Predicting content manipulations by open web proxies

A thesis submitted to the
College of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Zahra Nezhadian

Permission to Use

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Disclaimer

Reference in this thesis to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favoring by the University of Saskatchewan. The views and opinions of the author expressed herein do not state or reflect those of the University of Saskatchewan, and shall not be used for advertising or product endorsement purposes.

Requests for permission to copy or to make other uses of materials in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building, 110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan S7N 5C9 Canada

OR

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9 Canada

Abstract

The need for anonymity and privacy has given rise to open web proxies that act as gateways relaying traffic between web servers and their clients, allowing users to access otherwise not accessible content. As the open web proxy ecosystem continues to grow, more and more studies point out the extent of content alteration on the Internet. The content alterations applied by proxies include both benign and malicious modifications, such as adding crypto-mining scripts or adding injections. While some content modifications such as add injections can be prevented using blocker tools, adding scripts to JavaScript files cannot be detected with any antivirus or blocker tool. The widespread use of proxies and their malicious behaviour motivated us to focus on the feasibility of predicting these manipulations to choose a proxy for daily use carefully.

While the previous studies focused on the detection and analysis of content manipulation by proxies, we present a novel approach for predicting the types of content alterations that might be silently applied by open proxies. Besides, this approach allows us to predict the injection of any extra file by open proxies. The predictions in this study indicate changes without a need to fetch the data through a proxy first.

The leveraged dataset in this work is created by collecting website content of 1028 domains fetched through 1293 proxies as the initial steps of this study. Then, we derive 13 types of content modification through a detailed analysis of content manipulations on collected content. Then the detected content modification types are utilized to form our dataset for prediction analysis.

This research allows us to accurately predict proxy behaviour over a particular website, enabling us to recognize malicious and benign proxies and cautiously select a proxy to connect to. This study predicted the type of content modifications with 92% accuracy. In addition, the injection of extra files was predicted with 99% accuracy. Besides, our study reveals an important observation that the majority of proxies manipulate website content based on technical information of the website and its web server.

Acknowledgements

I would like to express my gratitude to my advisor, Dr. Natalia Stakhanova, whose sincerity and encouragement I will never forget. Her guidance and advice carried me through all the stages of writing this thesis.

I wish to express my strong appreciation to my wonderful parents whose constant love and support keep me motivated. I owe my deepest gratitude to my supportive spouse, Arash, for providing me the unconditional love and support throughout the entire thesis process.

I would also like to give special thanks to my friend Farzaneh Abazari, a postdoc member at The CyberLab, whose help, encouragement, and suggestions have contributed immensely to the evolution of my ideas on the thesis.

I would also like to thank our laboratory technician, Enrico Branca, for all the technical knowledge and cybersecurity perspectives he shared with me while working on different projects.

I also thank my dear friend, Mamraz, for kindly listening to my concerns and supporting me.

To my love.

Contents

Permission to Use . . . . . **i**

Abstract . . . . . **ii**

Acknowledgements . . . . . **iii**

Contents . . . . . **v**

List of Tables . . . . . **vii**

List of Figures . . . . . **viii**

List of Abbreviations . . . . . **ix**

1 Introduction **1**
1.1 Contribution 3
1.2 Thesis Structure 3

2 Background **5**
2.1 Proxy 5
2.1.1 What is a proxy 5
2.1.2 Proxy Types 5
2.1.3 Proxy application 8
2.2 HTTP Headers 8

3 Related work **10**
3.1 Proxy Behaviour Analysis 10
3.2 Web Injection and Content Manipulation Analysis 12
3.3 Network Interception Studies 14
3.4 Malicious URL Detection 15
3.4.1 Summary 16

4 Analysis methodology **17**
4.1 Collection 17
4.2 Content manipulation detection 20
4.3 Characterizing modifications 21
4.4 Prediction 24
4.5 Summary 27

5 Data Collection and Its Analysis **28**
5.1 Experimental setup 28
5.1.1 Data collection setup 28
5.1.2 Manipulation detection setup 29
5.1.3 Characterizing manipulation setup 29
5.1.4 Prediction setup 29
5.2 Collected proxies and domains 30
5.2.1 Proxies 30
5.2.2 Domains 30
5.2.3 Collected Files of Dataset 32
5.3 Summary 33

6 Experimental results 34
6.1 Collected Data Analysis 34
6.2 Prediction Analysis 37
6.3 Summary 40

7 Conclusion 44
7.1 Future Work 44

References 46

List of Tables

4.1 List of features 20
4.2 The parameters of the classification algorithms 26

5.1 The collected data 30
5.2 The number of types of files. 32

6.1 Content manipulation by proxies 34
6.2 Top ASNs of content manipulating proxies 35
6.3 The number of collected and manipulated files in detail. 36
6.4 The size of extracted content snippets (# of characters) 37
6.5 Classification Accuracy of experiments using all features 38
6.6 Classification accuracy excluding proxy and its features from dataset 39
6.7 The types of modification seen on domains with not configured HTTP headers 41
6.8 The features (IG$>$ 0.01) selected for modification type prediction experiments given Table 6.5 (shown in order of their ranking from most to less significant) 42
6.9 The features selected for prediction of injected files experiments given Table 6.5 (shown in order of their ranking from most to less significant) 43

List of Figures

2.1 Proxy mechanism 5

4.1 The overview of the analysis pipeline 18
4.2 The overview of the data collection pipeline 18
4.3 Classification of content modifications by types 22

5.1 Distribution of proxies and domains over countries. 31

List of Abbreviations

ASN	Autonomous System Number
CSS	Cascading Style Sheets
CSV	Comma-separated values
DMP	Diff Match Patch
DOM	Document Object Model
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
ISP	Internet Service Providers
JS	JavaScript
OS	Operating System
RIR	Regional Internet registry
SHA1	Secure Hash Algorithm 1
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
URL	Uniform Resource Locator
XML	Extensible Markup Language

1 Introduction

Web proxies act as gateways relaying traffic between web servers and their clients. By using proxies, requests are sent to the proxy instead of the web server, then the proxy forwards the request to the server and passes back the response to the client. When users communicate via proxies, they can hide their IP address to the server.

Proxies help users to protect their privacy by hiding IP addresses. Proxies also allow users to activate anonymously over the internet and give users access to restricted content (e.g., due to censorship or regional content restrictions). The advantages given to users are based on the point that by using a proxy, the user's network information is replaced by proxy's (e.g. IP address), and the internet has access to the proxy's information.

Open proxies are unrestricted proxies and open to the public that users can freely use them. The need for anonymity and privacy has given rise to open proxies that allow users to access otherwise not accessible content at no charge. Open proxies are easily found over the internet, and several websites provide lists of free proxies for users. A number of recent studies showed that while this ecosystem of web proxies continues to grow, it presents many challenges [8, 32]. The majority of open proxies are recognized as unavailable and unresponsive, while operational proxies either have slow connectivity or have suspicious behaviour and are unreliable [8, 32]. The operational proxies are not available for more than a couple of days [32].

While the proxy ecosystem is fairly diverse, a number of recent studies showed that open web proxies have also been used for less legitimate purposes. