a holistic analytical report where the evolution of the necessary parts of malicious websites were studied. These parts being the operating system (in this case Microsoft Windows), malware and malicious websites.

1.7 Thesis outline

Chapter 2 An overview of the cyber threat landscape from both an academic and industry perspective. Malware taxonomy, a deeper dive into exploits, a malware timeline and the black markets so one can understand the whys and whos of the cybercrime economy.

Chapter 3 This chapter provides the necessary knowledge to understand how the Windows operating system has evolved over the years and how the security measures that has been implemented in it has affected malware development.

Chapter 4 The most relevant work done on malicious websites, how they entice users and what kind of threats are most commonly seen from such websites.

Chapter 5 An explanation of the methods that will be used to answer the research questions. This will show how existing literature can be combined with experimental data collected during the experimental part of the thesis to give insight into the cyber threat landscape.

Chapter 6 This chapter will go into detail on the experimental parts, how they were ran, what output we got and at least the results that were produced.

Chapter 7 A discussion on the results and how they are related to the evolution of malware and operating system(s) and the relevant work. Additionally, the implications for end-users and defenders. Recommendations on what are the most efficient and applicable measures towards a safer browser-experience will also be given.

Chapter 8 The summary of our findings and our final remarks on the cyber threat landscape on blacklisted malicious domains.

2 Cyber threat landscape evolution

In this section we are covering the cyber threat landscape evolution. We first have a malware taxonomy that explains the main threats and threat categories that are mentioned in this thesis. Then a timeline of the top 5 threat categories from 2006 to 2018. After the timeline we present noteworthy takeaways from the industry about IT-industry and malware developmental evolution. Lastly, we cover how big a business the cybecrime industry has become.

2.1 Malware taxonomy

Naming of malware is often independent from company to company, which is something that is seen in the reports that we will be covering later in this thesis. Therefore, we have tried to normalize the categories that are mentioned and available in their statistics when we are making our assessments. E.g. trojans are often called "misc. trojans" by Microsoft in their reports (two example reports: [29, 30]), we have normalized this to "trojans" since that is what F-Secure and Symantec are using. This thesis mainly focuses on threat categories instead of families since we want the bigger picture in the threat landscape which makes it clearer since malware families have much worse naming discrepancies.

Adware Programs with the intentions of showing advertisements that will intervene with the user flow of a machine is considered adware. Since ad-revenue is important for companies in today's day and age there has been made guidelines for the Windows operating system to how your ads can behave so that you are not classified as adware [14].

Backdoor The program makes it possible for an attacker to access a computer remotely without the user's knowledge [15].

Browser modifiers These are modifiers that change how the browser operates. These can come in various forms, some could e.g. be changing the standard search engine used so that profit is made for the attacker by using some kind of affiliate network, see section 2.4 for a more in-depth view into the profiteering side of malware.

Exploits An exploit is code that uses a vulnerability to do some action by exploiting it. This is described more in section 2.2.

Potentially Unwanted Application (PUA) Are applications that can have an impact on the computer in a way that are negative for the user. Applications that can be classified as PUAs can be operating close to the line between being non-malicious and malicious. An example application could be a program pushing advertisements as pop-ups [16], although they can operate similar to malicious applications they are not classified as that [17].

Trojan A program that is either by intent or by error capable of something outside of its intended function [18]. This could typically be a program that will *help* someone play a pirated media file, the helping program could give an attacker full access to the machine when opened and thus confidential information could be leaked. The compromised machine can also be used in a network of compromised computers, a botnet.

Trojan downloaders and droppers These are programs that download trojans or has them included in raw-format, ready to be created via e.g. an Office macro that will open Powershell and turn the raw code for the trojan into an executable. These are often included in documents such as PDF or Office documents.

Virus A program that can insert itself into other files and execute an action as defined in [18] is a virus. Further it can have multiple phases and come in many variants.

2.2 Exploit kits

When programs are developed and created, they are normally created by humans. Humans are prone to make errors and thus errors can be introduced into programs. Additionally, there could be aspects the developers did not consider when creating the program. Thus, these missing considerations and error are often called vulnerabilities. A vulnerability could be seen as a flaw in the software and this can be exploited by someone. This someone is called an attacker by Bishop in [18], chapter 23. Vulnerabilities are present in programs, operating systems, firmware for different devices and so on. The more programs that are installed on a system increases the vulnerability area as mentioned in [19] where they look at vulnerabilities and their exploitation. Additionally, they saw that for each major update of the Windows operating system the number of vulnerabilities has decreased for both the OS and the included browser, Internet Explorer. This is also explained further in depth in section 3 where the major security features are described.

An exploit kit is a set of software tools that consists of exploits and a control panel. The included exploits are often the selling point of them in addition to the design, support system and the update structure for the kit. Jones [20] mentions some typical control panel configuration options such as which exploits to use, what payload to deliver and statistics (successful and failed infections). These kits are HTML-based applications as explained in [21] and [22] figure 6. Different parameters are sent via HTML-requests from the client to the server and back again, these parameters can vary from each implementation but typically the user-agent, language, referrer, IP address and cookies [23]. These parameters are used to build a fingerprint. E.g. a machine running Windows Vista and using IE 8.0 would have the user-agent string as "*Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0)*"[1]. On the server side, the exploit kit will evaluate the user-agent and see if it is on the list of supported user-agents. If the visitor's parameters are fulfilling all requirements in the exploit kit, then it will attack the visitor in the background with the exploit(s) that are appropriate for that fingerprint. The user-agent can also be used as a way to detect robots that automatically crawl

[1]https://blogs.msdn.microsoft.com/ie/2008/02/21/the-internet-explorer-8-user-agent-string/, retrieved 5.5.19.

websites, some of these will have a user-agent string such as "weCrawler v2.0", in these cases the exploit kit will return websites that look benign [21] so that the ones responsible for the robot does not get any useful information.

Even with HTML based malware it is not easy to analyze, identify and understand what is happening. This is because they employ protection mechanisms and have been doing that for quite a while, F-Secure mentioned this feature of a botnet in 2007 already [24]. When someone tried to access the botnet's sites repeatedly, they would be DDoSed as a protection mechanism from the malware developers. Exploit kits does not employ the same measures, they utilize obfuscation [20], polymorphic obfuscation [21]; the BlackHole exploit kit even had implemented checks that scanned its binaries with antivirus engines, and if they had detection signature for it then it would change its obfuscation function so that the signatures did not catch it. They also utilize encryption as seen in a F-Secure report [25].

In addition to protection mechanisms are exploit kits utilizing advanced methods to host their exploit kits. Talos [26] studied how the Angler exploit kit utilized subdomains in tiers. Instead of being taken directly to the attack domain you were taken through one or more "gates". By having a structure like this one can have e.g. the final domains being changed quickly after a short period or after x-amount of hits. Talos saw that some domains were only up for a few minutes, but in that time a few victims had already visited them. Victims can be attacked via malvertisement and since these domains are changed to quickly for blacklists to have any impact the users are infected by these if they are running vulnerable systems. File-less payloads that reside in memory are also utilized as mentioned by [27] in addition to payloads that are sent in bit by bit and assembled on the inside of a protected system by Powershell scripts. Zero-days are also in demand by exploit kit developers. In 2013 the developer of BlackHole exploit kit bought zero-days after he announced a $100.00 USD budget for buying exploits[2], something which were seen on the detections throughout 2013 [28].

2.3 Malware evolution

This section covers industry reports from big, serious actors in the cyber security industry. Many of these firms have very talented people working for them that each year, or even twice a year, writes comprehensive cyber threat landscape reports. These are often broad and cover many areas of the landscape, thus we have had to go through and find the most relevant parts for this thesis. A deeper explanation of why we have chosen to use these reports can be seen in section 5.1. The reports that are used were chosen on availability and relevance and we chose to utilize reports from Microsoft, Symantec and F-Secure. Symantec and F-Secure are both two solid cyber security companies with a long time in the industry and thus have a big customer base, a solid sensor network and cover a big attack surface and thus has a lot of relevant data for a thesis such as this. Microsoft is primarily known for their Windows operating system which is the most used operating system in the world[3], with this reach they have by extension a huge *sensor* network through all computers running the

[2]https://krebsonsecurity.com/2016/04/blackhole-exploit-kit-author-gets-8-years, retrieved 23.5.19
[3]https://netmarketshare.com/operating-system-market-share.aspx, retrieved 24.5.19

Windows operating system. Primarily Microsoft and F-Secure are used, in some cases all 3, this is clearly shown in each table indicated in the header row under the year in question. Some reports are biannual, but there are some variations, in the tables it is not made a distinction since we are working on a yearly basis.

The following reports are utilized in the creation of the tables in this section:

Microsoft [29, 30, 11, 31, 32, 33, 34, 35, 12, 36, 22, 37, 38, 39, 40, 41, 42, 43]

Symantec [44, 16, 45, 3, 4, 7, 46, 8]

F-Secure [24, 47, 48, 49, 28, 25, 50, 51, 13, 52, 53]

2.3.1 Top 5 malware categories '06-'18

In this section we have extracted the threat categories presented each year. Some years are missing information or listings that makes us unable to discern the threat categories or estimate them based on the threat families/types listed. All blank cells are thus blank on purpose. Some years do not have enough categories listed to be able to fill all cells from top 1 to 5. The tables were not better as sideways tables; therefore, they are partitioned so that they fit best on the page's width in a chronological order.

Table 1: Colored after appearing consecutively for n amount of years

$>= 0$
$>= 3$
$>= 6$
$>= 9$

Table 2: Top 5 malware categories years 2006-2007

Top 5 malware categories	2006			2007		
	Microsoft	F-Secure	Symantec	Microsoft	F-Secure	Symantec
1	Misc PUAs		Worm	Trojan downloaders & droppers		Trojans
2	Adware		Trojans	Misc PUAs		Worm
3	Worms		Virus	Adware		Virus
4	Backdoors		Backdoor	Trojans		
5	Trojan downloaders & droppers			Backdoors		

Table 3: Top 5 malware categories years 2008-2010

2008			2009		2010	
Microsoft	F-Secure	Symantec	Microsoft	F-Secure	Microsoft	F-Secure
Trojans		Trojans	Trojasn		Trojans	
Trojan down-loaders & droppers		Backdoor	Worms		Misc PUAs	
Misc PUAs		Worm	Trojan down-loaders & droppers		Adware	
Adware			Adware		Worms	
Worms			Misc PUAs		Trojan down-loaders & droppers	

Table 4: Top 5 malware categories years 2011-2014

2011		2012		2013		2014	
Microsoft	F-Secure	Microsoft	F-Secure	Microsoft	F-Secure	Microsoft	F-Secure
Misc PUAs		Trojans		Trojans		Trojans	
Trojans		Misc PUAs		Trojan download-ers and droppers		Worms	
Adware		Adware		Worms		Adware	
Worms		Worms		Exploits		Browser modifiers	
Trojan down-loaders & droppers		Exploits		Passworld stealers and monitoring tools		Exploits	

Table 5: Top 5 malware categories years 2015-2017

2015		2016			2017	
Microsoft	F-Secure	Microsoft	F-Secure	Symantec	Microsoft	Symantec
Browser modifiers		Trojans			Trojans	
Trojans		Browser modifiers			PUAs	
Worms		Software bundles			Other malware	
Software bundles		Worms			Browser modifiers	
Downloaders and droppers		Other malware			Worms	

Table 6: Top 5 malware categories for the year 2018

2018		
Microsoft	F-Secure	Symantec
		Trojans
		Virus
		Other

From the timeline we can see that two categories that are notorious in the cyber threat landscape are worms and trojans. Worms were not as popular in 2018, but they were present consecutively from 2006 to 2017. The nature of both is such that they by their definition are infecting after a user has in some sense let them in.

2.3.2 Noteworthy takeaways from industry reports

The noteworthy takeaways were identified when going through the reports and when we saw that in some way, they had a big impact on the cyber threat landscape. This enables us to identify trends and major changes for both the IT industry and the malware *industry*. Two key topics that we think of today as given, was fleshed out in 2006 and 2007 already. Both topics are mentioned already in 2006, see table 7, the first being that the malware industry are shifting from caring about their reputation to caring about their coffers and how to fill them with gold instead of wasting their time for *fame*. The second topic are web based malicious activity, with XSS attacks being launched at the then famous MySpace.

The tables were not better as sideways tables; therefore, they are partitioned so that they fit best on the page's width in a chronological order.

Table 7: Noteworthy takeaways years 2006-2007

Topic	2006		2007	
	F-Secure	Symantec	F-Secure	Symantec
IT Industry change		MS Visual Studio has security features that can be enabled when compiling. MS also has started using security development lifecycle, which will strengthen their programs against exploitation.		IE7 implements security features to stop ActiveX exploitation.
Noteworthy malware evolution	Web worms utilizing Cross Site Scripting (XSS) on the rise. Monetary gain is the new motivation.	Monitary gain is the focus, a change from technical status. Client-side applications are targeted more.	DDoS as a service. Botnet with protection mechanism against researchers. Malvertising. Ready-made attack kits (forerunner for Exploit Kits).	Increased professionalism and the increase in use of simple trojans with droppers are two reasons for the increase in sample volume.

Table 8: Noteworthy takeaways years 2008-2010

2008			2009		2010	
Microsoft	F-Secure	Symantec	Microsoft	F-Secure	Microsoft	F-Secure
				Mozilla started notifying users of outdated versions of Flash Player.		
Conficker, advanced worm with spreading capabilities through network drives, removable drives etc. Drive-by downloads and malicious websites an increasing problem.	Increased security with email attachments drives malware authors to create drive-by downloads, often links in mails to drive-by sites. Increasing use of protection such as packing, encryption and obfuscation of known malware.	Web is the new target field of malicious activity.	Majority of deliveries by drive-by downloads.	Analysis of Conficker shows that it is written by "professional" malware developers.		

Table 9: Noteworthy takeaways years 2011-2014

2011		2012		2013		2014	
Microsoft	F-Secure	Microsoft	F-Secure	Microsoft	F-Secure	Microsoft	F-Secure
		Windows 8 with new telemtry capability to detect antivirus status.			Java 7 Update 11 sets the default security level to high, users must now actively click run before the applet is executed.		
Blackhole exploit kit, a large advanced kit with strong capability.	Nation states have become implicated or suspected in a number of cyber attacks		Wordpress is heavily targeted by exploit kit where compromised sites are used for redirects. Malvertising is rapidly growing.		Exploit kits using AES or XOR encryption on payloads. Exploit kits using more zero-days. Mevade first to utilize Tor in communication with C&C servers.		

Table 10: Noteworthy takeaways years 2015-2017

2015		2016			2017	
Microsoft	F-Secure	Microsoft	F-Secure	Symantec	Microsoft	Symantec
	Flash is being phased out from browsers.	Java 7 Update 51 requires applets to be digitally signed, Oracle also announced Java browser plugin to be deprecated by 2017.				
			IoT malware, Mirai managed biggest DDoS attacks.		Petya and WannaCry, ransomware families with advanced propagation methods.	

Table 11: Noteworthy takeaways for the year 2018

2018		
Microsoft	F-Secure	Symantec
Ransomware declined. Cryptomining on the rise. Software supply chain attacks are increasing.	Cryptomining on the rise. Ransomware declined.	1/10 URLs are malicious. Cryptomining follow coin valuation. Supply chain attacks increased. Formjacking is on the rise.

2.4 The cybercrime black markets

Cybercrime is here and has been for many years as can be seen in the timeline presented in section 2.3. The professionalism was on the rise around 2006-2008 and the shift to monetary gain instead of technical status was becoming the norm. Academics have researched the markets that cybercrime is using to sell their stolen goods (such as identities, accounts of various sorts, credit cards and so on [54, 4]). A problem is that it is hard to tell exactly how profitable and how large the criminal enterprises' revenue streams are [54] since they are not held accountable to the board and the public as if they were a company traded on a stock exchange. Therefore, most numbers of the profitability are based on estimates which have been calculated based on the volume of occurrences, marketing, posts and such on forums that are used by these criminals. Even if they are criminals, they have rules on these forums where they trade their goods, it has resemblance to the pirate code shown in the Pirates of the Caribbean movies where pirates *must* respect the *code*. The rules and behavior of such forums were researched in [55] where they found that users of these forums that followed the rules and had a good *reputation* within the forum did better business. If someone did not follow the rules they were banned or suspended from the forums. That criminals have standards are even mentioned in one of F-Secure's reports where criminals behind ransomware operations gave extensions, discounts and were overall helpful to "customers" that needed support [13].

Not only are confidential information sold on these forums, but also vulnerabilities, exploits, exploit kits, various malware, etc. This was the focus of [56] where they looked at the time of a vulnerability becoming published till an exploit that exploited the vulnerability was put on the market. For selected vulnerabilities the demand made the development faster, while for others the development was slower than what was initially thought.

Hopkins and Dehghantanha [27] investigated the ways that the exploit kit market has evolved to maximize profits. The automation that exploit kits opens for gives the attackers more time to identify interesting targets and follow up on other matters. Overall this helps increase the profitability. Exploit kits has also been spreading more and more ransomware in a way to increase profits, although that could be debated given how a big part of exploit kit business has been by selling access to machines that they have access to as seen in [57]. Grier et al. covered many ways that exploit kits are being used to profit from. Many of these methods are based on fraud, or at least by using victims in various redirection ways;

- Clickfraud in which clicks on ads are registered by the compromised machine
- Browser hijacking in which traffic is redirected either the victims search engine is replaced, or all traffic is routed via a proxy which is generating the attacker money via the traffic produced
- Use the victim as a proxy server or for hosting malware

In addition to redirection, criminals can sell successful infection and various other methods, they do after all have access to the victim's machine. Additionally, an increase in malvertising on legitimate websites are increasing the success of attackers and their profits, this combined with the new developments of content stored on third-party sites in various forms makes it harder to do forensic analysis and content can be dynamically changed very quickly.

Overall in our studies of reports and academic research that the internet is an active war zone where the criminals are ever-increasing their efforts to make more money, almost like companies on the stock exchange. That we have a cyber threat landscape like we have today is caused by many factors, but especially the improvements of fundamental security features in operating systems, web browsers and installed applications which are explained more in the chapter 3. The time when malicious URLs were few in between are over given Symantec's latest report [8] for 2018 where every tenth URL is malicious.

3 Microsoft Windows operating system and system security measures evolution

3.1 Microsoft Windows operating system security measures

As with the previous section based on reports, section 2.3, this section is based solely on Microsoft's Security Intelligence Reports. In these reports we are presented with insights into the data they are generating from all their users, every nook and cranny of the operating system is available for these authors. This makes them able to analyze malware and cyber incidents in a way no other organization like e.g. Symantec and F-Secure can. Since Microsoft are the ones developing the operating system and the tools that they include with it they can update and upgrade the different solutions they deliver and get instantaneous feedback on what is working and what is not.

Explanation of key security features introduced over the years that is shown in table 12:

ASLR - Address Space Layout Randomization Predictability is easy to exploit, therefore it was easy to exploit previous versions prior to Windows Vista since you knew where system processes were in memory. To counter this, ADSLR introduced randomness so that attackers will not know where a given system process is loaded in memory [11].

DEP - Data Execution Prevention One of the protections against buffer overflow attacks. This enables the system to mark a program's memory page(s) as non-executable so that code in these regions cannot execute [11]. So, if an attacker manages to put code into a page that are non-executable, it will not execute.

UAC - User Access Control In older versions of Windows everything normally ran with administrator privileges if the account that you were using had administrator privileges. This changed in Windows Vista so that programs which wanted to run with administrator privileges prompted the user with a box that covered the screen and asked if the user wishes to run the program as administrator.

ActiveX controls Provided by Microsoft to create applications that can e.g. extended the web browser with different features such as inter-communication between parts of the application, storage and object access [60]. They can enhance the browsing experience, but in some cases if old versions of the ActiveX controls are running then they can be exploited [61]. Typical examples of this are old versions of Adobe Flash and the Java browser plugin.

AutoRun in Windows 7 Notorious malware such as the Conficker worm exploited the AutoRun feature for USB storage devices, this caused Microsoft to re-design the AutoRun feature in Windows 7 [62]. An option to "install or run" when inserting a USB storage device made

Table 12: Windows operating system security measures evolution

Year	Major event	Description	Source
2002	/SafeSEH and /GH (compiler flags)	In Visual C++ .NET the compiler flags were introduced. These increases the application's resilience to stack-based buffer overruns.	Volume 8 [11]
2003	Scheduled security updates	Microsoft started with regular security updates every second calendar Tuesday of every month. Additionally, they opened for out-of-band security updates in critical cases.	Volume 6 [29]
2004	Windows XP SP2	A major update that introduced new features in Windows such as the Security Center, improved Windows Firewall, a pop-up blocker in IE and other configuration options that made the OS safer. DEP was one of them in addition to better heap protection through heap manager enhancements.	Volume 7 [30], Volume 8 [11]
2005	Malicious Software Removal Tool	Anti-malware software that Microsoft updates monthly through Windows Update and Microsoft Update for free to Windows users.	Volume 7 [30]
2006	Windows Vista and Windows Server 2008	Introduced new features such as UAC and ASLR.	Volume 7 [30]
2008	Windows Vista SP1 and Windows Server 2008 RTM	Structured Exception Handler Overwrite Protection (SEHOP) was implemented to stop exception handler exploitation.	Volume 8 [11]
2009	Windows 7 and Windows Server 2008 R2	Safe Unlinking in the kernel pool is an enhancement to kernel security so that malware cannot so easily exploit kernel pool overruns.	Volume 8 [11]

2009	Enhanced Mitigation Experience Toolkit	The Enhanced Mitigation Experience Toolkit (EMET) was released in 2009 to be an extra safety layer for Windows XP, Vista, 7, Server 2003, Server 2008 and Server 2008 R2.	Volume 12 [34] and [58]
2011	Change AutoRun feature in Windows XP and Windows Vista	Changed the AutoRun feature to behave like the default in Windows 7. Was pushed in an automatic update.	Volume 10 [32]
2011	Infection rates for 64-bit Windows editions surpasses 32-bit Windows editions	The infection rates Windows Vista SP1 and SP2 64-bit versions were higher than the 32-bit versions.	Volume 12 [34]
2012	Windows 8	Microsoft added real-time antimalware and antispyware to the default configuration of Windows 8.	Volume 14 [12]
2013	Windows 8.1	Machines upgraded from Windows 8 to Windows 8.1 will have their default real-time security software changed to Windows Defender if their previous software was determined incompatible with Windows 8.1.	Volume 17 [37]
2013	Internet Explorer 11	IExtensionValidation interface in IE11 introduced a new mechanism that enables security software to determine if a website is secure before allowing ActiveX controls to run, thus Java exploits cannot run on the machine.	Volume 19 [39]
2014	Updates for Internet Explorer 8 to 11	Out-of-date ActiveX controls will be blocked, such as outdated versions of Java.	Volume 19 [39]
2015	Windows 10 and Microsoft Edge	Microsoft Edge, the default browser in Windows 10, was released without support for Java or other ActiveX plugins.	Volume 20 [40]

2015	Windows 10 — Windows Defender activation	Windows Defender is also automatically activated upon installation if no other real time security product is detected. For Windows 8 and 8.1 Windows Defender also gets enabled automatically after a few days after installation if no other real-time security product is detected.	Volume 20 [40]
2015	Windows 10 — Windows Defender cloud sample submission	If enabled in Windows Defender settings, Windows Defender will upload suspicious, but undetected files, to their cloud backend where the file will be analyzed with machine learning, heuristics and automated file analysis to determine if it is malicious or not.	Volume 21 [41]
2019	Windows 10 — Windows Sandbox	Microsoft introduced a sandbox solution which creates a temporary version of Windows 10 in which you can install applications or visit websites which will be run isolated from the host.	[59]

users click it instead of browsing the storage device in the file browser. The problem was easily solved by removing the option to "install of run" when a USB storage device is detected.

3.2 Browser security

Since browsers are something which is still a fundamental part of interacting with the internet, we have included a section of the fundamentally most important security features that are implemented the last years to prevent browsers from being directly exploited. The focus has not been on the encryption side of browsers. In [63], Hein, Morozov and Saiedian made a survey on the client-side web threats and counter measures that could be applied. These attacks are targeting users directly via their browser by abusing the trust that users have to the web they are browsing. This abuse of the trust relationship is common occurrence as it utilizes basic social engineering techniques (aptly described in [10] and further mentioned in section 4.1). One of the most basic defenses is avoidance as explained by [63], and blacklists is permanent avoidance. Other avoidance measures that can be installed in browsers is trust measurement applications that have a trust factor, often calculated by users which scores a website on a scale or by good or bad. Another basic defense is limitation of JavaScript on a per website basis, in a way this expands upon the *trust rating* since it requires the user to make decisions on which websites to trust as to allow JavaScript on them. Once its enabled and deemed trustworthy anything can be run from the website, thus if an attacker takes control of the website post-JavaScript enabling, the user can be attacked from a website it previously had deemed trustworthy. These approaches all require the user to do something, which often leads to problems since they are often the most unreliable part of the chain. That is where the new advances in development for both browsers and search engines are making the choices of the user count less in the critical security decisions. Browsers are becoming more and more secure by design and search engines have automated scans and warnings of search results, examples of this are Google Safe Browsing[1] and Microsoft's SmartScreen Filter[2]. Both will be explained more in section 3.2.2.

3.2.1 Most used browsers

Figure 1 shows the browser distribution over the last 11 months. This statistic is gotten from NetMarketShare with monthly selection from 2018-05 to 2019-04 with Desktop/laptop selected as device. This means that Chrome and Firefox have 75.44% market share together. The other two in the top 4 is Internet Explorer and Edge, the former included in all Windows versions and the latter was introduced in Windows 10.

Chrome is developed by Google and have since the beginning of development pushed the envelope on performance, this is reflected in many benchmarks through the years[3][4]. Firefox is much

[1] `https://safebrowsing.google.com/`, retrieved 16.5.19

[2] `https://support.microsoft.com/en-us/help/17443/windows-internet-explorer-smartscreen-filter-faq`, retrieved 16.5.19

[3] `https://www.pcworld.com/article/3213031/best-web-browsers.html?page=2`, retrieved 16.5.19

[4] `https://www.phoronix.com/scan.php?page=news_item&px=Firefox-66-Chrome-73-Benchmarks`, retrieved 16.5.19

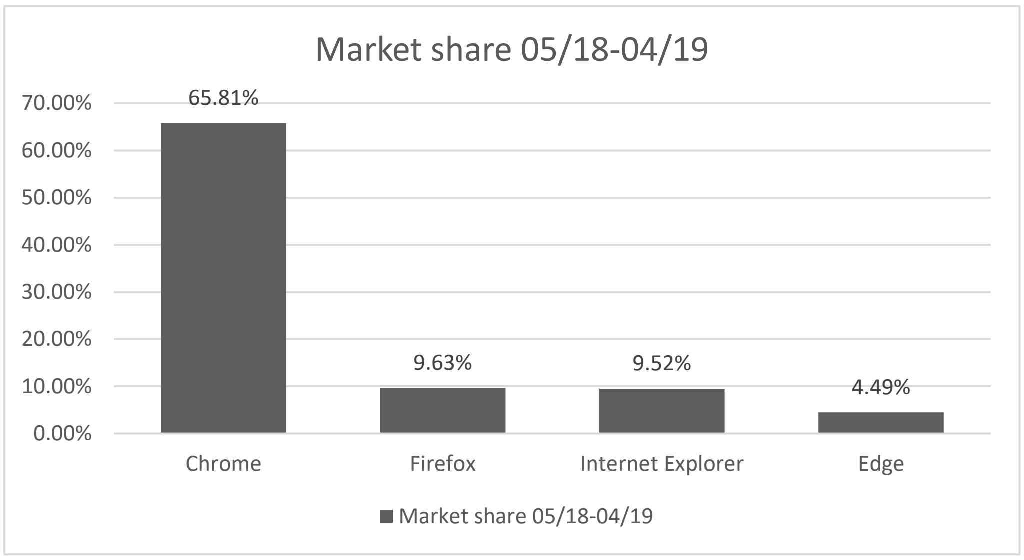


Figure 1: Browser statistics from NetMarketShare [1]

older than Chrome and has been around for very long, version 1.0 was released in 2004[5], but was available before that in pre-1.0 version. Firefox was revitalized when they released a major rewrite with version 57, Firefox Quantum[6] which introduced lots of improvements, multi-processing and a new extension API standard being two of them.

3.2.2 Security measures

A challenge when isolating pages in a browser is web applications that require communication across multiple pages, e.g. a form for an accounting program residing in the *main* page. If the form is isolated from the main page, it will not function properly. This was a concern which had to be considered when multi-processing for web browsers was initially developed and Reis and Gribble goes over it in [2]. A three-component approach was devised that was backwards compatible and made ready for modern use. In figure 2, the architecture is illustrated with key information noted with callouts on the figure. The browser kernel takes care of basic browser functionality like bookmarks, history etc., the rendering engine takes care of rendering JavaScript and CSS while plug-ins can be loaded in their own process, per plug-in. When the paper was written Chrome did not have full site isolation (published in April 2009, Chrome was still on version $\sim 1.0.154$[7]). This was enabled by default[8] in Chrome version 67 (released 2018-05-29[9]).

The multi-process design in Chrome can be further strengthened by running rendering engines in a sandboxed environment. This means that e.g. JavaScript rendering is limited to run in a limited environment (sandboxes are further explained in 5.3.4) and thus if it is malicious will be limited in what it can do.

Site isolation by Chromium's standard [64] is that the rendering process in figure 2 only contains pages from one web site. Initially when Reis and Gribble talked about it, each rendering process was on a per web program instance where a *web program instance* was defined as pages of a web program that was closely related. Thus, in the new standard this definition is concretized so that pages from only one website can be loaded, since the old wording opened for multiple interpretations. The new, concretized design thus limits the access to cross-site based actions, especially iframe exploitation which is often utilized on malicious websites.

On the other side, Mozilla had problems with an API that was giving permissions way too freely away [65]. This caused plugins to access functionality that went beyond what they used and what the new secure browser design was trying to do by limiting the attack surface it made available. To remedy this, Mozilla developed an API that could almost be used interchangeably between the most used browsers, especially Chrome [66].

Additionally, as mentioned in the introduction to Browser security, the browsers today are utilizing filtering technologies while browsing and looking up things on search engines. When a website's

[5]https://website-archive.mozilla.org/www.mozilla.org/firefox_releasenotes/en-US/firefox/releases/1.0.html, retrieved 16.5.19

[6]https://blog.mozilla.org/blog/2017/11/14/introducing-firefox-quantum/, retrieved 16.5.19

[7]https://en.wikipedia.org/wiki/Google_Chrome_version_history, retrieved 16.5.19

[8]https://www.chromium.org/Home/chromium-security/site-isolation, retrieved 16.5.19

[9]https://chromereleases.googleblog.com/2018/05/stable-channel-update-for-desktop_58.html, retrieved 16.5.19

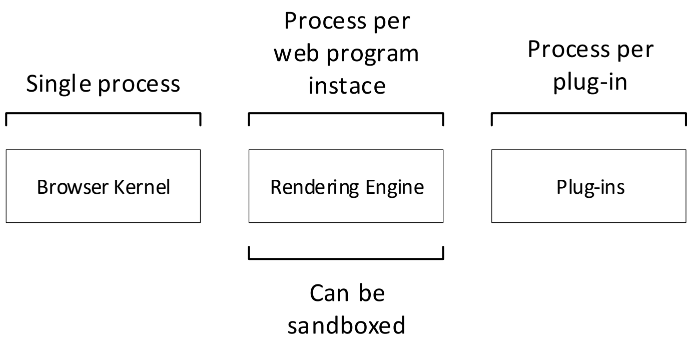


Figure 2: Illustration of multi-processor components in [2], based on Figure 5 and section 3.1, created in Visio Professional

URL is shown, the hash of that URL is checked against their database [37]. If it is contained in the database the result will be displayed, if not the URL will be scanned by their detection engines. This technique is also used on downloads where files will have their hash, or their certificate checked against the database, this is a highly efficient solution that will block malware before it can do anything. Examples of how effective e.g. SmartScreen is working is shown in their *Security Intelligence reports*, a concrete example from page 50 in Volume 23 [43]; SmartScreen detected 12.1 malware hosting sites per 1000 internet hosts worldwide in 2H17. These are detected and blocked by SmartScreen so that malware, phishing and otherwise malicious websites are stopped from being effective. In addition to these filters, browsers have also been equipped with clear warnings of websites where you enter private information that is being sent over unencrypted connections [67, 68]. This will help users understand the risks they are facing by using websites with lacking security that are handling their private information.

4 State of the Art

[1] What has been missing from the cyber security industry is a comprehensive analysis of the threats, vulnerabilities and risks that are focused solely on blacklisted malicious websites from an unprotected user's approach. A study that studies the different distributions, the threats they present, the vulnerabilities that are used and the risks they pose for the internet user. Therefore, we have conducted a comprehensive literature study in both section 2 and 3. By utilizing reports that look at the threat landscape through the view of the organization that have written them such as [29, 44, 24]. These reports identify, enumerate and explain the threats they see in their systems. The attacked and infected users are customers of the companies that are creating these reports and thus much of the information they have is sensitive and confidential, even so there is much information available in these reports. As seen in section 2, we can get a more holistic perspective on cyber threats when we can see the landscape from *normal* companies in the industry providing security services and one of the companies that are responsible for one of the operating systems used by most people in the world[2]. Combining the findings from these cyber threat landscape reports with previous academic research in the key topics for this thesis; social engineering in cyberspace and cyber threats from malicious websites. These findings will, when combined with our experimental parts, most likely help us answer the main research question as seen in section 1.5; What kind of cyber threats and content can be found on domains that are blacklisted and labeled as malicious by DNS-BH?

4.1 Social Engineering in cyberspace

One of the interesting things with malicious websites is how they attract users. The persons using computer systems are an exploitable part of the computer ecosystem which is easier than targeting e.g. the operating system itself. They are the ones setting up exploitable IoT devices that can be captured by criminals and used in botnets [69], they are also the ones that can be tricked into visiting malicious websites as seen in [70, 10].

When comparing non-expert and expert security practices [71] there were multiple interesting findings. The non-expert, the average user, were more inclined to follow advise and more norm-like security practices that were popular around mid-2000s such as browsing known websites and using antivirus solutions. Not that these practices are necessarily bad, but what is a “known” website can vary extremely much from person to person. Additionally, antivirus solutions do not necessarily protect you from everything. This is where the expert practices come into play since one of the most used practice was updating software. Software can quickly become outdated and some programs

[1]This chapter (4) is heavily influenced by the pre-project document created for IMT4205 with the title "The cyber threat landscape on publicly available websites labeled as malicious", authored by Øyvind Jensen, which was submitted December 2018.

[2]https://netmarketshare.com/operating-system-market-share.aspx, retrieved 17.5.19

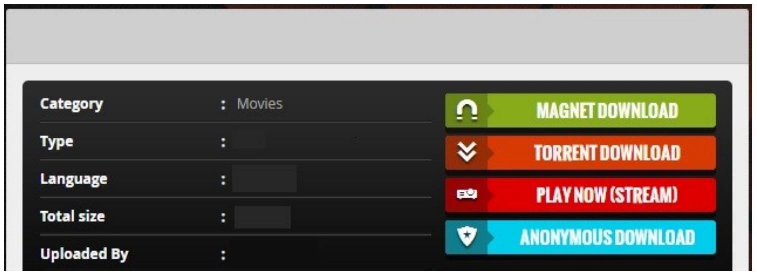

Category : Movies
Type : [redacted]
Language : [redacted]
Total size : [redacted]
Uploaded By :

Figure 3: Download button example from a file-sharing website

more than others, such as browsers and PDF-readers. By updating these, especially the browsers, the users can stay protected much more easily by e.g. getting the updates to blacklists and new features such as multithreaded support and sandboxing as mentioned in [72]. Another key aspect of personal information and account security on the internet was the handling of passwords. A non-expert was more prone to often change passwords and instead of using password managers as the experts they would try to remember them. Additionally, the expert users had 2-factor authentication high on the list of important security measures, this is most likely because it is a much safer way to secure accounts. An attacker will have a much harder time getting access to both your computer and your phone.

The social engineering attacks that are most often seen [10] are obfuscated URLs that can be spread via e.g. Twitter with its 280-character limit[3], phishing emails, drive-by downloads, spoofed websites and scareware. Spoofed websites are often part of a phishing phase [70] in which a fake version of a known website is created, the URL to that fake website is distributed by e.g. mass mailing and users that access the website can thus be lured into thinking it is the actual website it is trying to imitate. When a user has opened the website a drive-by download can happen as seen in [73, 10, 74]. This is a successful social engineered attack where a user has been tricked into visiting this website and gotten malicious files downloaded to their computer.

Drive-by downloads is not the only way a user can get malicious files downloaded on their computer. Often a user will be enticed by a download button [75] or something similar in which the graphical user interface has been tailored to exploit the trust the user has to it [10]. An example of both methods being utilized can be seen in figure 3 where the user is presented with 4 buttons, 2 from a file-sharing-site and 2 from advertisers. The "Play now (stream)" and "Anonymous download" are both the buttons that are inserted on the site by advertisers and could lead anywhere. A user could also download software deliberately from a suspicious source that is malicious without the user knowing it [76]. The downloaded software could be a variety of malware, but often it is trojans. Attackers doing social engineering has a goal in mind and that is private information because that is how they make their salaries as explained in the cybercrime black markets section 2.4.

[3]https://www.washingtonpost.com/news/the-switch/wp/2017/11/07/twitter-is-officially-doubling-the-character-limit-to-280/ retrieved 6.12.18

4.2 Cyber threats from malicious websites

The intertwinement of social engineering and cyber threats is understandable when you often must have social engineering to attack a user. Therefore, there has been put so much resources into identifying malicious activity and malicious websites so that the human factor is taken out of the equation. Browsers should have the defenses to be able to stop the user from being exploited built-in by default, but that is a hard task given how browsers and rendering of websites have been developed over the years. Sandboxed browsers that can multi-thread has become standard in the recent years[4][5], but the users that are without this protection and malware that can override or circumvent the sandboxed environments are still threats. JavaScript and its integration in browsers as seen in [77, 78, 72] make it possible for malware to attack users through their browsers with JavaScript. Often, it is malicious code that only exploits versions of browsers and in some cases particular browser versions, extension and plug-in combinations. This makes it hard to detect malicious websites causing them to go undetected and not being blocked by blacklists.

Even with the cloaking capabilities of JavaScript code, there are still ways to detect and mitigate malicious websites. In [79] they look at what is the best way for search engines to intervene against malicious websites. They come up with a solution that makes a website lose relevancy as a form for punishment when it is detected as bad. This works as a carrot to quickly respond to the infection and makes it cheaper to detect.

Another approach is to use host-based features and the URLs themselves to create machine learning models that automatically detects malicious websites. They did it in [80] with an acceptable outcome.

Taking it further than using the basic contents and host-based features one can use multiple layers as seen in [81], where both the application- and the network-layer traffic were used to detect malicious websites.

Some, [82], have taken it even further by utilizing more advanced features where they look at a combining multiple machine learning methods to build a huge associative model to detect malicious websites. This approach was mentioned by [74] where they foresaw a hybrid approach utilizing new technology to build a better detection solution. Many years before that, it was mentioned in [73] that a combination approach was in the works, but that it was yet to see daylight.

Recently there has been built a detection model based on combining content on websites and the path clients take to reach a website. This model was described in [83] where they built it on the features that can be gathered from redirection, HTML and JavaScript. One of the biggest problems with malicious websites is their evasive nature, where it is very hard to find malicious data because of environment checks (fingerprinting, see 2.2 and 5.3) that are being done by these websites. If the visitor does not have a fingerprint that matches the fingerprints supported by the exploit kit, the malicious website will hide their malicious data by e.g. redirecting to a benign site. Their model

[4]`https://wiki.mozilla.org/Security/Sandbox#Current_Status`, retrieved 6.12.18

[5]`https://chromium.googlesource.com/chromium/src/+/master/docs/design/sandbox.md`, retrieved 6.12.18

is based on a honeyclient[6] that collects data and redirect paths on these websites. They designed it to detect both malicious redirect graphs with exploit URLs and evasion redirection graphs.

Finally, Shibahara et al. [83] categorizes the most common systems for detecting malicious websites that are utilizing drive-by downloads:

Large-scale user traffic These are models that rely on many users or machines to collect and analyze websites so that a detection decision can be made. Such a system was mentioned previously, [79].

System behavior By observing the system one can analyze what happens to it and from these observations come to a classification decision based on the behavior of the system. This is done with sandboxes as described in sections 5.3.4 and 6.2.3.

Web content and redirection Models that are utilizing content and redirection information in some way is most models that have been mentioned in this related work. This is because they align most with the research questions in this thesis and thus also might be most relevant in answering them. These are: [80, 81, 74, 73, 83].

[6]A client that is created with vulnerabilities so that it will be attacked when visiting malicious websites since it will seem like it can be exploited [84].

5 Choice of methods

[1]

5.1 Literature Review

To be able to execute a thesis such as this one with multiple parts, as mentioned in section 1.6; the operating system(s), malware and malicious websites. Understanding the problem area requires knowledge of the different parts involved in the problem. Our current threat landscape is not a static one, it is evolving and has been evolving to where it is. So, to grasp the landscape we have today, retrospect is required. To get knowledge and insight of the evolution(s) we can look at reports that commercial companies give out on a regular basis. These reports date back many years and contains information about the cyber threat landscape from the company's perspective.

During the pre-project phase of this thesis some key topics were identified; social engineering, drive-by downloads, malicious website, threat landscape and exploit kits. An assortment of papers on these sections were collected and curated. These topics inspired the author to build up a strong *background* in the first sections of the thesis. An illustration of how these ties into answering the thesis' research questions can be seen in table 4.

This interweavement of the relevant topics helps with the understanding that the cyber threat landscape is largely holistic, and one part influences another, often in ways that cannot be seen immediately.

RQ 1 What kind of cyber threats and content can be found on domains that are blacklisted and labeled as malicious by DNS-BH? To answer this research question, it is necessary to understand what threats are generally on malicious websites. Understanding the threats can be done by understanding malicious websites and thus the background literature study will help with part of the answer to the research question (specifically sections 2, 3 and 4). The content on the websites is not often explained in depth in reports (such as those commercial reports or state-of-the-art research), therefore the content needs to be collected and analyzed and thus that is why the content on websites are collected. Content collection is explained in section 5.3 and the execution of it in chapter 6.

RQ 1.1.a What kind of malicious executables can be found from domains that are marked as malicious by DNS-BH? By collecting the links that are present on the websites we can identify potential links that lead to downloadable executables. These can then be scanned to

[1]The outline of this chapter, 5, and some of the wording is based on the work done in the pre-project for IMT4205, by Øyvind Jensen, which was submitted December 2018.

Table 13: Literature topics and thesis sections

Topic(s)	**Section # of this thesis**	**Description**
Social engineering, malicious website, drive-by downloads, exploit kits	4	The state of art section covers mainly social engineering in cyberspace and malicious websites. Sub-topics that appear in the research for these are drive-by downloads and exploit kits.
Exploit kits	2.2	The importance of exploit kits in malicious websites are made clear both in the academic papers and in the commercial reports.
Threat landscape, malicious website, drive-by downloads, exploit kits, social engineering	2	The commercial reports are covering the threat landscapes and in these reports, which are quite extensive, all relevant topics are covered. This shows the extensive interleaving the topics have.
Threat landscape, malicious website, drive-by downloads, exploit kits, social engineering	3	The commercial reports from Microsoft goes in-depth on both the malware landscape and on the changes that they are making on the Windows operating system. This gives a clear account of why the different measures are being made. Browsers are also playing a major part in the threat landscape of malicious websites since they are the ones displaying the content and being used as a stepping stone in the malicious activities.

see if they are malicious and their distribution can give us valuable information. How it is collected is explained in section 5.3.

RQ 1.1.b What kind of connections, if any, can be found between websites labeled as malicious by DNS-BH and their content? By utilizing topic modeling on the whole collected dataset (see section 5.4.4) we might find topics that are present on multiple websites and thus can be used to connect multiple domains together.

RQ 1.2.a What can the malware type distribution of software downloaded from websites labeled as malicious by DNS-BH tell us about the websites? Having the crawler download all files that are found on the websites will enable us to analyze, identify and classify the software and eventual malware that might be downloaded. This again can tell us which type of threat-actor that are operating the website, and we might be able to connect it to the background literature study sections.

RQ 1.2.b Are there any groups or APTs that are running these sites, and if so, can the same entity be connected to multiple domains? This is a question that can be answered by multiple parts of both the background literature study (especially 2.2, 2 and 4), the eventual actual downloaded software and the manual analysis of the websites (section 5.4.3).

RQ 1.3.a How can domain and website infrastructure information be used to detect compromised domains? Section 5.3.3 describes the different resources that can be used to get information necessary to possibly be able to answer this research question.

RQ 1.4.a Which social engineering techniques are used to entice users to visit, interact with, freely give private information to websites labeled as malicious by DNS-BH? To understand which social engineering techniques that are used to trick people on the internet we use information gathered in the background literature review, specifically social engineering in cyberspace (section 4.1). Additionally, the content (section 5.3) and the topic modeling (section 5.4.4) can be used to see which social engineering techniques that are employed. The manual analysis in section 5.4.3 will also give an insight into this since we will browse the websites of these domains and see how they are potentially (limited by the fingerprint at least) presented to the average user.

RQ 1.5.a Is it reasonable to rely on automated systems to detect and categorize malicious websites for end-users? By using the information gathered from reports (sections 2 and 3) we can see if the automated systems that are in use today are working as intended and if they are protecting users. Additionally, we can specifically see how operating system evolution has impacted malware by looking at the potentially different results from running our selection of malicious domains through our sandbox solution (see section 5.3.4 and 6.2.3).

Label	Count
Malware	1,139
Malicious	768
Total domains	1,907

Table 14: Type distribution in selection from blacklist

5.2 The overall procedure

The thesis-work will be based on collection and analysis of domains that are blacklisted by DNS-BH. To visualize the flow of the thesis-work a figure has been created, as can be seen in figure 1. Both the literature work as explained in section 5.1 and the experimental work (sections 5.3 and 5.4) will both be used together to answer our research questions and deliver an analysis on the cyber threat landscape on blacklisted malicious domains.

DNS-BH Blacklist

The blacklist that is used as a source for malicious domains in this thesis is acquired from Risk-Analytics[2]. It is a blacklist created to be used as a prevention for malicious activity, in this case, malicious websites. Our version of the blacklist is dated: 21st of February 2019 00:52.

When selecting the domains relevant for this thesis, there were two labels; *malware* and *malicious*. A website labeled with malware is generally a website that is spreading malware binaries and a website labeled with malicious is a website that is using e.g. drive-by downloads, exploit kits, etc. to attack the user directly in the browser. The distribution in our selection can be seen in table 5.2.

5.3 Data collection

To conduct the experiment a necessity is that we must collect new data from real, malicious sources. These sources are the domains explained in section 5.2. The data that will be collected is most likely:

- The domain and information possible to extract from it (see section 5.3.3)
- The website contents will be downloaded and stored to be used for analysis
- The eventual malicious scripts that are only executed in the browser environment on certain fingerprints (see section 2.2).
- The eventual executables from the domains will be downloaded:
    - This includes JavaScript files that are available, executables and such that are available from links collected by our crawler
    - Eventual drive-by downloads
    - These executables will be scanned with VirusTotal to see if they have been detected by antivirus engines

[2]http://www.malwaredomains.com/

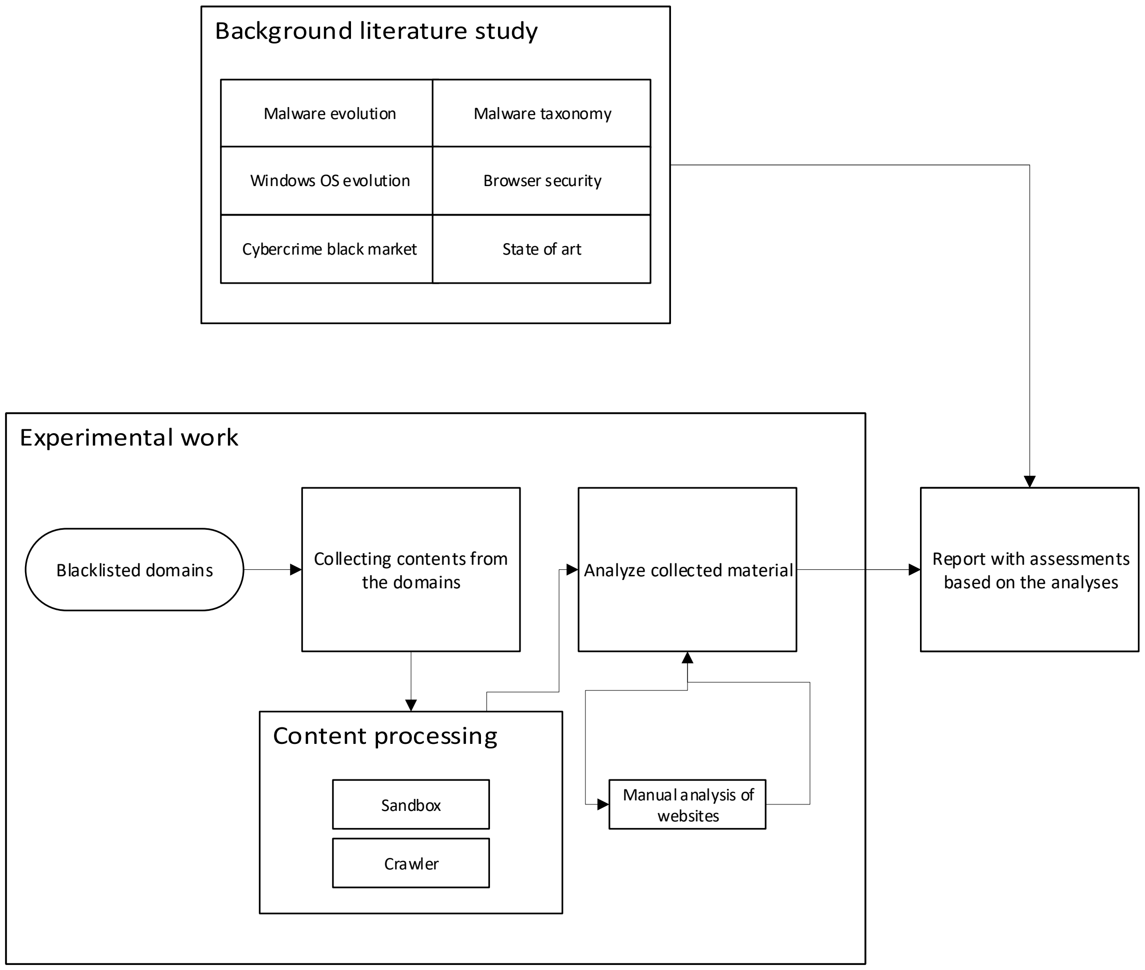
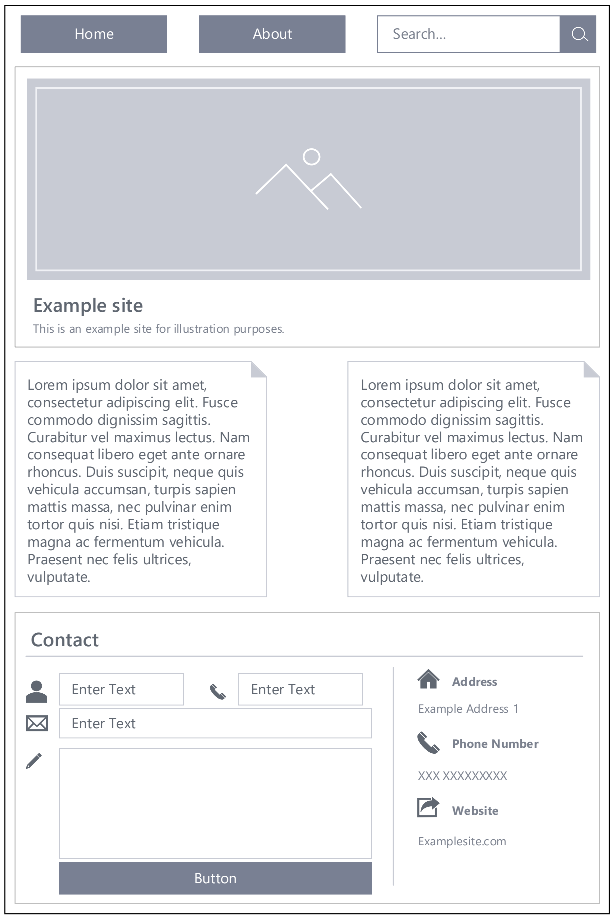


Figure 4: Illustration of the thesis workflow, created in Visio Professional

5.3.1 Crawler

Website has content in some form, often text, maybe some pictures and maybe some links. A typical website has a few sub-sections, e.g. a "home" and an "about". Traditionally these would be on their own, accessed by having a link to them in a navigational directory on the site. These would typically be stored on the server as *index.html* and *about.html*, maybe they would even use a document that contained some styles, *styles.css*. An example of such a typical website can be seen in figure 5. This example contains typical content that is useful for this thesis such as three (3) links, a picture, a search field, a contact form and three (3) textboxes. Our focus in this thesis have been on text and links contained on the websites crawled. A crawler is a program that is written to visit websites and do something "automatically". These actions are usually pre-defined, pre-programmed actions such as if an image is seen, record its location. By writing a crawler, the process of collecting data from websites can be automated as is the case in this thesis. The crawler that was designed and implemented in this thesis would hypothetically collect the following items from the website: picture, links and text.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce commodo dignissim sagittis. Curabitur vel maximus lectus. Nam consequat libero eget ante ornare rhoncus. Duis suscipit, neque quis vehicula accumsan, turpis sapien mattis massa, nec pulvinar enim tortor quis nisi. Etiam tristique magna ac fermentum vehicula. Praesent nec felis ultrices, vulputate.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce commodo dignissim sagittis. Curabitur vel maximus lectus. Nam consequat libero eget ante ornare rhoncus. Duis suscipit, neque quis vehicula accumsan, turpis sapien mattis massa, nec pulvinar enim tortor quis nisi. Etiam tristique magna ac fermentum vehicula. Praesent nec felis ultrices, vulputate.

Figure 5: Illustration of a traditional website, created in Visio Professional

The basic flow of our crawler is illustrated with the flowchart in figure 6. Think of the flow as filling a machine's fuel tank with fuel, in this case the blacklisted domains we want to crawl. When running, the crawler must check that there are domains left to scan and if there is, check if it has already been scanned since we do not want to do unnecessary work. After these checks we crawl the domain and collect its contents. On the website we might find links that lead to the website we are already on, internal links, or links that lead to external websites, external links[3]. We recursively follow the links found on the website, recording each visited link, till we either run out of links or reach the maximum depth. When finished crawling one domain in the blacklist the crawler checks if there are domains left to be crawled, if it is domains left then it repeats the previously explained process or it finishes running. The maximum depth is the number of links that the crawler follows into the domain. If a website has multiple levels, e.g. the home site in figure 5 has two navigation

[3]The crawler in figure 6 does not follow external links to make the figure and example clearer, but an augmented implementation is done and described in section 6.2.1.

options, home and about. Our crawler follows the about navigation button, it will then be on a depth of 2 since we started on 1. This means that if our max depth is 2, it will not browse any further than that.

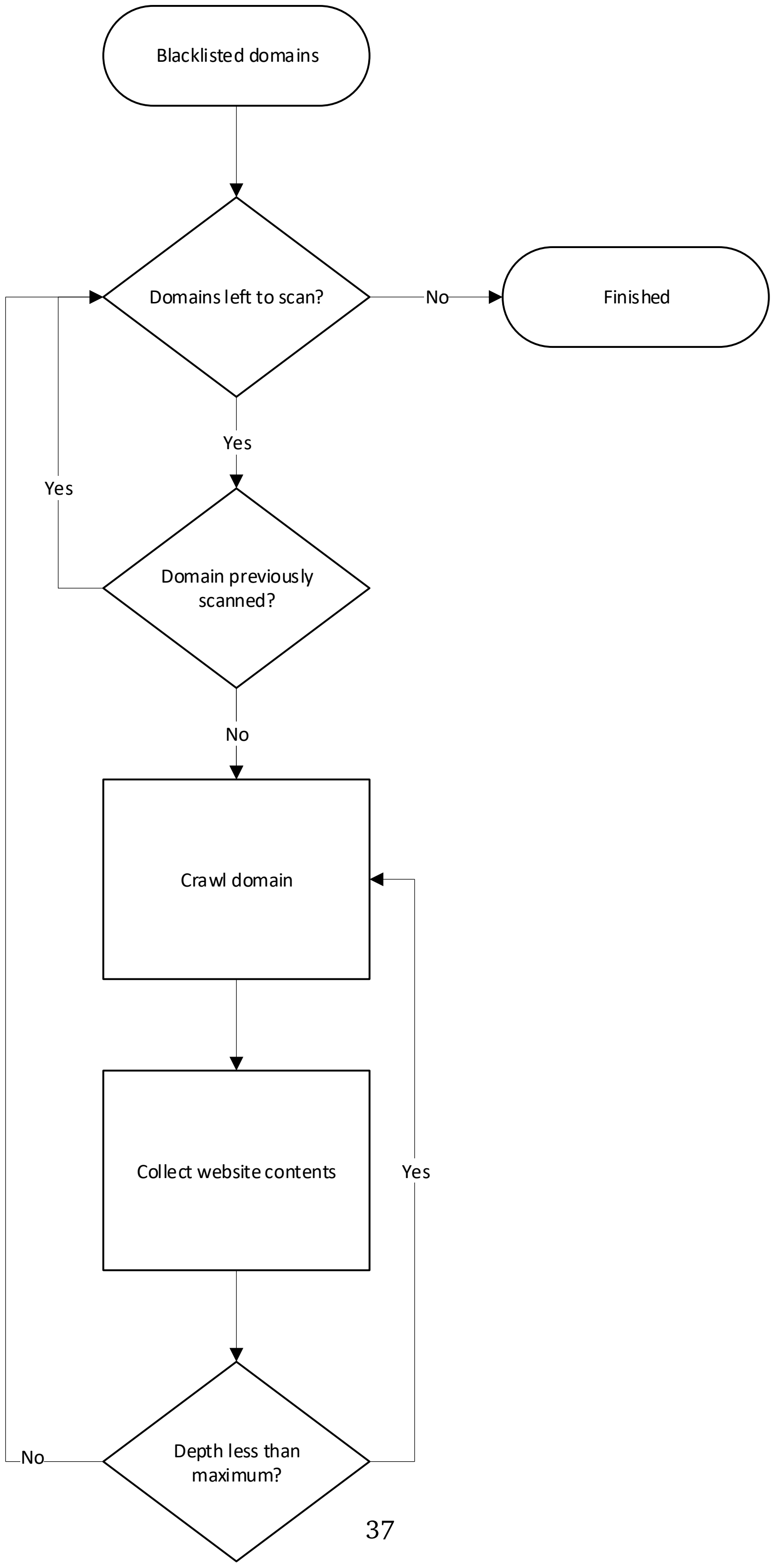


Figure 6: Flowchart illustrating the overall flow of the crawler, created in Visio Professional

5.3.2 Data storage structure

The experiment will produce a lot of data so there has to be a systematic approach when designing and setting up all the data storage. An overall look at the storage design:

- Saving data from the crawler in a folder-based way
  - One main folder that is named as the hash of the domain being saved
    - Files collected from the domain stored in the main folder
    - Additional website-content such as JavaScript, CSS, images and html-files stored in their own subfolder.

Following is two *tree* outputs from a Linux console from two different domains. One domain with external library utilization and one without. See listings 5.1 and 5.2. In the listings the files are hashed with sha-256, that is, the file name is a hash created by joining the hash of the file's URL with its file extension.

Listing 5.1: Example output from using tree command on the download folder of a website with little content such as no external JS, CSS, etc.

```
crawler/downloads/website_hash
├── a9d6126b956467447d1908896d6e0a20fd207cedfe3a63cd2e234a4d1a0bd612.txt
└── ca9d2781bb8d86eb2eabaa237e6bfe6070e3b3c1397083dc984a4affb17334b2
```

Listing 5.2: Example output from using tree command on the download folder of a website with external content

```
crawler/downloads/website_hash
├── 59435bb766cb9b20a24ba1a711c1e31fbc5231deeb4b848b70c01be7a31d2274.txt
├── a4d10f699d7d266f03fff25c83fc345aa73a75aefbc1ddb551fcd47d749f0025
├── dodgers.co.jp
│   ├── 0a5911ac__shopaccess.html
│   ├── 1d143812__index.html
│   ├── 4bb94c89__navigation.js
│   ├── 56f6362c__contact.html
│   ├── c1031439__linesale.html
│   ├── cc4b0c4f__salecalendar.html
│   ├── d28c7325__company.html
│   ├── e91a0b4f__style.css
│   ├── f2c8a491__salechirashi.html
│   └── img
├── platform.twitter.com
│   └── 7de36597__widgets.js
├── tentaklus_log.html
└── www.facebook.com
    └── plugins
```

In listing 5.2 output from a website that are utilizing external libraries are shown. The notable difference with 5.1 is that there are more folders here from external websites and a log-file.

5.3.3 Domain data collection

Doing such a study as this with websites and domains involved there are resources out there that have data on many, if not all domains. Therefore, we have used resources outside the crawler to collect more data on the domains. The external resources that have been used in this thesis are shown in figure 7 where the usage of these are illustrated. Further, in this section it is described in detail what possible information these different resources can give.

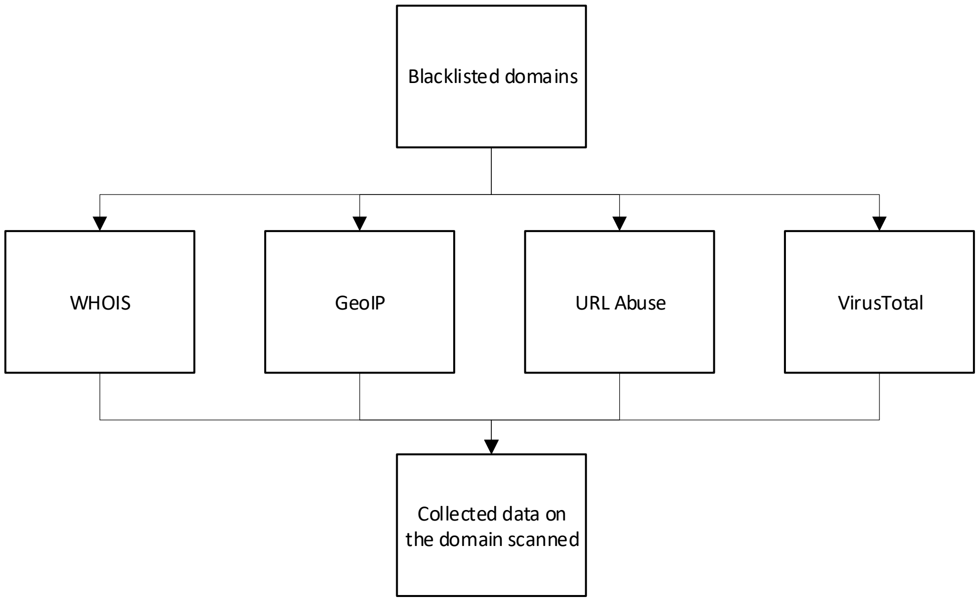


Figure 7: Flowchart illustrating the usage of domain data collection resources, created in Visio Professional

WHOIS When a domain is registered a WHOIS registry record is created, this record should ideally have information enough to be able to reach the ones responsible for the page[4]. Ideally this should work fine, but privacy and security concerns had to be taken since it might not be in everyone's interest to have their contact information in a publicly available registry just because they bought a domain. This opens for users to register with proxy or get their registrar (the ones registering their domain) to hide their contact information. As is often the case with benign abilities is that people with malicious or criminal intent can utilize it for their good. Since the option to hide information from WHOIS exist it also means that it is being utilized, ICANN has multiple studies that have investigated this listed on their sites[5]. In the end it means that WHOIS records does not necessarily give any viable information.

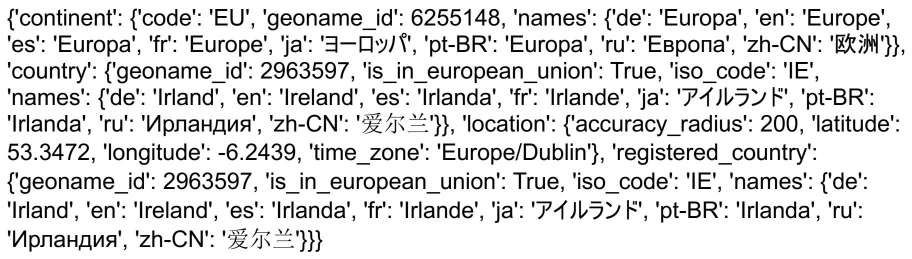
Listing 5.3: Example output from doing a WHOIS lookup on a malicious domain

```
{'contacts':
```

[4]`https://whois.icann.org/en/technical-overview`, retrieved 8.5.19.
[5]`https://whois.icann.org/en/history-whois`, retrieved 8.5.19.

```
                {
                        'registrant': None,
                        'tech': None,
                        'admin': None,
                        'billing': None},
        'domain':
                'safety.apple.com.pqdswhbg.acmvto2nbxciel7xc3lhmw9pi.download'}
```

GeoIP Another resource that can give interesting results of the geographical distribution of websites is GeoIP. It is not a way to get an exact location of a IP (as warned[6] by the provider of the GeoIP database provider used in the thesis), but it can be used to get an idea of which countries that the malicious domains are using to host their websites. In figure 8, we get GeoIP information in many languages with a multitude of keys, the information we want to get is which continent the domain belongs to. In this case it is Ireland.



```
{'continent': {'code': 'EU', 'geoname_id': 6255148, 'names': {'de': 'Europa', 'en': 'Europe',
'es': 'Europa', 'fr': 'Europe', 'ja': 'ヨーロッパ', 'pt-BR': 'Europa', 'ru': 'Европа', 'zh-CN': '欧洲'}},
'country': {'geoname_id': 2963597, 'is_in_european_union': True, 'iso_code': 'IE',
'names': {'de': 'Irland', 'en': 'Ireland', 'es': 'Irlanda', 'fr': 'Irlande', 'ja': 'アイルランド', 'pt-BR':
'Irlanda', 'ru': 'Ирландия', 'zh-CN': '爱尔兰'}}, 'location': {'accuracy_radius': 200, 'latitude':
53.3472, 'longitude': -6.2439, 'time_zone': 'Europe/Dublin'}, 'registered_country':
{'geoname_id': 2963597, 'is_in_european_union': True, 'iso_code': 'IE', 'names': {'de':
'Irland', 'en': 'Ireland', 'es': 'Irlanda', 'fr': 'Irlande', 'ja': 'アイルランド', 'pt-BR': 'Irlanda', 'ru':
'Ирландия', 'zh-CN': '爱尔兰'}}}
```

Figure 8: Example output from GeoIP from a malicious domain lookup

URL Abuse Another service that combines multiple other resources into one is URL Abuse[7]. A software created by the Computer Incident Response Center Luxembourg (CIRCL). It gives a broader understanding of the analyzed domains. In the example output in listing 5.4 we get a result array with results from different resources (such as Google Safe-Browsing, VirusTotal, HTTP redirects analysis, etc.) in addition to an "info" key. This "info" key tells us that cached content has been used, that means the domain was submitted to our local URL Abuse instance and then the output was gathered after it was finished looking up the information.

[6]https://dev.maxmind.com/geoip/geoip2/geolite2/, retrieved 8.5.19.
[7]https://github.com/CIRCL/url-abuse, retrieved 8.5.19.

Listing 5.4: Example of URL Abuse output from a malicious domain lookup

```
{'result':
[{'safety.apple.com.pqdswhbg.acmvto2nbxciel7xc3lhmw9pi.download':
 {'dns': [['217.78.1.23'], None],
  'lookyloo': 'link to lookyloo',
  '217.78.1.23':
        {
         'bgpranking':
          ['31122', '217.78.0.0/20', 'DIGIWEB-AS, IE',
          2.339071856287425e-05, 2973, 13953],
          'ipasn':
           {
            'asn': '31122',
            'prefix': '217.78.0.0/20'}}}}],
'info': 'Used cached content'}
```

VirusTotal VirusTotal is a service that gives users the ability to scan files or URLs with many commercial antivirus engines and blacklisting services[8]. It also has capabilities that lets the community contribute to e.g. URLs with findings[9]. Although for the data collection it will collect less detailed information through the free public API[10], but it will give enough information to get the number of positive detections and such so that statistical distributions can be made.

Listing 5.5: Example of VirusTotal detailed domain information

```
Entry: maliciouswebsite.com
Categories: malicious
Passive DNS Replication: date resolved and IP address
Whois lookup: Whois information
Observed subdomains: file.maliciouswebsite.com
URLs: date scanned, number of engines that detects the URL, URL
Downloaded files: date, no. of detections, file type, name
Communicating files: date, no. of detections, file type, name
```

[8] https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works

[9] An actual malicious URL analysis: https://www.virustotal.com/#/domain/shzwnsarin.com, retrieved 8.5.19.

[10] https://developers.virustotal.com/reference, retrieved 8.5.19.

Listing 5.6: Example of data extracted from a VirusTotal public API JSON response

```
scan_id: hash of scan
resource: adware-guard.com
url: http://adware-guard.com/
response_code: 1
scan_date: 2019-02-03 20:25:32
permalink: link to analysis
verbose_msg: Scan finished, scan information embedded in this object
filescan_id: None
positives: 4
total: 66
scans: All engines and their result (detected = boolean (true, false),
result = string (clean, malicious, etc))
```

5.3.4 Cuckoo Sandbox

Cuckoo[11] is an open source sandbox solution that is built using Python. A sandbox is an environment that is made for usage so that it does not matter if you make it dirty and messy. Usually sandboxes are built for kids so that they can play in them, for computers it is so that processes can run without harming the host system. The limitations of the sandbox vary from vendor to vendor, but usually a policy [18] governs it, defining what it can and cannot do. Usually this will be limitations on e.g. what files on the host system it can access, modify and run. Thus, if a malicious file is ran, it will ideally only affect the sandbox and not the host system.

What makes Cuckoo a good sandbox solution is that its open source nature makes it a natural part of any organization's malware analysis toolbox. It is also modular so that modules can be enabled, disabled, configured or created by the user(s) of that sandbox installation.

Virtual Machines

The virtual machines that has been either acquired from Microsoft in a VM-supported format or created by the author, has all been kept default. That is, the security settings that are recommended, by default, has been used and the included browsers has been used (IE or Edge). When each VM was made ready to be used for analysis they were updated with all the recent patches so that potential malware that could be able to execute on the machines were more realistic.

Configuration

The Windows VMs to be used with Cuckoo were configured as recommended in Cuckoo's documentation [12]. Network configuration[13] had to be done with care as to get it working as intended. When it is properly setup it can configure which route to use on each analysis, e.g. with direct internet access or via Tor. In this thesis, Tor routing was used exclusively. How this works in action

[11] `https://cuckoosandbox.org`, retrieved 10.5.19.
[12] `https://docs.cuckoosandbox.org/en/latest/`, retrieved 7.3.19.
[13] `https://docs.cuckoosandbox.org/en/latest/installation/host/routing/`, retrieved 7.3.19.

can be seen in figure 12 where the VMs that is being spun up by Cuckoo is interfacing with the internet through the host. The detailed VM configurations are explain in section 6.1.3 and module configuration and modification is explained in section 6.2.3.

Score

Cuckoo scores each analysis that are analyzed and reported. This score is based upon the signature severity as gathered from their source code[14]. It goes from 0 to 10, where 0 is least severe and 10 is most severe.

5.4 Data analysis

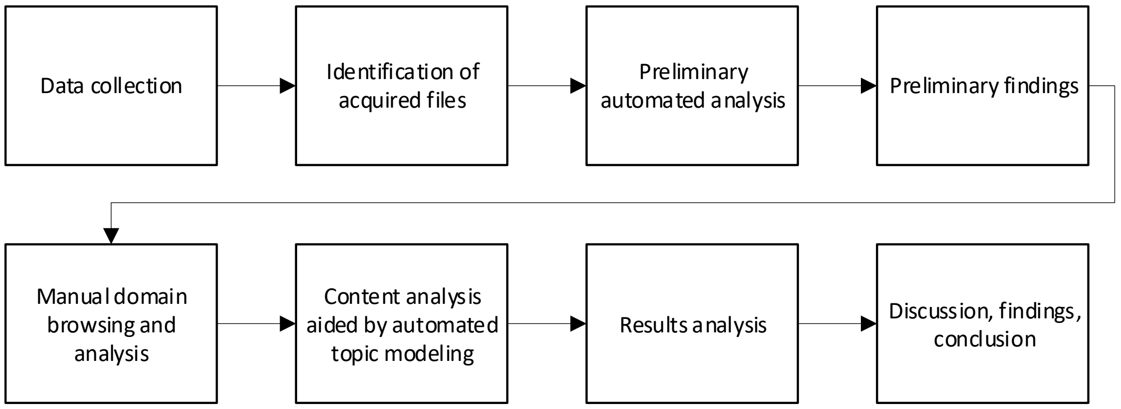

Data collection
Identification of acquired files
Preliminary automated analysis
Preliminary findings
Manual domain browsing and analysis
Content analysis aided by automated topic modeling
Results analysis
Discussion, findings, conclusion

Figure 9: Flowchart illustrating analysis stages and progression, created in Visio Professional

In figure 9, we see the overview of the progression that is planned for the analysis stages. When the data is collected, we identify the files that are downloaded, we then do preliminary analysis which will give us our preliminary findings. These will be used to conduct a manual domain analysis. With data from all these different parts we will be able to do content and results analysis on everything collected and find connections that might otherwise have not been made.

5.4.1 Identification of possible malware files

The analysis phase of the project is where the data that has previously been collected will be used. This phase will also add more data to the project. Files that previously have been downloaded needs to be scanned and this will require storing the data such that we have a connection to the domain; the hash of the domain name and hashes and information of all the files. VirusTotal will be used to lookup each file using their hash to see if the file has been seen before and if it has not then the file will be uploaded, and a Boolean tag will be added for either the file or the domain so that it can be

[14] `https://github.com/cuckoosandbox/cuckoo/blob/16b1a51acf0dc5708cfcd267c722f8aa5c56f6a3/cuckoo/core/plugins.py`, retrieved 25.5.19

scanned again later after the file(s) have been scanned by VirusTotal.

The new information that is collected on the domains will be held in a new data structure that will have:

- The domain
- The hashes of the files that were collected from the domain
- The VirusTotal reports on the file hashes

5.4.2 Preliminary automated analysis

This part of the analysis phase will go through the collected data. We will process all the possible parts that can give us information about nefarious activities on these websites in our selection of malicious domains. We create data-objects with the different data that has been collected, we output statistical properties such as average values, frequency distributions of origin countries, malware engines, words on websites, file types, etc. This stage tries to process all the random data into somewhat useful information that can be used later when trying to picture a holistic landscape.

5.4.3 Manual domain analysis

It will be beneficial for the thesis to do manual analysis of some of the domains to get a lay of the land. This way it is possible to see what kind of domains that are malicious and how they look to a normal user. This was recently done in [76] where this gave additional information about the malware and the delivery process. The information that we can obtain from this will help us understand the vulnerabilities that malware authors are targeting and utilizing. Additionally, this will help identify the risks that potential victims are exposed to (e.g. is the malicious website simply gathering passwords and usernames or is it more severe than that).

5.4.4 Content analysis

The extracted text-content will be processed so that it is possible to get statistical information from e.g. the words used and titles. It will also be necessary to see what come out from this content after being fed into a text- or content-analyzing tool that does text topic modeling. By being able to get information out from the content we get more information on what is being delivered through these websites, a holistic picture of the content on these malicious pages. It also helps us see if there are any connections between these malicious sites that will aid in our understanding of the threat landscape.

Topic modeling with LDA

Each website that is successfully crawled will most likely have some content in the body section of the HTML-document, this is the main content which is interesting for us to look at. Topic modeling can thus help us find commonalities between the websites when we feed all the bodies into a topic modeling algorithm. The algorithm that we have chosen to use is a very efficient topic modeling algorithm that was initially designed for doing topic modeling on text, the latent Dirichlet allocation (LDA) algorithm. The algorithm is using probabilities (weights) to calculate the topics of input documents [85]. By calculating the weighs for each word in a topic, the model is created by inserting words from the documents into topics and adjusting the weights for n passes.

5.4.5 Results analysis

Finally, when all the domains have been processed, contents collected, files analyzed, and the manual analysis completed we can analyze the results. They can tell us about the origin of the domains, what kind of content that were available, which kind of files that they spread, what kind of malicious activity that were being executed on these websites. This will be used to make prevention guidelines, threat-, risk- and vulnerability assessments of the domains.

5.5 Tools

When building the crawler and the necessary foundation for collecting content from websites, Python[15] will be utilized. It is a very flexible language with lots of modules for e.g. visiting websites[16], interacting with data[17]. For storing files, the normal Python file I/O library is used. Since the size of the blacklist with domains labeled as malicious is relatively small (1907, see table 5.2) it is very compact and thus to store and manipulate information we can use comma-separated format files with pandas. Pandas (which is a library for data analysis) will be used to analyze contents since it has great capabilities for data analysis. It can be used with data stored in something simple as a comma-separated file and the outputted results can be used in Excel to create graphs and do other data analysis on the results from Pandas.

5.5.1 Pandas DataFrame

Pandas is as mentioned in section 5.5, a library for Python that interacts with data. It can read data in many formats, such as flat files, comma-separated files (CSV), structured files (JSON, XML, etc) and more. In this work comma-separated files were utilized. Usually they can be as simple as:

Listing 5.7: CSV-file example

```
index,column_a,column_b
1,a,b
2,a,b
3,a,b
```

The library does not just handle file input/output, it also has a very powerful API that can be leveraged for different use-cases. You can have data in different Pandas data types (such as Series, DataFrames, Arrays, etc.). It has capabilities such as doing a $pandas.DataFrame.describe$[18] which will give descriptive stats of the data in the DataFrame. If the data is numerical it will give out the count, mean, standard deviation, minimum value, 25% percentile, 50% percentile, 75% percentile and the maximum value. If it is categorical it will give out count, unique, top and frequency. These values will enable the user to get an overview of the data.

[15] https://www.python.org/

[16] http://docs.python-requests.org/en/master/

[17] https://pandas.pydata.org/

[18] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html, retrieved 30.5.19.

Another powerful feature is the accessibility since you can work with data on a per-column basis if you would like to. This gives many options and is very flexible. An example based on how it has been leveraged in this thesis:

Listing 5.8: Column-wise DataFrame usage

```
>>> import pandas as pd
>>> domains = ['www.google.com', 'www.bbc.com', 'www.reddit.com', \
'www.nrk.no', 'www.google.com']
>>> domains_df = pd.DataFrame(domains, columns=['domains'])
>>> print(domains_df.domains.value_counts())
www.google.com    2
www.bbc.com       1
www.reddit.com    1
www.nrk.no        1
Name: domains, dtype: int64
```

In listing 5.8 the example shows a list of domains being made into a DataFrame and then the domain frequency distribution is printed out. This is an easy way to get information from large quantities of data and it can be further expanded upon, e.g. by showing only domains with a count larger than x (in this case we could output all domains with count > 1):

Listing 5.9: DataFrame value counts usage

```
>>> value_counts = domains_df.domains.value_counts()
>>> value_counts[value_counts > 1]
www.google.com    2
Name: domains, dtype: int64
```

5.5.2 VMware Workstation

To be able to run, create and use virtual machines we need to have the software for it. There are a few alternatives on the market, but the author had previous experience with VMware Workstation Pro V14. A license for the latest VMware Workstation (V15) was acquired from the VMware academic program[19]. Both Windows and Linux copies were acquired.

The VMware workstation software has many features, but only a little subset was leveraged during the thesis work. Notably the following features were used:

Snapshots Snapshots were used to ensure that the VMs used in Cuckoo were using the same snapshot for each analysis. For other VMs it was leveraged as backups in case of something going wrong.

VM Isolation VMs in VMware Workstation has the capability to share folders, drag-and-drop files and copy-paste. In cases where it was necessary to isolate the host from the VM in the best possible manner these features were disabled.

[19]https://vmware.ie.ntnu.no/

Network Virtualization For Cuckoo it was necessary to configure Network Virtualization since it has the ability to route traffic on a per analysis basis. This was done so that the VM running in the Cuckoo host was only connected to the Cuckoo host while the Cuckoo host was connected to the internet via a bridged network (explained in detail in section 5.3.4).

5.5.3 Clonezilla

Clonezilla[20] was leveraged to migrate to a SSD with larger capacity (500GB) during the Cuckoo experiment running phase. The original disk (240GB SSD) was too small to host the Cuckoo host which needed space for the 3 Windows VMs which was used.

5.5.4 Github

A coding project needs a versioning system. In this case, Github[21] provides easy access to Git versioning with free private repositories. With repositories we can track changes and read commit messages which explain what was added in a commit. This makes it easy to understand what has been going on in the different repositories. The repositories are private.

Manual crawler repository For the manual crawler a repository was created for the code and some processed data were stored in the same repository, the main repository. Most analysis were saved and updated in main repository.

Collected files An isolated repository were created for just the collected files which were stored as explained in section 5.3.2.

Cuckoo reports A repository where the Cuckoo generated reports are stored.

Cuckoo repository For automating Cuckoo operations some code had to be written. This was kept in the Cuckoo repository.

[20] `https://clonezilla.org/`, version 2.6 retrieved 8.3.19.
[21] `https://github.com/`, retrieved 8.3.19

6 Experiments and results

6.1 Environment setup

6.1.1 Host machine(s) for VMs

For the virtual machines, two hosts were used. One was mostly used when creating the Cuckoo Host VM while setting it up and configuring the necessary parts. When the Cuckoo Host VM were ready it was transferred over to the host machine that were going to run the analysis part. It was imported into VMware where it was configured as a copied VM and set to work on the new host.

Host 1 - Workstation Detailed specifications:

> **Operating System** Windows 10, Version 1803 (OS Build 17134)
>
> **Processor** AMD Ryzen 5 2600X @3.6 GHz, Turbo: 4.2 GHz, 6 cores, 12 threads
>
> **Mainboard** Asus ROG Strix X470-F, bios version 4204
>
> **Memory** Corsair Vengeance LPX DDR4 16GB 3000 MHz C16
>
> **Graphics** MSI GeForce GTX 1080 8GB
>
> **Disk(s)** List of utilized disks:
>
> - Stored Cuckoo host VM: WDC WD5000AAKS-75A7B0 500GB
> - Stored manual analysis VM and crawler VM: Samsung 850 EVO 250GB
> - Primary drive: Samsung 960 EVO 250GB

Host 2 - Dell Precision M4600 Detailed specifications:

> **Operating System** Linux Mint 19.1
>
> **Processor** Intel i7-2760QM @2.4 GHz, Turbo: 3.5 GHz, 4 cores, 8 threads
>
> **Mainboard** Original Dell, bios version A17
>
> **Memory** OEM supplied, DDR3 SDRAM 24GB 1333 MHz
>
> **Graphics** List of graphical processor units:
>
> - Intel HD Graphics 3000
> - Nvidia Quadro 1000M 2GB
>
> **Disk(s)** List of utilized disks:
>
> - Initial disk used as primary: Samsung 830 256GB

- External disk connected via USB3.0 for storing Cuckoo Host VM on: Intel 540s 240GB
- Used Clonezilla (see section 5.5.3) to migrate to a larger primary disk: Samsung 860 EVO 500GB

The virtualization that is being hosted on Host 1 - Workstation can be seen in figure 10 where all VMs are shown. All machines did not run in parallel on this system, at most one VM was on at a time. Development for the crawler (see section 5.3.1) was done on the Crawler VM. A copy of this VM was transferred to Host 2 and further updates to the codebase was updated via Git(hub) (see section 5.5.4).

On Host 2 - Dell Precision M4600 the virtualization can be seen in figure 11. The main difference from this host and Host 1 is that the manual analysis VM is not present. Additionally, the Cuckoo host VM had some small configuration changes made when running it, some of them to correct issues (see section 6.1.3) and some for performance enhancements (related to number of processors and memory, since Host 1 and Host 2 had different quantities of both). On this system as with Host 1, all VMs did not run at the same time. Instead the crawler VM ran till it was finished, then the Cuckoo Host VM ran (explained in depth in section 6.2).

6.1.2 Linux VMs used for crawling and analysis

Manual Analysis VM Used to manually analyze a selection of domains.

Detailed OS specifications: Linux Mint 19.1 Browser: Firefox 66.0.1 64-bit

Detailed VM specifications: Memory (RAM): 8GB, Processors: 6, Disk capacity: 20GB. Network adapter: Bridged network connection (connecting the VM directly to the external network).

Crawler VM Used for the crawler that was created in this thesis. See section 5.3, 5.3.1 and 6.2.1 for more details on the crawler.

Detailed OS specifications: Linux Mint 19.1

Detailed VM specifications: Memory (RAM): 19.5GB, Processors: 4, Disk capacity: 40GB. Network adapter: Bridged network connection (connecting the VM directly to the external network).

6.1.3 Windows VMs used for sandboxing analysis

Windows XP SP3 Acquired from Microsoft directly in the form of Windows XP Mode[1]. This was imported into VMware Workstation where the size of the virtual disk was changed to be dynamically allocated so that instead of taking the default 127GB on disk, it only takes up the size being used.

Detailed OS specifications: Windows XP, Version: 5.1, Build: 2600, Service Pack: 3, Browser version: Internet Explorer 8.0 (Final). Note that the TLS support for Windows XP is limited to 1.0 [86] so that websites that are utilizing encryption higher than this will not work if they

[1] https://www.microsoft.com/en-us/download/details.aspx?id=8002, retrieved 27.5.19

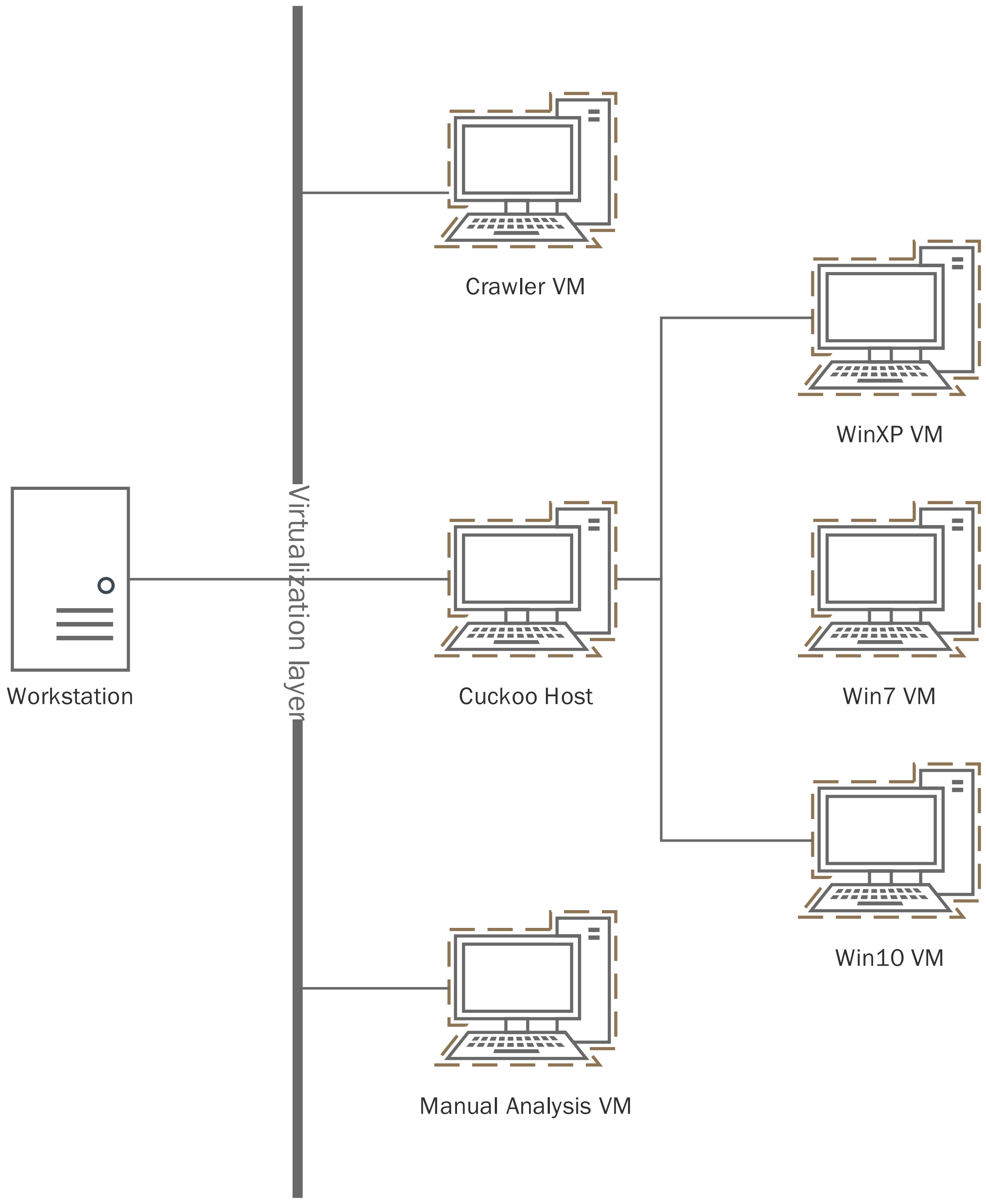


Figure 10: Illustration of the virtualization layer on the Workstation host machine, created in Visio Professional

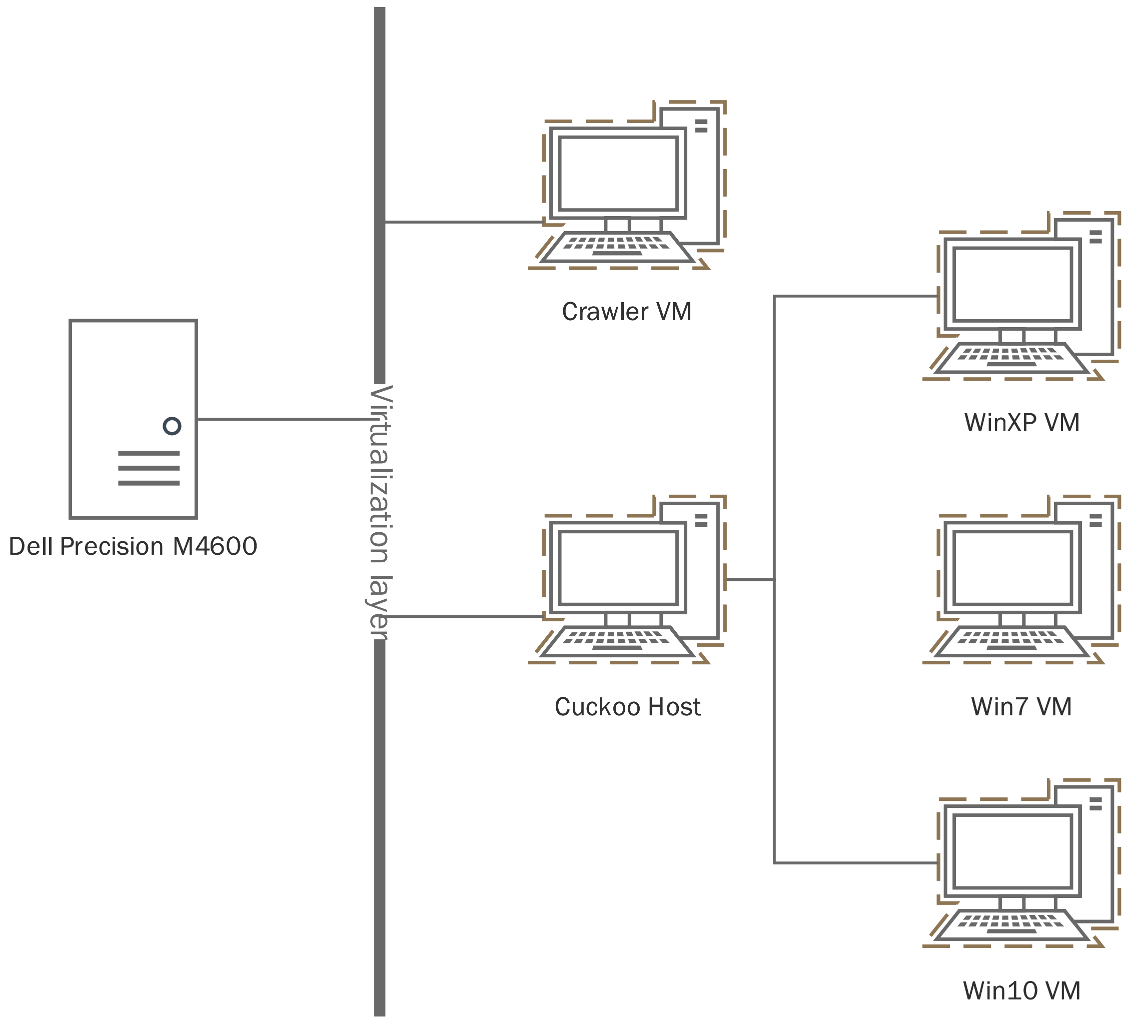


Figure 11: Illustration of the virtualization layer on the Dell Precision host machine, created in Visio Professional

do not have backwards compatibility added for Windows XP.
Detailed VM specifications: Memory (RAM): 2GB, Processors: 1, Disk capacity: 127GB. Network adapter: Connected to a custom Host-only virtual network adapter, VMnet1.

Windows 7 Acquired directly from Microsoft[2] with IE11 preinstalled as a VM for VMware. Updated to the latest available recommended updates after being acquired from Microsoft on 25.2.19.
Detailed OS specifications: Windows 7 Enterprise, Build: 7601, Service Pack: 1, Browser version: Internet Explorer 11.0.
Detailed VM specifications: Memory (RAM): 4GB, Processors: 2, Disk capacity: 40GB. Network adapter: Connected to a custom Host-only virtual network adapter, VMnet1.

Windows 10 Acquired directly from Microsoft[3] with Edge preinstalled as a VM for VMware. Updated to the latest available recommended updates after being acquired from Microsoft on 25.2.19.
Detailed OS specifications: Windows 10, Version: 1803, Build: 17134, Browser version: Microsoft Edge 42.17134.
Detailed VM specifications: Memory (RAM): 4GB, Processors: 2, Disk capacity: 40GB. Network adapter: Connected to a custom Host-only virtual network adapter, VMnet1.

Cuckoo Issues

While running the Cuckoo part of the thesis a few issues arose which was not accounted for beforehand.

- The disk which hosted the Cuckoo host ran out of space since VMware was configured to swap some VM memory (from RAM to disk). This was due to the host's disk was almost full so that VMware Workstation had its capacity threshold triggered so that it halted execution of the VM. This was remedied by increasing the capacity of the host disk. The system was cloned from the old to the new disk.
- Thermal issues arose twice before it was fixed by elevating the laptop on a laptop-stand in addition to having an external desk-fan that was helping the internal fans by blowing fresh air to the machine.

6.1.4 Network diagram

Since this thesis is analyzing malicious websites, we must actually visit them. This means typically that we will connect with a malicious server which will expose information about the visitor. The most relevant information is the IP-address of the visitor and other environmental features that creates a fingerprint. Thus, to ensure anonymity and security during the thesis execution some precautions were taken. See figure 12 for a detailed diagram showcasing the network configuration in detail.

[2] `https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/`, retrieved 27.5.19
[3] Same as the Windows 7 VM

VPN A VPN was utilized to protect the origin. The VPN location was set to the random assortment of servers in Sweden for the crawler and Cuckoo. For the manual analysis, servers in Denmark was utilized.

Tor For Cuckoo and the crawler, Tor was used in addition to the VPN. The default Tor configuration was not changed except for making it possible to utilize it as a proxy server for the crawler.

6.1.5 Software

The most notable software installation used in the thesis are listed below in no particular order or separation (between systems). For a detailed list of Python modules used please see Appendixes A and B.

- Python 2.7.15rc1[4]
- Python 2.7.15[5]
- Python 3.6.7[6]
- Mullvad VPN 2019.2 (Both Linux and Windows version used) [7]
- git 2.17.1[8]
- tor 0.3.2.10-1[9]
- tor-arm 2.0.4-3[10]
- pywebcopy 5.0.1[11]
- requests-html 0.9.0[12]
- stem 1.7.1[13]
- fake_useragent 0.1.11[14]
- Pandas 0.24.1[15]
- Cuckoo 2.0.6.2[16]
- postgresql 10.6 (10+190)[17] (Cuckoo dependency)
- MongoDB 4.0.6[18] (Cuckoo dependency)

[4]https://www.python.org/, retrieved 9.5.19.
[5]https://www.python.org/, retrieved 9.5.19.
[6]https://www.python.org/, retrieved 9.5.19.
[7]https://mullvad.net/en/, retrieved 9.5.19.
[8]https://git-scm.com/, retrieved 9.5.19.
[9]https://2019.www.torproject.org/index.html.en, retrieved 9.5.19.
[10]https://2019.www.torproject.org/index.html.en, retrieved 9.5.19.
[11]https://pypi.org/project/pywebcopy/, retrieved 9.5.19.
[12]https://html.python-requests.org/, retrieved 9.5.19.
[13]https://pypi.org/project/stem/, retrieved 9.5.19.
[14]https://pypi.org/project/fake-useragent/, retrieved 9.5.19.
[15]https://pypi.org/project/pandas/, retrieved 9.5.19.
[16]https://cuckoosandbox.org/, retrieved 9.5.19.
[17]https://www.postgresql.org/, retrieved 9.5.19.
[18]https://www.mongodb.com/, retrieved 9.5.19.

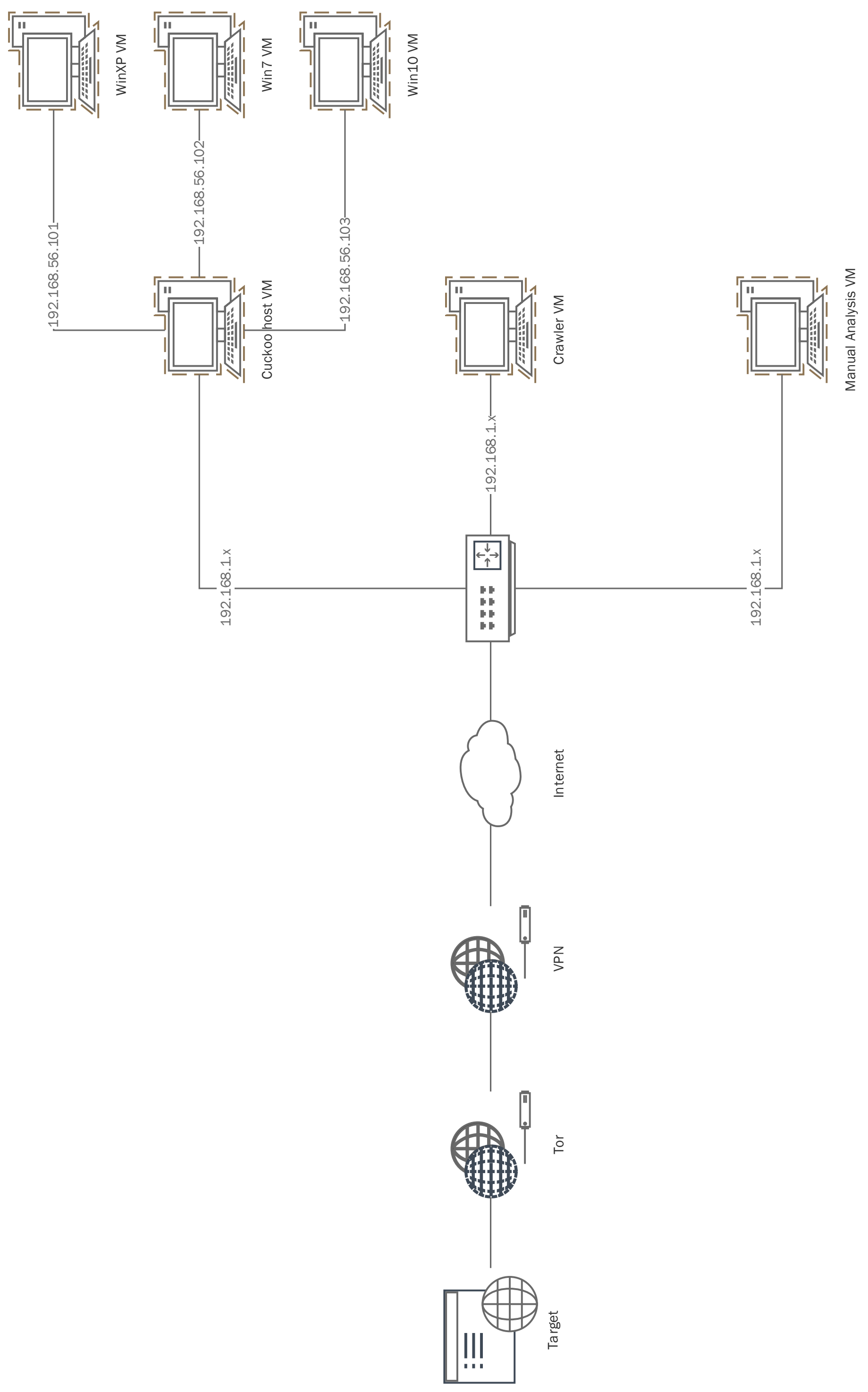


Figure 12: A network diagram of the internet-connected systems in this thesis, created in Visio Professional

6.2 Collection phase

The collection phase consists of three parts; crawler, domain data collection and Cuckoo as explained in section 5.

6.2.1 Crawler

Implementation

In listing 6.1 we can see an extracted part of the crawler's main function. This shows how the *visit_site* function (figure 13) is being utilized. The selected domains from the blacklist are scanned one by one, each being fed into the visit_site function. On the first run a few temporary settings are set for the pywebcopy module which is a module that is being used to take a copy of the website. An example where it collected additional data can be seen in listing 5.2. Listing 5.1 shows where it did not manage to collect any data. The Tor relay is also set up on first run of the crawler's main function to ensure that a connection to a Tor-proxy is live and working.

If a domain has already been checked then it goes to the next domain in the list to be scanned, this is useful when restarting the crawler so that it does not have to scan domains that has been scanned. The user-agent (explained in depth in section 2.2) is refreshed before visiting a new domain. When the index of our execution is $index \ mod \ 10 == 0$ we run the save procedures which are shown in depth in Appendix C, this is implemented to ensure that data is not lost upon crashes or timeouts.

Listing 6.1: Excerpt from the main crawling function

```python
def main():
        with open('selected_malware_domains.txt', 'r') as infile:
                malwaredomainlist = infile.read().splitlines()

        firstloop = True
        for index, domain in enumerate(malwaredomainlist):
                if firstloop is True:
                        firstloop = False
                        # Set options for pywebcopy
                        pywebcopy.config.setup_config(domain, \
                                                                download_path, "name")
                        pywebcopy.config['zip_project_folder'] = False
                        pywebcopy.config['BYPASS_ROBOTS'] = True
                        # Setup and signal the Tor proxy,
                        # acquiring a connection the Tor relay
                        with Controller.from_port(port=9051) as c:
                                c.authenticate()
                                c.signal(Signal.NEWNYM)
                if domain in checked_domains_list:
                        continue
                # Change to a random user agent on the new domain to be scanned
                headers['User-Agent'] = ua.random
                visit_site(domain, 0)

                if index % 10 == 0:
                        # Run save procedures.
                        # Used to be able to recover easily from problems.
```

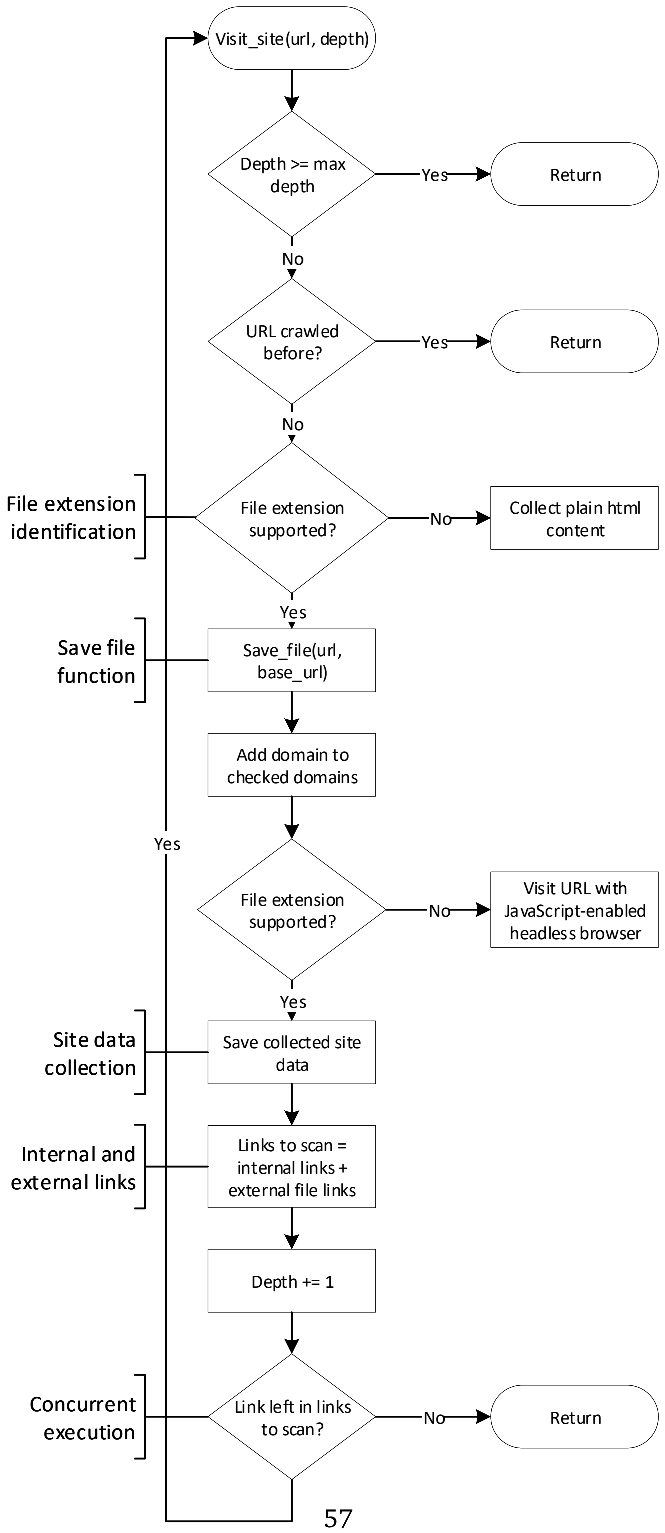

Visit_site(url, depth)
Depth >= max depth
Yes
Return
No
URL crawled before?
Yes
Return
No
File extension identification
File extension supported?
No
Collect plain html content
Yes
Save file function
Save_file(url, base_url)
Add domain to checked domains
Yes
File extension supported?
No
Visit URL with JavaScript-enabled headless browser
Yes
Site data collection
Save collected site data
Internal and external links
Links to scan = internal links + external file links
Depth += 1
Concurrent execution
Link left in links to scan?
No
Return

Figure 13: Flowchart of the crawler implementation, created in Visio Professional

The flowchart in figure 13 shows the *visit_site(url, depth)* function which was called in the main function (listing 6.1) of the crawler. This function is the one doing the main crawling work. It visits the URL which is given as an argument and extracts data directly from the websites, if the input is to a website and not directly to a file, in which case the file is downloaded by calling the *save_file(url, base_url)* function. Further if it is to a website it will also visit them with a headless browser with JavaScript support. This is to ensure that content from websites that additional content or redirects that are only enabled with JavaScript is also collected by our crawler (see section 2.2 about the workings of exploit kits). The following description listing will explain the text callouts to the left in the flowchart (figure 13):

File extension identification The extraction of the file extension of a link is attempted by seeing if there is an extension in the URL and if the extension is in the list of supported URLs. This list of supported file formats is as follows:

```
office_file = ['.odt', '.pdf', '.rtf', '.md', '.docx', '.tex',\
         '.txt', '.doc', '.xls', '.ods', '.xlsx', '.csv', '.sql']
executables = ['.cmd', ',bat', '.exe', '.py', '.vbs', '.msi',\
         '.dll', '.jar', '.wsf', '.bin', '.apk', ]
supported_file_extensions = ['.png', '.jpg', '.jpeg', '.js',
         '.mp3', '.mp4', '.swf', '.gif', '.bmp', '.ttf', '.zip',\
          '.7z', '.z', '.tar.gz', '.rar']
supported_file_extensions.extend(office_file)
supported_file_extensions.extend(executables)
```

The typical office files are collected to see if some websites in the blacklist selection are distributing malware that are utilizing social engineering by hiding malware in "benign" word-documents or PDFs. A few typical executable file extensions are included to see if these sites have direct links to malicious executables. Image-files, mp3-audio, mp4-video and flash-files to get a picture of the content-distribution. Flash has also been notorious in the past by being exploited by malware (see section 2.3). Archive file formats can contain compressed malicious files therefore, we also collect these.

Save file function An overall description of the save file function is given below:

1. Create download folder for the base_url if it does not exist
2. Create file name as a hash_url(url) + file_extension (extrapolated from the incoming URL)
3. Get headerinfo, try to determine the file extension from this information (by using the Content-Type header[19] - if it was possible to determine then change it to the one defined by the HTTP header

[19] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type, retrieved 13.5.19.

4. Additionally, if the header contains the 'Content-Length' entry[20] then the maximum file size supported is 10 MB
5. Download the file:
   - If the file-extension is either html or htm: download with pywebcopy
   - If not: download file as a byte-stream and write it to the file path created with (download folder + file name)

Listing 6.2: Utility function, hash_url, used by save_file function

```
def hash_url(url):
        url = url.encode('utf-8')
        return hashlib.sha256(url).hexdigest()
```

Listing 6.2 shows how the hashes of the URLs were created. This was used when storing files in accordance with section 5.3.2.

Site data collection In addition to downloading files, the thesis is focusing on the content. To understand content a basic description was given in the introductory part of section, 5.3.1, where different example parts on an example website were shown. The previously explained *save_file* function saves the pictures and eventual files, but the site data collection just saves the content in plaintext, such as an image's link or a website's script in text form. The data that is collected in plain text is as follows:

domain This is the domain that specifically was visited when creating this entry in the site data collection dataframe.

base_url The base url is used to identify the "base" domain, to extract this, the urllib.parse library[21] in Python is utilized. Specifically urllib.parse.urlsplit[22] was used. This splits a URL into components, where *netloc*, defined as "Network location part", was used to create the base_url.

user-agent The user-agent that was used to visit the domain in question is stored for possible statistical possibilities.

destination The final destination that was detected when running the redirect component of the *visit_site*. (The redirect part visits the domain with redirects disabled and then uses the url.next component if it is present till it is not. Thus, the final URL is then the destination.)

title The title, if present, of the website.

internal_links If the base_url is part of the URL, then it is considered an internal_link.

external_links Links that are not in internal links are considered external links.

[20]https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Length, retrieved 13.5.19.
[21]https://docs.python.org/3.6/library/urllib.parse.html, retrieved 13.5.19
[22]https://docs.python.org/3.6/library/urllib.parse.html#urllib.parse.urlsplit, retrieved 13.5.19

redirect_history The list of URLs seen when redirecting (as explained for the *destination* part of site data collection).

navigation The navigational text contained within the HTML-tag *nav*[23].

body_text The text that are contained in the body section[24] of the website, if any.

raw_html The raw html that are on the website, if any. This means that html-syntax are kept so that the webpage could be seen if e.g. the content in *raw_html* was opened as html in your browser.

pictures The links to eventual pictures on the website are stored.

scripts The eventual scripts are stored. This can be used e.g. for seeing if any obfuscation techniques or malicious scripts are embedded in the page.

inputs The eventual inputs on the website are stored. Could be used to identify websites that have input-forms that are potentially using these to gather personal information from the user.

body_text_js Text contained in the body that are seen by the JavaScript enabled browser.

raw_html_js Raw html that are seen on the website by the JavaScript enabled browser.

inputs_js Inputs that are seen on the website by the JavaScript enabled browser. These could be different than the ones seen without JavaScript enabled, e.g. in a way to make the website harder to visit by robots.

Internal and external links See *Site data collection*. Additionally, the links that are added to be visited are the internal links and the external links which are identified as leading to a file with a file extension that are supported in the crawler. If that is not the case, then it is not added to the list of links to visit.

Concurrent execution To speed up the crawling process, concurrency were utilized. This was seen to be very efficient in [87] chapter 17. The usage of ThreadPoolExecutor[25] were implemented in the same fashion as seen in the example on Python's documentation page for ThreadPoolExecutor[26]. The exact implementation can be seen in listing 6.3. In the excerpt it is shown that the *future_to_url* is not being used to get any returns from the threads, that is because the visit_site function does not return any data, it only stores data in globally accessible objects via append so that even if two threads crawl the same URL (by chance of concurrency) it does not corrupt any of the data since a duplicate entry for that *domain* will be stored in the *Site data collection* dataframe.

[23] `https://developer.mozilla.org/en-US/docs/Web/HTML/Element/nav`, retrieved 13.5.19
[24] `https://developer.mozilla.org/en-US/docs/Web/HTML/Element/body`, retrieved 13.5.19
[25] `https://docs.python.org/3.6/library/concurrent.futures.html#threadpoolexecutor`, retrieved 13.5.19
[26] `https://docs.python.org/3.6/library/concurrent.futures.html#threadpoolexecutor-example`, retrieved 13.5.19

Listing 6.3: Concurrency source code from visit_site function

```
if len(links_to_scan) == 0:
        # No links to scan
        return
workers = min(max_workers, len(links_to_scan))
with concurrent.futures.ThreadPoolExecutor(workers) as executor:
        future_to_url = \
                {executor.submit(visit_site, url, depth): \
                        url for url in links_to_scan}
```

Execution

When designing something to be automated one must think about what happens if the unexpected happen. The procedures to recover from an unforeseen crash or a forced shutdown. That is why the save procedures in the implementation section were mentioned. Ideally the automated crawler is launched and runs till there are no domains left to crawl. That did not happen. Many of the domains on the blacklist were not being hosted anymore so the crawler timed out in various sections of the implementation, and in some cases, e.g. when trying to render a website with the JavaScript enabled render it could crash, or when trying to request a domain that did not respond. To continue running in the instances where the crawler's exception handling[27] failed, we utilized a basic bash-script that would restart the crawler when an error message was detected from the crawler.

Limitations

- The user-agents utilized by the crawler is not limited to ones found explicitly in exploit kits.
- Due to time constraints the crawler was not run with different depth constraints and only one run was done with $max\_depth = 2$.
- The external links which were followed were only links that were identified to have a supported file extension; thus we might have missed some.
- Redirection was not limited and thus it could have been using much time redirecting, but during the development and execution of the crawler it was not detected as a problem or limitation and thus, the author did not implement it.

6.2.2 Domain data collection

Implementation details

One limitation with the public API access to VirusTotal is that you can only send a request per 15th second, 4 per minute. The other external data sources did not have such a limitation. To make it the most efficient collection possible, the VirusTotal gatherer ran on the Crawler VM on Host 1 while WHOIS, GeoIP and URL Abuse ran simultaneously on the Crawler VM on Host 2 (Host 1 and 2 are in detail explain in section 6.1.1). This was done so that it was not required to have things such as: sub-processing of VirusTotal and special configurations of the Tor-proxy setup.

[27] `https://wiki.python.org/moin/HandlingExceptions`, retrieved 25.5.19

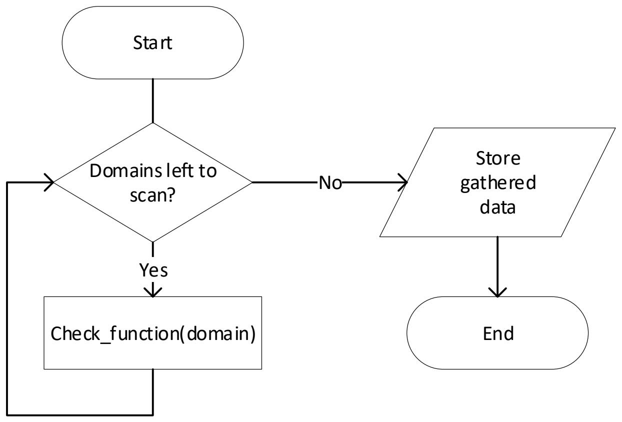


Figure 14: Illustration of the external gatherer scripts flow, created in Visio Professional

Figure 14 shows the general flow of the scripts that are gathering data from external resources. The $check\_function$ is either of the functions explained below (WHOIS, GeoIP, URL Abuse or Virus-Total). Domains are kept in a list that resides in memory since it is quite few in total, so a for-loop is used to iterate through the domains and call the necessary functions in each iteration, for an example see listing 6.4.

Listing 6.4: For-loop illustration

```
>>> domainlist = ['www.google.com', 'www.ntnu.no', 'www.duckduckgo.com']
>>> for domain in domainlist:
...     print(domain)
...
www.google.com
www.ntnu.no
www.duckduckgo.com
```

WHOIS When running the gatherer script an issue arose with timeouts that was not terminated by default with the WHOIS library[28] that was utilized. To solve this issue from hampering the execution a timeout library[29] was utilized to force WHOIS lookups to finish within the given timeout. When a timeout occurred an "empty" WHOIS object was inserted into the list of WHOIS lookups. This held the URL that timed out and the string "cannot get whois".

[28]https://github.com/joepie91/python-whois, retrieved 8.5.19.
[29]https://pypi.org/project/stopit/, retrieved 8.5.19.

Listing 6.5: WHOIS implementation and usage

```
# A way to force timeouts when doing whois-lookups
@stopit.threading_timeoutable(default="not finished")
def url_whois_check(url):
        try:
                urlwhois = pythonwhois.get_whois(url)
        except Exception as e:
                print('e' + ' ' + str(url))
                empty_whois = {'url': url, \
                        'whois': 'cannot get whois'}
                return empty_whois
        urlwhois.pop('raw', None)
        urlwhois['domain'] = url
        return urlwhois


# Inside the main loop of the gatherer script.
# The timeout used were 20 seconds, seen when calling the function
whois_data = url_whois_check(domain, timeout=20)
if whois_data == "not finished":
        notfinishedstring = f"Whois call for {domain} \
                                did not finish before timing out."
        whois_list.append(notfinishedstring)
        print(notfinishedstring)
else:
        whois_list.append(whois_data)
```

GeoIP The default socket library in Python was utilized to get host (IP) by name[30].

Listing 6.6: GeoIP usage

```
def url_geoip_check(url):
        try:
                ip_of_url = socket.gethostbyname(url)
        except socket.gaierror as e:
                print('e' + ' ' + str(url))
                empty_geoip = {'url': url, \
                        'geoip': 'cannot resolve ip'}
                return empty_geoip
        reader = geolite2.reader()
        geoip_data = reader.get(ip_of_url)
        geolite2.close()
        return geoip_data
```

[30]https://docs.python.org/3.6/library/socket.html#socket.gethostbyname, retrieved 8.5.19.

URL Abuse URL Abuse is implemented as a standalone module that can be imported and used in Python. The server must be running and then lookups can be executed.

Listing 6.7: URL Abuse usage

```
def url_abuse_check(url):
        urlabuse = PyURLAbuse('http://0.0.0.0:5200')
        response = urlabuse.run_query(url)
        return response
```

VirusTotal The Requests[31] library was used to communicate with VirusTotal's API.

Listing 6.8: VirusTotal usage

```
def virustotal_check(url):
        headers = {
                "Accept-Encoding": "gzip, defalte",
                "User-Agent": "gzip, Python3 unit"
        }

        parameters = {'apikey': 'insert apikey', 'resource': url}
        response = requests.post(\
        'https://www.virustotal.com/vtapi/v2/url/report', \
        params=parameters, headers=headers)
        vt_js = response.json()
        time.sleep(15)
        return vt_js
```

Execution

As with the crawler, some fault tolerance was implemented with adding which sites were checked, but since we were utilizing servers and services where we just checked the malicious domains, we were not relying on them resolving per se. The most important thing was having a list of checked websites for VirusTotal so that if our automated script were to crash, we would not have to start from scratch since the public VirusTotal API has a restriction of 4 requests each minute, or 1 per 15th second.

6.2.3 Cuckoo

Modules

In the beginning of section, 5.3.4, the modularity of Cuckoo was mentioned. This thesis used the following Cuckoo configurations (important changes listed; default value listing kept to a minimum):

Listing 6.9: Cuckoo configuration (cuckoo.conf)

```
[database]
connection = postgresql://cuckoo:cuckoo@localhost:5432/cuckoo
```

[31] https://2.python-requests.org/en/master/, retrieved 8.5.19.

Listing 6.10: VMware configuration (vmware.conf)

```
[vmware]
machines = cuckoo7,cuckoo10,cuckooxp
interface = vmnet1

[cuckoo10]
ip = 192.168.56.103
tags = win10
[cuckoo7]
ip = 192.168.56.102
tags = win7
[cuckooxp]
ip = 192.168.56.101
tags = winxp
```

Listing 6.11: Routing configuration (routing.conf)

```
[routing]
internet = ens33

[tor]
enabled = yes
dnsport = 5353
proxyport = 9040
```

Additionally, the Internet Explorer module had to be modified so that in the Windows 10 VM it would visit websites using the Microsoft Edge browser. To invoke Edge analysis mode, an argument, "edge", had to be supplied when submitting a new analysis to the Cuckoo host. The modified module can be seen in listing 6.12. This module utilizes the option that Microsoft Edge has, the ability to be started via shell-commands by calling *start microsoft-edge <target>*.

Listing 6.12: Modification to Internet Explorer Cuckoo module

```
def start(self, target):
        try:
                edge = self.options["edge"]
        except KeyError as e:
                edge = None
        if edge == "edge":
                """Microsoft Edge analysis package."""
                command = "start microsoft-edge:" + target
                return os.system(command)
        else:
                if "proxy" in self.options:
                self.setup_proxy(self.options["proxy"])

                # If it's a HTML file, force an extension, or otherwise Internet
        # Explorer will open it as a text file or something else non-html.
        if os.path.exists(target) and not target.endswith((\
                ".htm", ".html", ".mht", ".mhtml", ".url")):
                os.rename(target, target + ".html")
                target += ".html"
                log.info(\
                "Submitted file is missing extension, adding .html")

        iexplore = self.get_path("Internet Explorer")
        return self.execute(\
                iexplore, args=[target], maximize=False, mode="iexplore")
```

Execution

Cuckoo has multiple ways to add new analyses to the Cuckoo host, a web-based solution which you can host locally and add via HTTP requests or a python-based interface where you add items directly to the Cuckoo-database. We implemented a simple script using the latter method that would add 3 analyses for each domain, 1 for each of the VMs, an excerpt from the implementation can be seen in the listing 6.13 where the arguments that are supplied to the Cuckoo host is included. Note the additional *edge* argument for the Windows 10 analysis VM. We made an additional script that utilized the Cuckoo API to query the analysis status so that we restarted all analyses that had failed when our Cuckoo host had to be restarted since all analysis that were currently running when a restart/crash happened are marked as failed.

Listing 6.13: Excerpt from Cuckoo analysis insertion script

```
def scan_url(url):
        submitted_tasks.append(db.add_url(url,timeout=120,tags="winxp",\
                options="procmemdump=yes,route=tor"))
        submitted_tasks.append(db.add_url(url,timeout=120,tags="win10",\
                options="edge=edge,procmemdump=yes,route=tor"))
        submitted_tasks.append(db.add_url(url,timeout=120,tags="win7",\
                options="procmemdump=yes,route=tor"))
```

To check the status of our analyses we used curl to query the Cuckoo status as shown in the Cuckoo documentation for the API[32]. This returned a useful overview of the current status which showed how many tasks were in the queue, running and done.

6.3 Analysis phase

6.3.1 File and link analysis

The site data collection dataframe that was explained in section 6.2.1 collected 947 domains where 668 was unique and in the blacklist. There were some domains that had multiple sites in the dataframe because they had sites to resolve that the crawler visited, in listing 6.14 the 5 most resolved domains are shown.

[32] https://docs.cuckoosandbox.org/en/latest/usage/api/#cuckoo-status, retrieved 25.5.19

Listing 6.14: Top 5 resolved domains with their respective counts

```
www.calabriasportfishing.com 95
www.vishwaweighingsystem.com 24
hkitforce.com                17
webdesigning.name            17
podstrigis.com               16
```

We then gathered the file type information using the *file* command in Linux which was gathered by a script written in Python that would call a subprocess and collect the output for each, this output was then trimmed down since the filetype output can be very long as seen in listing 6.15. By trimming them down we were able to more easily see the file distribution and it can be seen in listing 6.16. By analyzing the *data*, *UTF-8* and *ASCII* files further we were able to determine that 9 of the data-files were binary files, but most likely corrupted files where the crawler must have crashed or timed out during creation or the files were originally garbled. The rest of the data-files were HTML-files with encoding issues. Out of the UTF-8 files, 2 were HTML-files the rest were logfiles created by the crawler. The same were the case with the ASCII files, none of them being HTML-files, all being logfiles.

Listing 6.15: Example filetype output from file command

```
HTML document, ISO-8859 text, with very long lines, with CRLF, LF line terminators
```

Listing 6.16: File types of all downloaded files by crawler

```
HTML document                              7064
UTF-8 Unicode text                          250
JPEG image data                             247
data                                        166
XML 1.0 document                             76
ASCII text                                   56
PDF document                                 24
PNG image data                               24
Zip archive data                              6
PHP script                                    4
Microsoft PowerPoint 2007+                    2
Composite Document File V2 Document           2
MPEG ADTS                                     1
Microsoft OOXML                               1
XML document                                  1
RAR archive data                              1
gzip compressed data                          1
-------------------------------------------
Total files                                7926
-------------------------------------------
```

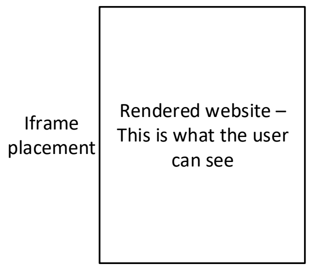


Figure 15: Iframe illustration, created in Visio Professional

Since we found no executables, scripts or similar interesting files in the files directly downloaded by our crawler, we checked the office files on VirusTotal, but no engines detected anything suspicious. We then used the ClamAV[33] antivirus engine to do a recursive scan on all folders within the crawler's download folder, this includes the files downloaded by the pywebcopy module that is creating its own folder inside the crawler's download folder as shown in section 5.3.2. ClamAV found 58 possible threats after scanning 26278 files where the classification distribution is show in figure 16. Two domains stood out since respectively 19 and 17 detections were found from them:

Domain A - 19 detections All detections were Html.Trojan.Iframe-87

Domain B - 17 detections All detections were Win.Malware.Iframe-6803839-0

These threats are typical delivery malware, by being embedded on websites their task is to infect users with malware, often trojans. Iframes are as mentioned previously in the thesis, easily embedded as images you cannot see with their size specified to 0 and their position rendered in the negatives, thus outside the browsing area. An example of this can be seen in figure 15 where the typical iframe placement is illustrated.

[33]`https://www.clamav.net/`, retrieved 26.5.19

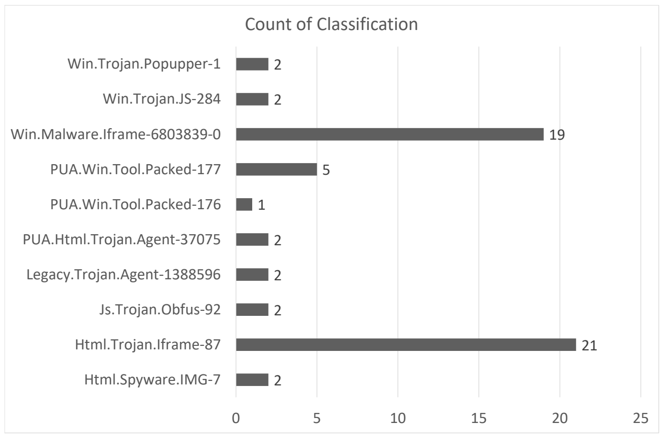


Figure 16: ClamAV classification distribution of all downloaded files from the crawler

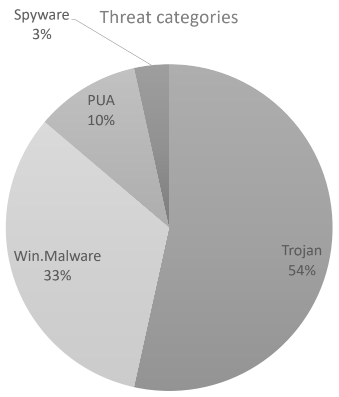


Figure 17: ClamAV threat categories

6.3.2 Text content analysis

Our content analysis started with cleaning our documents then we looked at words used in the body of all websites. This gives us indication of the complexity, e.g. if every site was just a template with "Error, the server did not respond" when an unsupported fingerprint is detected.

Listing 6.17: Words description on domains without JavaScript rendering enabled

```
Number of non-words (that are occurring more than 100 times): 51
Number of (any) words in body text that are occurring more than 100 times: 199
Total number of (any) words in body text: 63806
Total number of unique words: 131 Number of sites with content: 918
```

Listing 6.18: Top 10 most common words and their count on domains without JavaScript rendering enabled

```
the 2145
var 2112
font 1938
and 1634
to 1548
a 1222
in 1111
if 910
of 871
px 745
```

Listing 6.19: Word description on domains with JavaScript rendering enabled

```
Number of non-words (that are occurring more than 100 times): 69
Number of (any) words in body text that are occurring more than 100 times: 288
Total number of (any) words in body text: 80019
Total number of unique words: 188 Number of sites with content: 898
```

Listing 6.20: Top 10 most common words and their count on domains with JavaScript rendering enabled

```
in 3983
on 3316
be 2920
the 2884
not 2852
line 2685
should 2620
method 2595
strict 2591
standards 2591
```

We further removed the following: duplicate bodies in our dataframe, words without (Latin) characters a-z, A-Z and words with length less than 2. We did not see a clear difference in the basic word analysis between the pre-duplication removal and the post-duplication removal. The word description post-duplication removal can be found in appendix D.

As explained in section 5.4.4, LDA is the topic modeling algorithm that was used to analyze the body content. The library utilized were gensim and their LDA model implementation [34]. The model was created, ran and output modeled topics with the following settings:

[34]https://radimrehurek.com/gensim/models/ldamodel.html, retrieved 25.5.19

Listing 6.21: LDA model creation in Python

```
cleaned_bodies = [clean(body) for body in topic_text[body_type]]
cleaned_bodies = [body.split() for body in cleaned_bodies]

dictionary = corpora.Dictionary(cleaned_bodies)
word_matrix = [dictionary.doc2bow(body) for body in cleaned_bodies]

lda_model = gensim.models.ldamodel.LdaModel

ldamodel = lda_model(word_matrix, num_topics = 10,\
         id2word = dictionary, passes=1000)

print(ldamodel.print_topics(num_topics=10, num_words=3))
```

Made a dictionary from the cleaned bodies and then made a word matrix of them. The model was then created with the word matrix we just made and specified to find 10 topics. Further it shall do 1000 passes when training. We ran our model on both body content found on non-JavaScript and Java-Script rendered websites.

Listing 6.22: 10 topics found by our topic model together with their weights without JavaScript enabled, words occurring 100 or more times

```
[(0, '0.062*"solid" + 0.062*"format" + 0.062*"issues"'),
(1, '0.062*"color" + 0.062*"gt" + 0.062*"normal"'),
(2, '0.059*"ul" + 0.059*"right" + 0.059*"unicode"'),
(3, '0.062*"switch" + 0.062*"domain" + 0.062*"span"'),
(4, '0.059*"posted" + 0.059*"woocommerce" + 0.059*"contact"'),
(5, '0.066*"home" + 0.066*"ethernet" + 0.066*"services"'),
(6, '0.066*"true" + 0.066*"italic" + 0.066*"following"'),
(7, '0.053*"cdata" + 0.053*"view" + 0.053*"latin"'),
(8, '0.095*"pocket" + 0.095*"de" + 0.095*"template"'),
(9, '0.066*"txtnew" + 0.066*"li" + 0.066*"weekly"')]
```

Listing 6.23: 10 topics found by our topic model together with their weights with JavaScript enabled, words occurring 100 or more times

```
[(0, '0.043*"best" + 0.043*"format" + 0.043*"px"'),
(1, '0.042*"tm" + 0.042*"page" + 0.042*"none"'),
(2, '0.043*"woocommerce" + 0.043*"method" + 0.043*"de"'),
(3, '0.035*"href" + 0.035*"information" + 0.035*"solid"'),
(4, '0.042*"issues" + 0.042*"service" + 0.042*"ethernet"'),
(5, '0.043*"contact" + 0.043*"list" + 0.043*"quick"'),
(6, '0.042*"captured" + 0.042*"wake" + 0.042*"span"'),
(7, '0.043*"please" + 0.043*"return" + 0.043*"trending"'),
(8, '0.049*"htmldiv" + 0.049*"called" + 0.049*"prop"'),
(9, '0.032*"origami" + 0.032*"false" + 0.032*"following"')]
```

Listing 6.24: 10 topics found by our topic model together with their weights without JavaScript enabled, duplicates removed, filtered words

```
[(0, '0.237*"vc" + 0.049*"px" + 0.020*"av"'),
(1, '0.046*"de" + 0.020*"la" + 0.011*"en"'),
(2, '0.010*"de" + 0.008*"service" + 0.007*"server"'),
(3, '0.025*"gb" + 0.024*"px" + 0.023*"tm"'),
(4, '0.022*"var" + 0.013*"px" + 0.012*"dropdown"'),
(5, '0.085*"font" + 0.052*"format" + 0.045*"url"'),
(6, '0.035*"px" + 0.027*"background" + 0.027*"var"'),
(7, '0.039*"class" + 0.026*"woocommerce" + 0.020*"div"'),
(8, '0.049*"lt" + 0.025*"class" + 0.019*"view"'),
(9, '0.035*"var" + 0.023*"http" + 0.018*"document"')]
```

Listing 6.25: 10 topics found by our topic model together with their weights with JavaScript enabled, duplicates removed, filtered words

```
[(0, '0.133*"font" + 0.074*"format" + 0.067*"url"'),
(1, '0.171*"vc" + 0.037*"px" + 0.021*"woocommerce"'),
(2, '0.024*"var" + 0.019*"av" + 0.013*"de"'),
(3, '0.036*"eapps" + 0.010*"li" + 0.010*"header"'),
(4, '0.013*"service" + 0.007*"de" + 0.007*"page"'),
(5, '0.090*"home" + 0.089*"line" + 0.087*"strict"'),
(6, '0.087*"px" + 0.028*"background" + 0.028*"important"'),
(7, '0.037*"var" + 0.035*"class" + 0.023*"div"'),
(8, '0.022*"domain" + 0.019*"var" + 0.014*"wishlist"'),
(9, '0.020*"issue" + 0.011*"switch" + 0.009*"series"')]
```

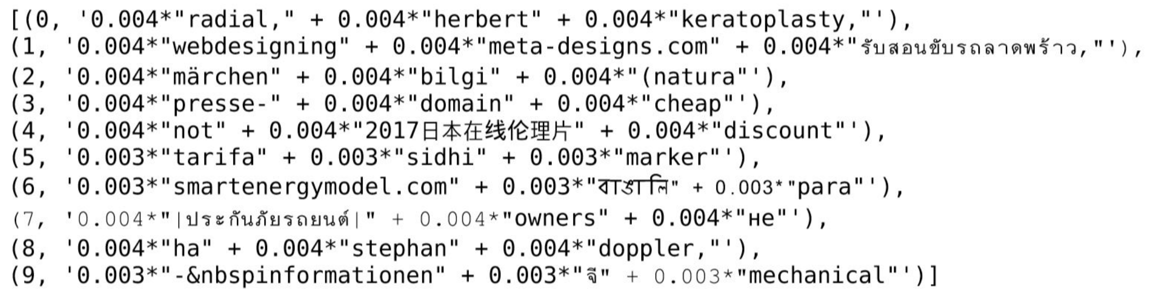
We also ran topic modeling on the titles of the websites that we crawled. There were many non-Latin characters detected as can be seen in the figure 18. Something which stands out in all the modeled topics and most occurring words are the amount of source code, this is unusual for e.g. benign websites since users do not want to see garbled looking websites, they want nice designs.

6.3.3 GeoIP analysis

From the GeoIP we made a list of the top 10 most represented countries seen in listing 6.26

Listing 6.26: Top 10 most represented countries

```
United States                341
Ireland                       96
Germany                       86
Netherlands                   37
China                         31
British Virgin Islands        28
Japan                         26
```



```
[(0, '0.004*"radial," + 0.004*"herbert" + 0.004*"keratoplasty,"'),
(1, '0.004*"webdesigning" + 0.004*"meta-designs.com" + 0.004*"รับสอนขับรถลาดพร้าว,"'),
(2, '0.004*"märchen" + 0.004*"bilgi" + 0.004*"(natura"'),
(3, '0.004*"presse-" + 0.004*"domain" + 0.004*"cheap"'),
(4, '0.004*"not" + 0.004*"2017日本在线伦理片" + 0.004*"discount"'),
(5, '0.003*"tarifa" + 0.003*"sidhi" + 0.003*"marker"'),
(6, '0.003*"smartenergymodel.com" + 0.003*"বাঙালি" + 0.003*"para"'),
(7, '0.004*"|ประกันภัยรถยนต์|" + 0.004*"owners" + 0.004*"не"'),
(8, '0.004*"ha" + 0.004*"stephan" + 0.004*"doppler,"'),
(9, '0.003*"-&nbspinformationen" + 0.003*"จี" + 0.003*"mechanical"')]
```

Figure 18: Topic modeling output from modeling website titles

```
Russia                           26
Bulgaria                         25
France                           21
```

6.3.4 WHOIS analysis

WHOIS data is as described in the method section 5 not reliable not much weight will be given, but we did do some frequency analysis on the results from that too, seen in listing 6.27. Number 4 in the list has been translated using Google Translate, Chinese to English[35].

Listing 6.27: Top 5 names from WHOIS

```
Private Person                       23
Redacted for Privacy Purposes         7
REDACTED FOR PRIVACY                  3
Chen Jianjun                          2
SAKURA Internet Domain Registration   2
```

6.3.5 URL Abuse analysis

From URL Abuse the most interesting data was the BGP Ranking[36]. This can give information over the ranking of the DNS compared with each other, in listing 6.28 is the statistical calculations of the domains checked with URL Abuse that produced BGP Ranking information. When comparing the rankings to the ones seen on the BGP Rankings' page we saw that the ranks we found were not particularly high. We did also do frequency distribution calculation on the DNS servers in the dataframe with BGP Ranking, this produced some interesting results seen in listing 6.29.

Listing 6.28: BGP Ranking description generated from dataframe with Pandas

```
count     930.000000
mean        0.001333
```

[35] https://translate.google.com/#view=home&op=translate&sl=auto&tl=en&text=%E9%99%88%E5%BB%BA%E5%86%9B, retrieved 26.5.19
[36] https://www.circl.lu/projects/bgpranking/, retrieved 26.5.19

```
std          0.004602
min          0.000000
25%          0.000023
50%          0.000166
75%          0.000742
max          0.042969
```

Listing 6.29: Top 10 DNS Servers in BGP Ranking dataframe

```
217.78.1.23                96
82.118.242.92              25
204.11.56.48               15
23.20.239.12               13
2a01:238:20a:202:1072::     5
208.91.197.44               4
23.253.126.58               4
199.59.242.151              4
104.239.157.210             4
184.168.131.241             4
```

6.3.6 VirusTotal analysis

VirusTotal had records for all the domains, except 1 in our selection from the blacklist. This is not surprising given that they are blacklisted for malicious activity. In listing 6.30 is the classifications that these domains have in the VirusTotal database from the different vendors that have data on these domains. Listing 6.31 shows the top 10 antivirus engines (vendors) that have data on the domains in question. The number of positive engine scan results per domain are shown in table 15.

Listing 6.30: Classification of blacklisted domains in VirusTotal with their respective count

```
malicious site                          4890
malware site                            3477
phishing site                           1136
```

Listing 6.31: Top 10 antivirus engines

```
Malware Domain  Blocklist  1887
AutoShun                   1340
G-Data                     1308
Fortinet                   1048
BitDefender                 836
Sophos                      681
Avira                       487
Google  Safebrowsing        438
Kaspersky                   328
Dr.Web                      260
```

Number of positives	Number of results
4	409
5	356
3	343
6	236
7	208

Table 15: The top 5 scans ranked on their number of positive AV engine results

6.3.7 Cuckoo analysis

For the Cuckoo analysis part, we removed the domains that were dead when we ran domain data collection and used the WHOIS lookups to filter out dead domains. Therefore, we had a list of 1781 of the original selected malicious domains that were explained in section 5.2. After parsing all the Cuckoo generated reported we filtered out all analyses that did not resolve. In listings 6.32, 6.33 and 6.34 we have listed the score distributions calculated with Pandas using the description command.

Listing 6.32: Cuckoo analyses scores for the Windows XP VM

```
count  641.000000
mean  3.542590
std  0.742933
min  0.000000
25%  3.400000
50%  3.400000
75%  3.800000
max  6.400000
```

Listing 6.33: Cuckoo analyses scores for the Windows 7 VM

```
count  638.000000
mean  4.453292
std  0.912041
min  0.400000
25%  3.800000
50%  4.400000
75%  4.800000
max  8.000000
```

Listing 6.34: Cuckoo analyses scores for the Windows 10 VM

```
count  647.000000
mean  2.012056
std  0.835240
min  0.400000
25%  1.400000
50%  2.200000
```

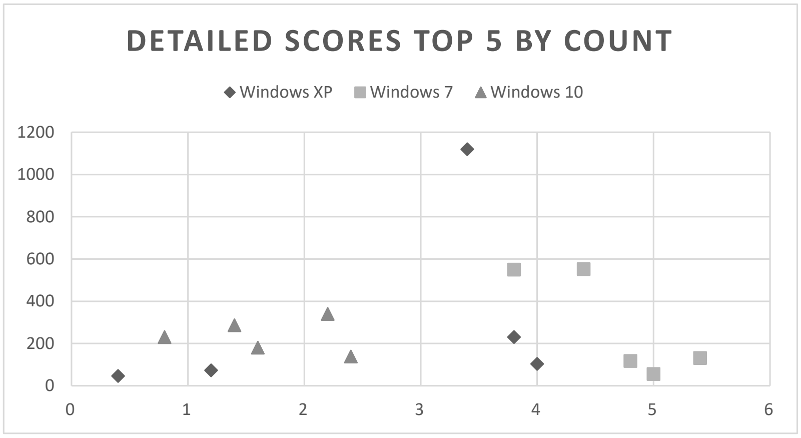


Figure 19: Detailed Cuckoo scores for the 5 most common scores per analysis machine

```
75% 2.400000
max 4.200000
```

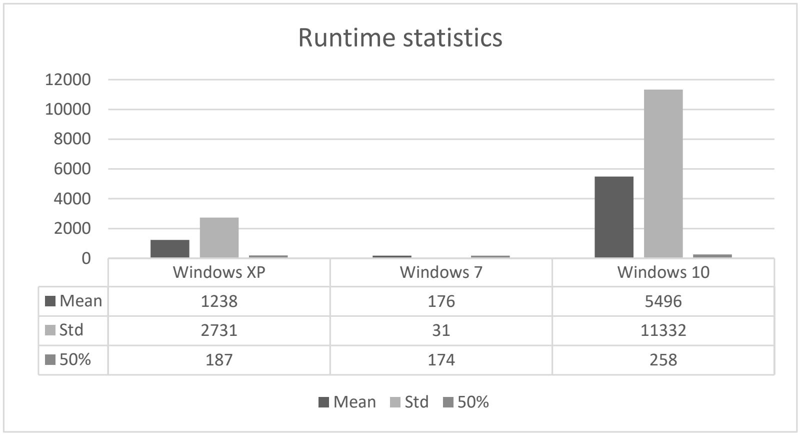
Execution issues

Some issues arose during the Cuckoo analysis, we mentioned some in 6.1.3, but we did experience two errors with our VM images which caused some notable differences in the runtime duration of our scans. The VMs which had issues were the Windows XP VM and the Windows 10 VM, the Windows XP got an issue with licensing, so we had to fix the license then take a new snapshot and nothing else was changed. The Windows 10 had an obtrusive system process that was hanging up the Cuckoo host, this was resolved, a new snapshot taken, and the analysis could continue. In figure 20, we see that the Windows 7 VM was the most efficient VM throughout the whole analysis. When not suffering from issues, the Windows XP VM did analyze in a pretty efficient manner compared with the Windows 7 VM. The Windows 10 VM on the other hand used quite a lot of time even when not suffering from issues.

6.3.8 Manual analysis

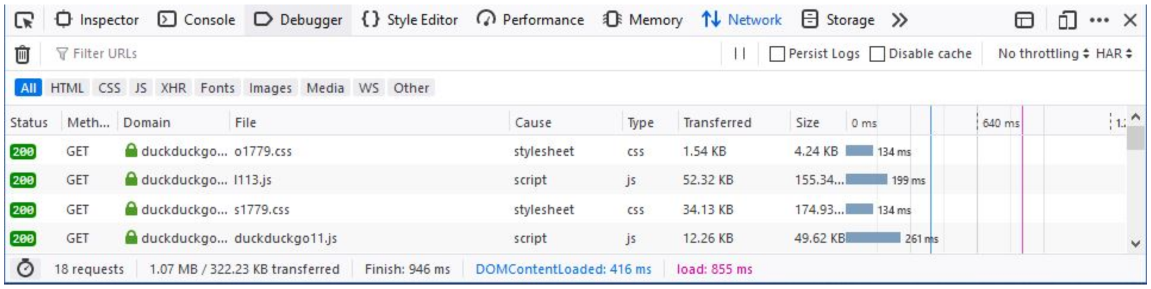
To select which domains to analyze in our manual analysis we decided to use measures that can easily be selected and confirmed by others doing similar experiments. The selected measures were; Cuckoo scores above 4.5 and VirusTotal reports with more than 4 positive results on the particular domain. This produced a list of 92 unique domains.

Further, to analyze these the *Manual Analysis* VM from section 6.1.2 were used. When browsing these domains, the web console in Firefox was open, in it the network and inspector tabs were utilized. The network tab allowed us to see the connections as they were made, and the inspector



	Windows XP	Windows 7	Windows 10
■ Mean	1238	176	5496
■ Std	2731	31	11332
■ 50%	187	174	258

Figure 20: The runtime of the Cuckoo analysis VMs



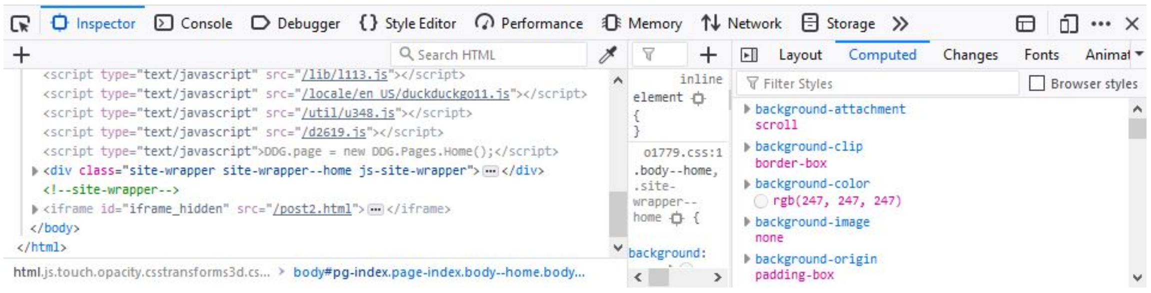
Status	Meth...	Domain	File	Cause	Type	Transferred	Size	
200	GET	duckduckgo...	o1779.css	stylesheet	css	1.54 KB	4.24 KB	134 ms
200	GET	duckduckgo...	l113.js	script	js	52.32 KB	155.34...	199 ms
200	GET	duckduckgo...	s1779.css	stylesheet	css	34.13 KB	174.93...	134 ms
200	GET	duckduckgo...	duckduckgo11.js	script	js	12.26 KB	49.62 KB	261 ms

18 requests | 1.07 MB / 322.23 KB transferred | Finish: 946 ms | DOMContentLoaded: 416 ms | load: 855 ms

Figure 21: The network tab in the Firefox web console

tab allowed us to see the source code of the website directly. Examples of these tools can be seen in figure 21 and 22. This allows us to see and follow network requests and inspect to see if there are iframes or other hidden parts on the websites. To save our analysis we had an Excel spreadsheet open with the following columns: domains, descriptions, type, social engineering techniques, private information requests, reloaded, warning from Google Safe Browsing, links/redirect landings. Of these columns the most relevant became the domains, description, type and Google Safe Browsing. The "type" column is the determined type of the domain by us when analyzing. The distribution of these can be seen in figure 23. Additionally, in figure 24, can we see the Google Safe Browsing warnings that showed up. The "Yes" option in the chart is warnings for malicious activity, but not specified further.

To further analyze the domains, the 30 first were checked manually in VirusTotal where there is



```
    <script type="text/javascript" src="/lib/l113.js"></script>
    <script type="text/javascript" src="/locale/en_US/duckduckgo11.js"></script>
    <script type="text/javascript" src="/util/u348.js"></script>
    <script type="text/javascript" src="/d2619.js"></script>
    <script type="text/javascript">DDG.page = new DDG.Pages.Home();</script>
  ▸ <div class="site-wrapper site-wrapper--home js-site-wrapper">…</div>
    <!--site-wrapper-->
  ▸ <iframe id="iframe_hidden" src="/post2.html">…</iframe>
  </body>
</html>
html.js.touch.opacity.csstransforms3d.cs… > body#pg-index.page-index.body--home.body…
```

```
       inline
element {
}
 o1779.css:1
.body--home,
.site-
wrapper--
home {
background:
```

```
▸ background-attachment
  scroll
▸ background-clip
  border-box
▸ background-color
  rgb(247, 247, 247)
▸ background-image
  none
▸ background-origin
  padding-box
```

Figure 22: The inspector tab in the Firefox web console

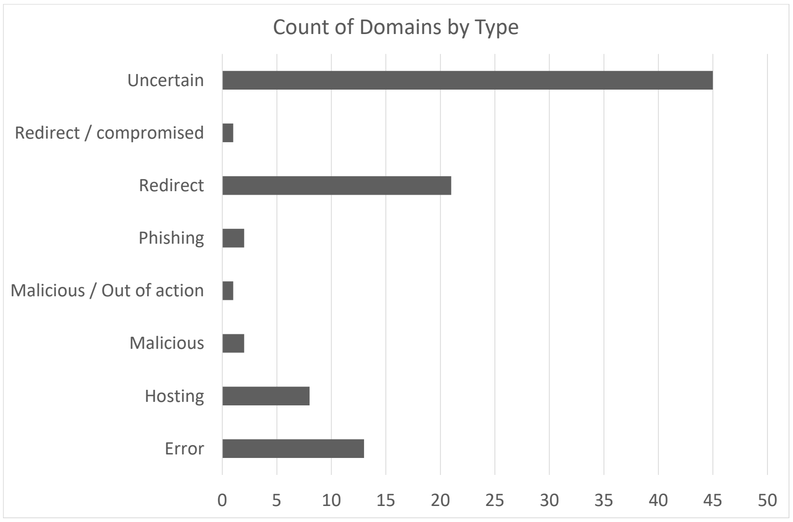


Figure 23: The types of domains visited during the analysis

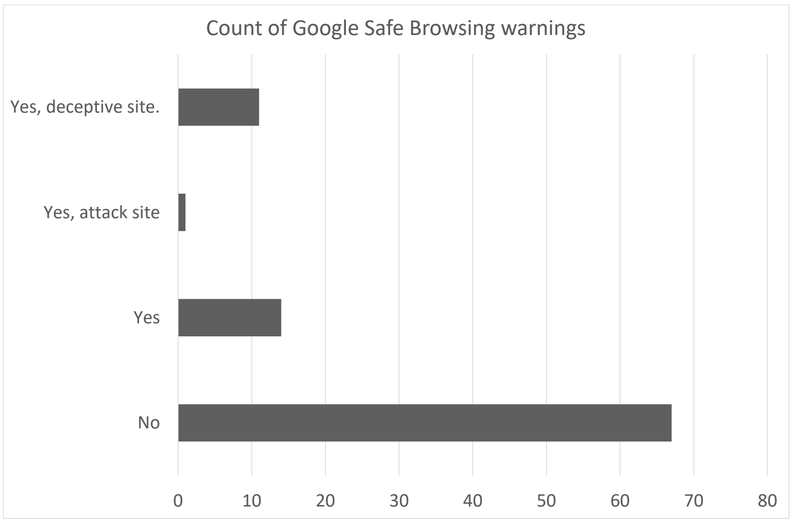
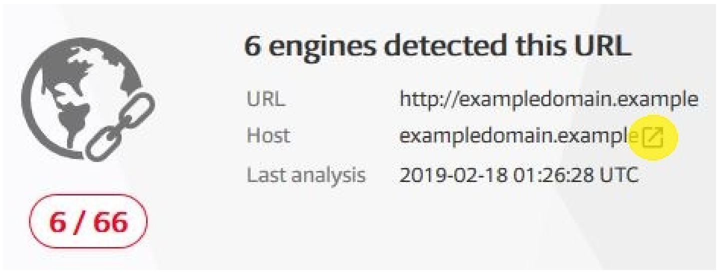


Figure 24: Google Safe Browsing warnings shown during analysis



6 engines detected this URL

URL http://exampledomain.example
Host exampledomain.example
Last analysis 2019-02-18 01:26:28 UTC

6 / 66

Figure 25: A snippet of the VirusTotal website, advanced domain information link is highlighted in yellow

an option to get a more detailed view of domains when using their website. In figure 25, the advanced domain information link is shown. This gives us the following additional information about domains[37]; categories it has been classified as, passive DNS replication to get the IP the address resolves to, WHOIS lookup, observed subdomains, URLs associated with the site (e.g. download links), downloaded files and lastly the communicating files which communicates with the domain on execution or opening. For the 30 analyzed domains we saw that 27 of the domains had files in subfolders. This means that when visiting the domain, you are not seeing the subfolder if it has not been linked to. One of the most interesting things to get from doing this was the threat categories present on the analyzed domains, these can be seen in figure 27.

[37]In this example the following domain was used:https://www.virustotal.com/#/domain/shzwnsarin.com, retrieved 26.5.19

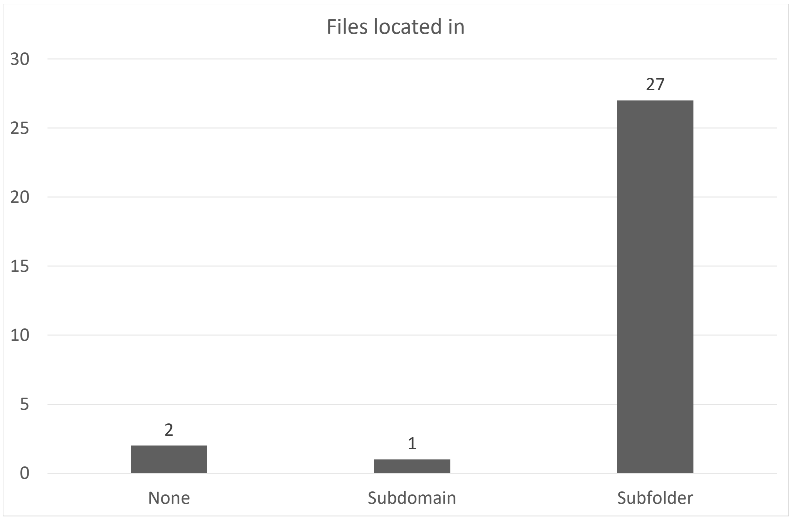


Figure 26: Location of files shown when analyzing domains on VirusTotal

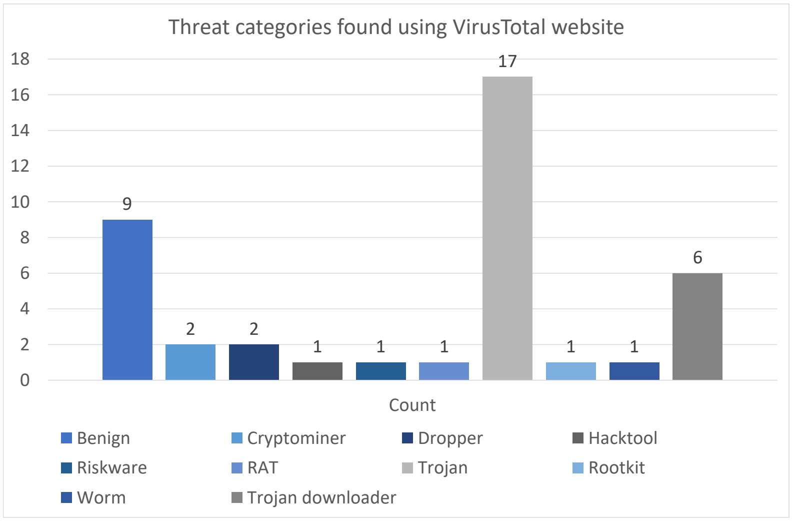


Figure 27: Threat categories found by using VirusTotal's advanced domain information

7 Discussion

The collection phase where the crawler was collecting files and content from the malicious domains did not result in the *classical* malicious executables one could expect from domains that are labeled as spreading malware and being malicious. This was clearly shown in section 6.3 and listing 6.16. On the other hand, it lines up with what the landscape has evolved to over the years as shown in the malware evolution section 2.3 and the key takeaways in section 2.3.2. A shift that already started in 2006 towards the internet and the possibilities for exploitation and the increased attack surface. The notion that you should trust a website was still a thing in 2008 since Microsoft recommended to "only browse sites you trust" [29], this notion is abolished today for many reasons, but on obvious one being the advertising networks that are on the majority of sites you should *trust*. Additionally, the bigger you get, the juicer you become as a target as seen in attacks on MySpace in 2005 and 2006 already [24] and Facebook in 2010 [47].

The kind of content that is on websites that could entice users to visit them is a question that we can try to answer by looking at the text content analysis in section 6.3.2. Looking at the top 10 most common words for websites rendered without JavaScript, listing 6.18, one could almost think we are looking at source code given "px", "var", "font", but it seems when looking at the bodies we collected and during our manual analysis that some sites were broken in some way and showed source code in their HTML bodies. The most common words with JavaScript enabled are quite different and resembles more natural language. So, from this we can assume users will see more natural language given that most users are browsing with JavaScript rendering enabled (since many websites will say that they cannot deliver their content without JavaScript and thus one could assume that a regular user would not be bothered to change on and off JavaScript). To get a better understanding of this content we can look at the topic modeling results which possibly can shine more light on the actual contents. For the first listing, 6.22, we see many topics that can be connected to code and formatting issues, specifically topics 0, 1, 2, 3, 6, 7, 8 and 9. Topics 4 and 5 on the other hand give us interesting information. Woocommerce[1] is one of the most popular commerce plugins for Wordpress and the associated words, posted and contact are regularly seen on webshops. This tells us that many of the websites in our dataset is most likely compromised wordpress sites which has been a target of malware campaigns for a while as seen in our malware evolution section 2.3. Topic 5 have combined the words home, ethernet and services which might point to websites that deliver services, such as small businesses that might use Wordpress[2] since it is an easy content management solution to use and setup. Woocommerce is seen in all modeled topics, but source code related words are also present in all, so it seems that many of these websites are

[1]https://woocommerce.com/, retrieved 26.5.19
[2]https://wordpress.org/, retrieved 26.5.19

having problems with their source code. It might also be that injected code is being misrepresented for our crawler so that we get it in raw code form instead of it executing.

Some detections were made by ClamAV, most of these being iframe-based malware as seen in figure 16 with the categories being aligned with the categories that the overall landscape have produced the last 12 years as seen in the malware evolution section 2.3. When we then manually analyzed these domains, we saw that most of the malware one could download from them was unavailable for direct download since they resided in subfolders as seen in figure 26 where 27 of 30 domains are utilizing subfolders for delivery of malware. There could be multiple reasons for this, they could be malware hosts, or they could be showing different websites, benign looking ones, when browsing them manually or with a crawler. The threats that were found on the detailed VirusTotal domain information again shows that the blacklisted domains are delivering malware that aligns with the general cyber threat landscape with trojans being the most popular malware type to deliver. Additionally, PUAs, hacktools and such are also being detected by the lots which are also seen by us on the blacklisted domains.

When looking at the Cuckoo results we get interesting results, Windows 10 is by far the most secure operating system with the default configuration. In listing 6.34, we see that the mean score is 2 for the OS and in figure 6.3.7 most of the Windows 10's scores are at the lower end. Windows XP is by the mean the second most secure system as seen in listing 6.32, but in figure 6.3.7 it is both represented at the lowest scoring point with 0 score and at the highest by volume at ~ 3.4 score. Windows 7 is well represented at the highest echelon, scoring the highest of all 3-operating system version. The focus towards Windows 7 from malware authors is showing up in Microsoft's own intelligence reports, e.g. in Volume 18, figure 33 [38] where Windows 7 have a higher infection rate than Windows Vista, Windows 8 and Windows 8.1. Windows XP is not shown in their reports from 2014 and onward since it had passed the end of life support in 2014 as mentioned in the introduction of this thesis. Even though it is not supported anymore it proved in our small sample size to be more secure on average than Windows 7 which is a much newer operating system and used by many more. It could also be that some security features that are mentioned in the Microsoft Windows operating system and security measures evolution has not been enabled since by default they are opt-in [11]. Further, without many of these enabled and a bigger focus on Windows 7 by malware developers, Windows XP could skirt by, even though that is unlikely since exploits for Windows XP are probably included in the exploit kits by default as seen in our exploit kit section 2.2. Our Windows XP VM was installed with just the default applications that comes with Windows and updated to IE 8.0 so it might be that the attack surface is limited as recommended by [19] so that exploits would not exploit it as easily.

An interesting statistic from Volume 14 [12] from Microsoft is the comparison in infection rates for protected versus unprotected systems. For Windows XP SP3 in December of 2012 a system running with protection enabled was 4 times less likely to be infected then a system without protection. For Windows 7 the numbers were even higher where the Windows 7 retail version with protection had an infection rate of 4.8% compared to the unprotected version which had an infection rate of 34%. This goes to show that it does not help only running a newer system, you also must run

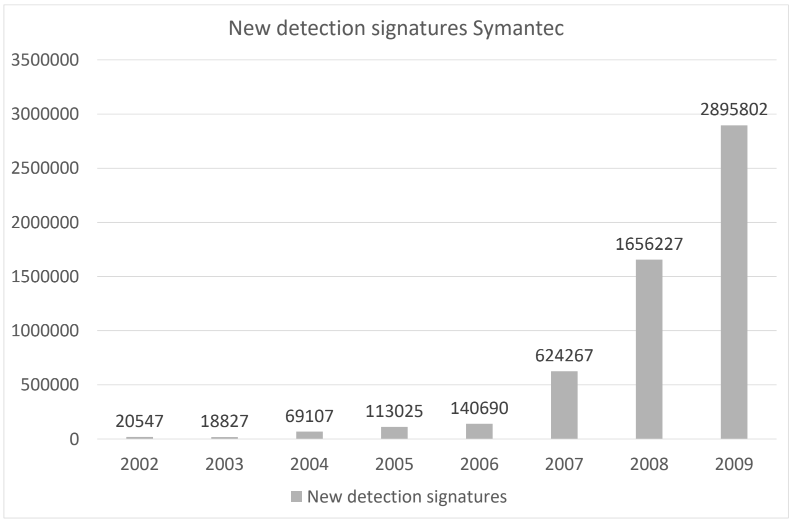


Figure 28: New detection signatures created each year, 2002 to 2009 — number from [3, 4]

real-time protection.

Further, the threats we have seen downloaded from our crawler and executed on our VMs correlates well with other information that Symantec has seen over the years. In their 2018 internet security threat report [8] they say that 1 in 10 URLs are malicious. This is an interesting fact when set in perspective with their internet security threat report volume 18 [88] where only 1 in 532 of websites were found to be infected with malware. What this means is that malware since 2006, as seen in our noteworthy takeaways section 2.3.2, have turned towards web-based malware and attacks since that is the avenue that is their opening. The reasons behind this are many, but mainly being advanced filtering of mail so that malware being spread via mail is not as effective as it was in the early half of the 2000 decade, the Windows operating system has been hardened over many iterations and the newest version has real time antivirus, security features such as UAC, ASLR, DEP, kernel unlinking and so on. Even with increased defenses, the attackers have not been resting either since these developmental changes has created an arms race on both sides. Symantec has statistics for the malware volume since 2002 to 2018 which is shown in figure 28 and 29. They had to be split up since the explosion that happened from 2009 to 2010 skewed the numbers so much, multiple reasons for this increase; polymorphism, obfuscation, encryption and the use of simple droppers as seen by Symantec in 2007 already from our noteworthy takeaways section 2.3.2 are some of the reasons.

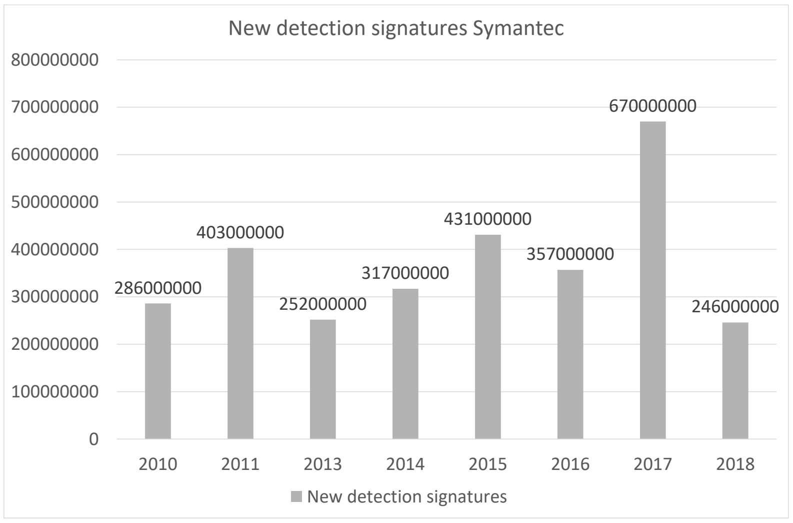


Figure 29: New detection signatures created each year, 2010 to 2018 — numbers from [5, 6, 7, 8]

7.1 Theoretical implications

The thesis is aiming to strengthen the understanding of what is on blacklisted malicious domains. Previously there have been research done on detecting malicious domains and reports on the cyber threat landscape on endpoints that the authors of said reports has had access to statistics from. We wanted to combine the two different approaches presented before and go directly to the source and do a comprehensive study on what we could find. Our main research question, **what kind of cyber threats and content can be found on domains that are blacklisted and labeled as malicious by DNS-BH?** will be answered by answering our detailed sub-questions in the following description:

RQ 1.1.a We found no malicious executables that we might have hypothesized we would find when we created the research questions. Since we were studying websites labeled as spreading malware, we were certain that we would be presented with websites that had direct links to malicious binaries. This was not the case as shown in our file and link analysis, section 6.3.1. Instead, we found malicious iframes and HTML based trojans. Our advanced analysis with VirusTotal revealed that these domains were used as distribution points for various kinds of malware stored in subfolders as seen in figure 26. The malware that were distributed were mainly trojans. This is in line with the threat landscape that is today.

RQ 1.1.b We have found multiple connections between malicious domains and the content shown. Exploit kits with fingerprinting as presented in section 2.2 have been encountered many times where we have been redirected to benign websites, but before this redirection there has been multiple network requests that has happened which has exchanged information about our different systems. We have also seen many instances of what can be assumed as compromised websites since our topic modeling are all presenting us with Woocommerce and our manual analysis also showed multiple Wordpress based websites that were seemingly benign, made by ordinary people, but nonetheless blacklisted. One connection with the ones that were manually checked was that they were not updated lately so it could be assumed that they might be abandoned by their creators and thus they will not detect that their website is blacklisted. You could also argue that people with small websites using Wordpress will not have the technical capability or interest to go around searching for their own domain to check if they are blacklisted or not, would they even know what it is.

RQ 1.2.a It can tell us that they are generalizable malicious domains given the distributions seen in figure 17 and 27 that are on the intrusion and infection stages of the attack process as outline in F-Secure's 2015 threat report [13]. The intrusion stage is the exploit kit's job and the infection stage are the typical trojan installation. This is one of the reasons trojans are so persistent and present in all years of threat categories. They are exploiting users in one way or another to gain access to their systems.

RQ 1.2.b There are some indications that this is a reality, from our URL Abuse analysis seen in listing 6.29 we have multiple domains that have the same DNS server. It is probable that it is a weakness from the registrar as seen by Talos with GoDaddy and a large usage of accounts from GoDaddy that were used to create subdomains [26]. Additionally, during the manual analysis it was seen that 4 domains were connected with the same server where network requests were being made with different parameters just as explained in section 2.2 with how exploit kits evade detection.

RQ 1.3.a and planned contribution With the domain and website infrastructure we have collected it is limited how we can detect compromised domains. Detecting a domain that has been taken is hard to do if there are no outward change, but we could build a machine learning model that could do unsupervised learning if we had a larger dataset that we could collect data from. We have already detected some anomalies in the topic modeling where we are seeing quite a lot of source code as content on these websites. A normal benign, updated website would not usually leak source code in the body since it would be following HTML compliance so that users would not be bothered by seeing incomprehensive content. Further, we could build up a database of benign infrastructure information that can be used as a baseline, but this would require a dataset of benign websites that were using external resources.

RQ 1.4.a The techniques were most clearly seen when doing the manual analysis. During the analysis we came across 9 of 92 domains with social engineering. All of them except one were domains that we ended up on after being redirected there, some of them even changed upon reload while others stayed the same. The typical get-rich-fast scheme were presented in the form of an ad for Bitcoin investment, one with the warning that you must enable some features in the browser by following the given link and the rest recommending you change your DNS server to a *much* faster one. If a user gives information away to the get-rich fast scheme you give away name, mail and phone number and if you enable some feature by following the link you can be led to an arbitrary site. Lastly, if you change your DNS server to their you will be giving them insight into all your web browsing.

RQ 1.5.a Throughout the creation of the malware and operating system evolution sections we have been presented with nice statistics from both Symantec, Microsoft and F-Secure about their filtering and blocking capabilities. They are all utilizing heuristics, signature-based lookups and machine learning to automate the detection and categorization of malware and malicious websites. Mozilla has had Google Safe Browsing since 2006[3] and Microsoft has had SmartScreen, Bing integrated filtering on indexing and lastly their Advanced Threat Protection for Office 365 [39]. All these services are making us able to detect malware almost instantaneously by using heuristics and machine learning when a new version or type is detected. Therefore, we would absolutely say it is reasonable to rely on automated systems to detect and categorize malicious websites for end-users.

7.2 Practical implications

During the thesis we have been exposed to many end-user recommendations and best practices, some of them very good while others except way too much of a user. Based on our study we will therefore list a few of the most effective recommendations for protecting your machine for use while connected to the internet:

- Updates are key. The most important thing a user can do is updating their systems and their applications [3, 4, 29, 11, 37, 43, 52]. This will ensure that vulnerabilities that are exploited will not work on your system, it is the most efficient precaution against malware that a user can do for free with little to no effort.
- Use antivirus software with real-time protection from a trusted vendor [3, 4, 29, 30, 37, 43, 52]. If you are using a Windows operating system you could use their free solution for all Windows versions since Windows XP to Windows 10.
- Take backups of your important files so that nothing is lost to ransomware or other unfortunate events [52, 43].
- Do not install programs and plugins that you will not use [52, 29]. This will just increase your attack surface with no benefit, see section 2.2 for more on how this is utilized by attackers.
- The old principle of the least amount of privilege by having a user that is a standard user without administrator rights that you use for regular computer usage. Then you can use the administrator account when you need administrative rights [30, 43]. A *light* version of this is keeping UAC enabled so that everything cannot run as administrator on Windows operating systems. This ensures that malware does not get to run with administrator privilege easily

[3] `https://wiki.mozilla.org/Security/Safe_Browsing`, retrieved 27.5.19

since you must either actively log into your administrator account or press yes in the UAC prompt.

8 Summary and future work

In this thesis we have researched the cyber threat landscape on blacklisted malicious domains. We have shown that the landscape on blacklisted domains are not different from what is considered the general cyber threat landscape by the industry. Even with a limited dataset with many non-resolving domains we have gathered enough data to see that the focus of malicious domains is on exploit kits and not on directly spreading malware via direct links to malicious files. Further, we have shown the difference in resilience made by the different operating systems utilized by the user and how this affect the chances of survival on the online battleground that malicious websites create. We have shone light on the malicious threat landscape on the internet and seen that it is a landscape that has been there for many years and has been evolving alongside the operating systems, the browsers and applications that users are using every day. The threats we have seen are mainly exploit kits used by malicious domains and the most commonly delivered malware is trojans. This means that the most notorious threats are the ones that are let in by users, often unknowingly since it happens in the background. By demystifying the landscape we have further shown that there is not necessarily a need for users to use any special software other than a solid real-time protection software that is receiving updates at a regular pace together with sample analysis enabled, an operating system that is updated and have security features enabled and an updated browser that has multithreading, sandboxing and a modern API that is limiting the capabilities that an extension can have.

Further research

For further research we have identified improvements that could be done with the dataset we used or by creating a new dataset.

Crawler improvement

The crawler could be improved so that the limitations which was explained in section 6.2.1 was mitigated. Just adjusting the depth and user-agents utilized would most likely improve the results remarkably. User-agents is the most vital one given our extensive description in section 2.2. Another interesting approach is using optical character recognition (OCR) which have been utilized before e.g. in Grier et al. [57] where they created histograms which showed exploit kits which opened a program or if the VM had a blank screen. It could be extended by combining OCR with topic modeling so that websites could be classified based on their generated topics and how the websites "look" when doing OCR on it.

Dataset

Our dataset is very small after filtering out the non-malicious domains from the original DNS-BH blacklist. An improvement to get a more precise picture could be to build a larger blacklist dataset by basing it on multiple publicly available datasets.

Cuckoo improvement

To get a broader picture, more VMs could have been used such as having one or two Ubuntu VMs with different versions of Ubuntu Linux installed. This will help with the understanding of Linux malware, although the user base is quite low compared to Windows. Our existing machines could also have been made more *real* by expanding their attack surface by installing typical applications seen on home-computers. We could also have extended our variety of Microsoft Windows operating systems by having e.g. Windows 8 and Windows 8.1 VMs.

Trojans and social engineering

A threat category that appeared each year in our malware evolution is the trojan category. One could make a case that trojans are notoriously present, but a key thing lies in the definition of the trojan. It is named after the trojan horse and as explained in the malware taxonomy section 2.1. In the case of malware, it hides its intentions, a well-known example being the "codecs" needed for pirated media consumption as seen by Bosco and Shalaginov [76]. Therefore, it would be interesting to do a focus study in the future on how trojans have such an integral part in the malware ecosystem that they are utilized by threat actors everywhere.

Bibliography

[1] NetMarketShare. *Browser market share*, 2019. URL: https://netmarketshare.com/browser-market-share.aspx?id=browsersDesktopVersions.

[2] Reis, C. & Gribble, S. D. 2009. Isolating web programs in modern browser architectures. In *EuroSys*.

[3] Symantec. Symantec internet security threat report volume xiii: trends for july–december 07, 2008. URL: https://www.symantec.com/content/dam/symantec/docs/security-center/archives/istr-08-april-en.pdf.

[4] Symantec. Symantec internet security threat report volume xiv: trends for 2008, 2009. URL: https://www.symantec.com/content/dam/symantec/docs/security-center/archives/istr-global-09-april-volume-xiv-en.pdf.

[5] Symantec. Symantec internet security threat report volume 17: trends for 2011, 2012. URL: https://www.symantec.com/content/dam/symantec/docs/security-center/archives/istr-12-april-volume-17-en.pdf.

[6] Symantec. Symantec internet security threat report volume 21, 2016. URL: https://www.symantec.com/content/dam/symantec/docs/security-center/archives/istr-16-april-volume-21-en.pdf.

[7] Symantec. Symantec internet security threat report volume 22, 2017. URL: https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf.

[8] Symantec. Symantec internet security threat report volume 24, 2019. URL: https://interactive.symantec.com/istr24-web.

[9] TANENBAUM, A. 2010. *Computer Networks 5th edition*. Pearson Education. URL: https://www.ebook.de/de/product/30065351/a_tanenbaum_computer_networks.html.

[10] Heartfield, R. & Loukas, G. 2016. A taxonomy of attacks and a survey of defence mechanisms for semantic social engineering attacks. *ACM Computing Surveys*, 48(3), 37.

[11] Microsoft. Microsoft security intelligence report volume 8: July - december 2009, 2009. URL: https://go.microsoft.com/fwlink/?linkid=2047622.

[12] Microsoft. Microsoft security intelligence report volume 14: July - december 2012, 2012. URL: https://go.microsoft.com/fwlink/p/?linkid=2036147.

[13] F-Secure. Threat report 2015, 2016. URL: https://www.f-secure.com/documents/996508/1030743/Threat_Report_2015.pdf.

[14] Microsoft. 2014. Adware: A new approach. [Last Accessed: May 24th 2019]. URL: https://www.microsoft.com/security/blog/2014/04/02/adware-a-new-approach/.

[15] Microsoft. 2019. Potentially unwanted application (pua). [Last Acessed: May 24th 2019]. URL: https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/criteria#potentially-unwanted-application-pua.

[16] Symantec. Symantec internet security threat report volume xi: Trends for july–december 06, 2007. URL: https://www.symantec.com/content/dam/symantec/docs/security-center/archives/istr-07-march-en.pdf.

[17] Microsoft. 2019. Malware. [Last Accessed: May 24th 2019]. URL: https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/criteria#malware.

[18] Bishop, M. 2002. *Computer Security: Art and Science*. Addison-Wesley Professional. URL: https://www.amazon.com/Computer-Security-Science-Matt-Bishop/dp/0201440997?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0201440997.

[19] Nayak, K., Marino, D., Efstathopoulos, P., & Dumitras, T. 2014. Some vulnerabilities are different than others - studying vulnerabilities and attack surfaces in the wild. In *RAID*.

[20] Jones, J. 2012. State of web exploit kits.

[21] Kotov, V. & Massacci, F. 2013. Anatomy of exploit kits - preliminary analysis of exploit kits as software artefacts. In *ESSoS*.

[22] Microsoft. Microsoft security intelligence report volume 16: July - december 2013, 2013. URL: https://go.microsoft.com/fwlink/p/?linkid=2036139.

[23] Maio, G. D., Kapravelos, A., Shoshitaishvili, Y., Krügel, C., & Vigna, G. 2014. Pexy: The other side of exploit kits. In *DIMVA*.

[24] F-Secure. Threat summaries volume 1, 2014. URL: https://www.f-secure.com/documents/996508/1030743/threat_summaries_2006_2002.pdf.

[25] F-Secure. Threat report h2 2013, 2014. URL: https://www.f-secure.com/documents/996508/1030743/Threat_Report_H2_2013.pdf.

[26] Talos. 2015. Threat spotlight: Angler lurking in the domain shadows. [Last Accessed: May 20th 2019]. URL: https://blogs.cisco.com/security/talos/angler-domain-shadowing.

[27] Hopkins, M. W. & Dehghantanha, A. 2015. Exploit kits: The production line of the cybercrime economy? *2015 Second International Conference on Information Security and Cyber Forensics (InfoSec)*, 23–27.

[28] F-Secure. Threat report h1 2013, 2013. URL: https://www.f-secure.com/documents/996508/1030743/Threat_Report_H1_2013.pdf.

[29] Microsoft. Microsoft security intelligence report volume 6: July - december 2008, 2008. URL: https://go.microsoft.com/fwlink/p/?linkid=2036319.

[30] Microsoft. Microsoft security intelligence report volume 7: January - june 2009, 2009. URL: https://go.microsoft.com/fwlink/p/?linkid=2036168.

[31] Microsoft. Microsoft security intelligence report volume 9: January - june 2010, 2010. URL: https://go.microsoft.com/fwlink/p/?linkid=2036307.

[32] Microsoft. Microsoft security intelligence report volume 10: July - december 2010, 2010. URL: https://go.microsoft.com/fwlink/p/?linkid=2036302.

[33] Microsoft. Microsoft security intelligence report volume 11: January - june 2011, 2011. URL: https://go.microsoft.com/fwlink/p/?linkid=2036161.

[34] Microsoft. Microsoft security intelligence report volume 12: July - december 2011, 2011. URL: https://go.microsoft.com/fwlink/p/?linkid=2036296.

[35] Microsoft. Microsoft security intelligence report volume 13: January - june 2012, 2012. URL: https://go.microsoft.com/fwlink/?linkid=2036150.

[36] Microsoft. Microsoft security intelligence report volume 15: January - june 2013, 2013. URL: https://go.microsoft.com/fwlink/p/?linkid=2036144.

[37] Microsoft. Microsoft security intelligence report volume 17: January - june 2014, 2014. URL: https://go.microsoft.com/fwlink/p?linkid=2036137.

[38] Microsoft. Microsoft security intelligence report volume 18: July - december 2014, 2014. URL: https://go.microsoft.com/fwlink/p/?linkid=2036260.

[39] Microsoft. Microsoft security intelligence report volume 19: January - june 2015, 2015. URL: https://go.microsoft.com/fwlink/p/?linkid=2036257.

[40] Microsoft. Microsoft security intelligence report volume 20: July - december 2015, 2015. URL: https://go.microsoft.com/fwlink/p/?linkid=2036113.

[41] Microsoft. Microsoft security intelligence report volume 21: January - june 2016, 2016. URL: https://go.microsoft.com/fwlink/p/?linkid=2036108.

[42] Microsoft. Microsoft security intelligence report volume 22: January - march 2017, 2017. URL: https://go.microsoft.com/fwlink/p/?linkid=2036244.

[43] Microsoft. Microsoft security intelligence report volume 23: February - december 2017, 2017. URL: https://go.microsoft.com/fwlink/p/?linkid=2073690.

[44] Symantec. Symantec internet security threat report volume x: Trends for january 06–june 06, 2006. URL: https://www.symantec.com/content/dam/symantec/docs/security-center/archives/istr-06-sept-en.pdf.

[45] Symantec. Symantec internet security threat report volume xii: Trends for january–june 07, 2007. URL: https://www.symantec.com/content/dam/symantec/docs/security-center/archives/istr-07-sept-en.pdf.

[46] Symantec. Symantec internet security threat report volume 23, 2018. URL: https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-2018-en.pdf.

[47] F-Secure. Threat summaries volume 2, 2014. URL: https://www.f-secure.com/documents/996508/1030743/threat_summaries_2011_2007.pdf.

[48] F-Secure. Threat report h1 2012, 2012. URL: https://www.f-secure.com/documents/996508/1030743/Threat_Report_H1_2012.pdf.

[49] F-Secure. Threat report h2 2012, 2013. URL: https://www.f-secure.com/documents/996508/1030743/Threat_Report_H2_2012.pdf.

[50] F-Secure. Threat report h1 2014, 2014. URL: https://www.f-secure.com/documents/996508/1030743/Threat_Report_H1_2014.pdf.

[51] F-Secure. Threat report h2 2014, 2015. URL: https://www.f-secure.com/documents/996508/1030743/Threat_Report_H2_2014.

[52] F-Secure. The state of cyber security 2017, 2017. URL: https://www.f-secure.com/documents/996508/1030743/cyber-security-report-2017.

[53] F-Secure. Attack landscape 2018 h1, 2018. URL: https://blog.f-secure.com/attack-landscape-2018-far/.

[54] McGuire, M. 2018. Into the web of profit. URL: https://www.bromium.com/resource/into-the-web-of-profit/.

[55] Radianti, J. 2010. A study of a social behavior inside the online black markets. *2010 Fourth International Conference on Emerging Security Information, Systems and Technologies*, 189–194.

[56] Allodi, L. 2017. Economic factors of vulnerability trade and exploitation. In *ACM Conference on Computer and Communications Security*.

[57] Grier, C., Ballard, L., Caballero, J., Chachra, N., Dietrich, C. J., Levchenko, K., Mavrommatis, P., McCoy, D., Nappa, A., Pitsillidis, A., Provos, N., Rafique, M. Z., Rajab, M. A., Rossow, C., Thomas, K., Paxson, V., Savage, S., & Voelker, G. M. 2012. Manufacturing compromise: the emergence of exploit-as-a-service. In *ACM Conference on Computer and Communications Security*.

[58] Microsoft. 2016. Moving beyond emet. [Last Accessed: May 26th 2019]. URL: https://blogs.technet.microsoft.com/srd/2016/11/03/beyond-emet/.

[59] Windows. 2018. Windows sandbox. [Last Accessed May 28th 2019]. URL: https://techcommunity.microsoft.com/t5/Windows-Kernel-Internals/Windows-Sandbox/ba-p/301849.

[60] Microsoft. Activex controls. [Last Accessed: May 25th 2019]. URL: https://docs.microsoft.com/en-us/windows/desktop/com/activex-controls.

[61] Microsoft. Use activex controls for internet explorer 11 and internet explorer 10. [Last Accessed: May 25th 2019]. URL: https://support.microsoft.com/en-us/help/17469/windows-internet-explorer-use-activex-controls.

[62] Microsoft. 2009. Autorun changes in windows 7. [Last Accessed: May 25th 2019]. URL: https://blogs.technet.microsoft.com/srd/2009/04/28/autorun-changes-in-windows-7/.

[63] Hein, D. D., Morozov, S., & Saiedian, H. 2012. A survey of client-side web threats and counter-threat measures. *Security and Communication Networks*, 5, 535–544.

[64] The Chromium Projects. *Site Isolation Design Document*. URL: https://www.chromium.org/developers/design-documents/site-isolation.

[65] Barth, A., Felt, A. P., Saxena, P., & Boodman, A. 2010. Protecting browsers from extension vulnerabilities. In *NDSS*.

[66] Mozilla. Differences between api implementations. [Last Accessed: May 24th 2019]. URL: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Differences_between_API_implementations.

[67] Google. 2016. Moving towards a more secure web. [Last Accessed May 27th 2019]. URL: https://security.googleblog.com/2016/09/moving-towards-more-secure-web.html.

[68] Mozilla. How do i tell if my connection to a website is secure? [Last Accessed May 27th 2019]. URL: https://support.mozilla.org/en-US/kb/how-do-i-tell-if-my-connection-is-secure.

[69] Akamai. [state of the internet] / security q4 2017 report. Report, 2017. URL: https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q4-2017-state-of-the-internet-security-report.pdf.

[70] Gupta, B. B., Arachchilage, N. A. G., & Psannis, K. E. 2017. Defending against phishing attacks: Taxonomy of methods, current issues and future directions.

[71] Ion, I., Reeder, R., & Consolvo, S. 2015. "... no one can hack my mind": Comparing expert and non-expert security practices. In *SOUPS*, volume 15, 1–20.

[72] Satish, P. S. & Chavan, R. 2017. Web browser security: Different attacks detection and prevention techniques. *International Journal of Computer Applications*, 170(9). URL: https://pdfs.semanticscholar.org/87c1/36d91b8fcaefe133932ef841d99fda2657f1.pdf.

[73] Eshete, B., Villafiorita, A., & Weldemariam, K. 2011. Malicious website detection: Effectiveness and efficiency issues. In *2011 First SysSec Workshop*, 123–126. doi:10.1109/SysSec.2011.9.

[74] Sood, A. K. & Zeadally, S. 2016. Drive-by download attacks: A comparative study. *IT Professional*, 18(5), 18–25. doi:10.1109/MITP.2016.85.

[75] Nelms, T., Perdisci, R., Antonakakis, M., & Ahamad, M. 2016. Towards measuring and mitigating social engineering software download attacks. In *USENIX Security Symposium*, 773–789.

[76] Bosco, F. & Shalaginov, A. Identification and analysis of malware on selected suspected copyright-infringing websites. Report, EUIPO, 2018. doi:10.2814/004056.

[77] Cova, M., Kruegel, C., & Vigna, G. 2010. Detection and analysis of drive-by-download attacks and malicious javascript code. In *Proceedings of the 19th international conference on World wide web*, 281–290. ACM.

[78] Kolbitsch, C., Livshits, B., Zorn, B., & Seifert, C. 2012. Rozzle: De-cloaking internet malware. In *Security and Privacy (SP), 2012 IEEE Symposium on*, 443–457. IEEE.

[79] Edwards, B., Moore, T., Stelle, G., Hofmeyr, S., & Forrest, S. 2012. Beyond the blacklist: modeling malware spread and the effect of interventions. In *Proceedings of the 2012 New Security Paradigms Workshop*, 53–66. ACM.

[80] Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. 2009. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1245–1254. ACM.

[81] Xu, L., Zhan, Z., Xu, S., & Ye, K. 2013. Cross-layer detection of malicious websites. In *Proceedings of the third ACM conference on Data and application security and privacy*, 141–152. ACM.

[82] Wen, S., Zhao, Z., & Yan, H. 2018. Detecting malicious websites in depth through analyzing topics and web-pages. In *Proceedings of the 2nd International Conference on Cryptography, Security and Privacy*, 128–133. ACM.

[83] Shibahara, T., Takata, Y., Akiyama, M., Yagi, T., & Yada, T. 2017. Detecting malicious websites by integrating malicious, benign, and compromised redirection subgraph similarities. *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, 01, 655–664.

[84] MITRE. Honeyclient. [Last Accessed May 29th 2019]. URL: https://www.mitre.org/research/technology-transfer/technology-licensing/honeyclient.

[85] Grus, J. 2019. *Data Science from Scratch*. O'Reilly UK Ltd. URL: http://shop.oreilly.com/product/0636920179337.do.

[86] Microsoft. 2011. Support for ssl/tls protocols on windows. [Last Accessed May 27th 2019]. URL: https://blogs.msdn.microsoft.com/kaushal/2011/10/02/support-for-ssltls-protocols-on-windows/.

[87] Ramalho, L. 2015. *Fluent Python*. O'Reilly UK Ltd. URL: http://shop.oreilly.com/product/0636920032519.do.

[88] Symantec. Symantec internet security threat report volume 18: 2012 trends, 2013. URL: https://www.symantec.com/content/dam/symantec/docs/security-center/archives/istr-13-april-volume-18-en.pdf.

Appendices

A Detailed module listing for Crawler VM

Module listing created using the pip command **pip freeze** which lists all current modules and their version number:

```
backports.functools-lru-cache==1.5
beautifulsoup4==4.7.1
bgpranking-web==1.1.1
bs4==0.0.1
certifi==2018.11.29
chardet==3.0.4
configobj==5.0.6
cssselect==1.0.3
enum34==1.1.6
fake-useragent==0.1.11
idna==2.8
lxml==4.3.1
oletools==0.53.1
parse==1.11.1
pipenv==2018.11.26
pycairo==1.16.2
pycrypto==2.6.1
pygobject==3.26.1
pyparsing==2.3.1
pyparted==3.11.1
pyquery==1.4.0
python-apt==1.6.3+ubuntu1
python-xlib==0.20
pywebcopy==5.0.1
pyxdg==0.25
requests==2.21.0
setproctitle==1.1.10
six==1.12.0
soupsieve==1.8
typing==3.6.6
urllib3==1.24.1
virtualenv==16.4.0
virtualenv-clone==0.5.1
w3lib==1.20.0
```

The Crawler VM also utilized Python 3.6 so **pip3 freeze** lists the modules used for Python 3.6:

```
appdirs==1.4.3
apt-clone==0.2.1
apturl==0.5.2
asn1crypto==0.24.0
beautifulsoup4==4.7.1
boto==2.49.0
boto3==1.9.126
botocore==1.12.126
Brlapi==0.6.6
bs4==0.0.1
bz2file==0.98
certifi==2019.3.9
cffi==1.12.2
chardet==3.0.4
command-not-found==0.3
configobj==5.0.6
cryptography==2.6.1
cssselect==1.0.3
cupshelpers==1.0
decorator==4.4.0
defer==1.0.6
dnspython==1.16.0
docutils==0.14
EasyProcess==0.2.5
entrypoint2==0.0.0
et-xmlfile==1.0.1
fake-useragent==0.1.11
gensim==3.7.1
httplib2==0.9.2
idna==2.8
jdcal==1.4
jmespath==0.9.4
louis==3.5.0
lxml==4.3.1
macaroonbakery==1.1.3
Mako==1.0.7
Markdown==3.0.1
Markups==3.0.0
MarkupSafe==1.0
maxminddb==1.4.1
maxminddb-geolite2==2018.703
nemo-emblems==4.0.2
nltk==3.4
numpy==1.16.2
nyx==2.0.4
```

```
onboard==1.4.1
openpyxl==2.6.2
PAM==0.4.2
pandas==0.24.1
parse==1.11.1
patool==1.12
pexpect==4.2.1
Pillow==5.1.0
protobuf==3.0.0
psutil==5.4.2
pycairo==1.16.2
pycparser==2.19
pycrypto==2.6.1
pycups==1.9.73
pycurl==7.43.0.1
pyee==5.0.0
pyenchant==2.0.0
Pygments==2.3.1
pygobject==3.26.1
PyICU==1.9.8
pyinotify==0.9.6
pymacaroons==0.13.0
PyNaCl==1.1.2
pyOpenSSL==19.0.0
pyppeteer==0.0.25
PyQt5==5.12
PyQt5-sip==4.19.14
pyquery==1.4.0
pyRFC3339==1.0
PySocks==1.6.8
python-apt==1.6.3+ubuntu1
python-dateutil==2.8.0
python-debian==0.1.32
python-geoip==1.2
python-geoip-geolite2==2015.303
python-markdown-math==0.6
python-xapp==1.4.0
python-xlib==0.20
pythonwhois==2.4.3
pytz==2018.9
pyunpack==0.1.2
pywebcopy==5.0.1
pyxdg==0.25
PyYAML==3.12
reportlab==3.4.0
requests==2.21.0
```

```
requests-html==0.9.0
requests-unixsocket==0.1.5
ReText==7.0.4
s3transfer==0.2.0
scipy==1.2.1
sessioninstaller==0.0.0
setproctitle==1.1.10
singledispatch==3.4.0.3
six==1.12.0
smart-open==1.8.0
soupsieve==1.8
stem==1.7.1
stopit==1.1.2
system-service==0.3
systemd-python==234
tqdm==4.31.1
ubuntu-drivers-common==0.0.0
ufw==0.35
urllib3==1.24.1
w3lib==1.20.0
websockets==7.0
xkit==0.0.0
```

B Detailed module listing for Cuckoo Host VM

Module listing created using the pip command **pip freeze** which lists all current modules and their version number.

```
alembic==0.8.8
androguard==3.0.1
asn1crypto==0.24.0
beautifulsoup4==4.5.3
bottle==0.12.13
cairocffi==0.9.0
CairoSVG==1.0.22
capstone==3.0.5rc2
certifi==2018.1.18
cffi==1.12.0
chardet==2.3.0
click==6.6
colorama==0.3.7
configobj==5.0.6
configparser==3.5.0
cryptography==2.5
cssselect2==0.2.1
Cuckoo==2.0.6.2
distorm3==3.4.1
Django==1.8.4
django-extensions==1.6.7
dpkt==1.8.7
dumbnet==1.12
ecdsa==0.13
egghatch==0.2.3
elasticsearch==5.3.0
enum34==1.1.6
et-xmlfile==1.0.1
Flask==0.12.2
Flask-SQLAlchemy==2.1
functools32==3.2.3.post2
future==0.17.1
html5lib==1.0.1
HTTPReplay==0.2.4
idna==2.8
ipaddr==2.2.0
ipaddress==1.0.22
```

```
itsdangerous==1.1.0
jdcal==1.4
Jinja2==2.9.6
jsbeautifier==1.6.2
jsonschema==2.6.0
libvirt-python==4.0.0
M2Crypto==0.27.0
Mako==1.0.7
MarkupSafe==1.1.0
olefile==0.43
oletools==0.51
openpyxl==2.6.0
pdfrw==0.4
peepdf==0.4.2
pefile==2017.11.5
pefile2==1.2.11
Pillow==3.2.0
psycopg2==2.7.7
pycairo==1.16.2
pycparser==2.19
pycrypto==2.6.1
pydeep==0.4
pyelftools==0.24
pygobject==3.26.1
pyguacamole==0.6
pymisp==2.4.54
pymongo==3.0.3
pyOpenSSL==19.0.0
pyparted==3.11.1
Pyphen==0.9.5
python-apt==1.6.3+ubuntu1
python-dateutil==2.4.2
python-editor==1.0.4
python-magic==0.4.12
python-xlib==0.20
pythonaes==1.0
pyxdg==0.25
PyYAML==3.12
requests==2.13.0
roach==0.1.2
scapy==2.3.2
setproctitle==1.1.10
SFlock==0.3.8
six==1.12.0
SQLAlchemy==1.0.8
suricatasc==0.9
```

```
tinycss2==0.6.1
tlslite-ng==0.6.0
typing==3.6.2
unicorn==1.0.1
urllib3==1.24.1
vboxapi==1.0
wakeonlan==0.2.2
WeasyPrint==0.42
webencodings==0.5.1
Werkzeug==0.14.1
yara-python==3.6.3
```

C Save procedures for the crawler

Listing C.1: Save procedure

```
if index % 10 == 0:
        checked_domains = pd.DataFrame(checked_domains_list)
        checked_domains_path = complete_datapath + '/' + "checked_domains.csv"
        checked_domains.to_csv(checked_domains_path, index=False)
        data_storage_path = complete_datapath + '/' + "data_storage.csv"
        with open(data_storage_path, 'w') as outfile:
                writer = csv.writer(outfile)
                first_run = True
                for data_dict in data_storage:
                        if first_run == True:
                                first_run = False
                                writer.writerow(data_dict.keys())
                                writer.writerow(data_dict.values())
                        else:
                                writer.writerow(data_dict.values())
```

Listing C.2: Load procedure

```
try:
        with open('data/data_storage.csv', 'r') as infile:
                data_storage = []
                try:
                        imported_data = pd.read_csv(infile)
                        for i in range(0, len(imported_data)):
                                data_storage.append(imported_data.loc[i].to_dict())
                except pd.errors.EmptyDataError as e:
                        logger.error(e)
                        pass
except IOError as e:
        logger.error("No previous data storage detected")
        data_storage = []
try:
        checked_domains = pd.read_csv("data/checked_domains.csv",\
                 names=['url'], skiprows=1)
        checked_domains_list = list(checked_domains.url)
        logger.info("Loaded the following checked domains: "\
                 + checked_domains.url.to_string())
except IOError as e:
        logger.error("No previous checked_domains save detected.")
        checked_domains_list = []
```

D Post-duplication removal word description

Listing D.1: Words description on domains without JavaScript rendering enabled, post-duplication removal

```
Number of non-words (that are occurring more than 100 times): 50
Number of (any) words in body text that are occurring more than 100 times: 196
Total number of (any) words in body text: 63806
Total number of unique words: 130 Number of sites with content: 906
```

Listing D.2: Word description on domains with JavaScript rendering enabled, post-duplication removal

```
Number of non-words (that are occurring more than 100 times): 68
Number of (any) words in body text that are occurring more than 100 times: 286
Total number of (any) words in body text: 80019
Total number of unique words: 188 Number of sites with content: 886
```

Øyvind Jensen

NTNU
Norwegian University of
Science and Technology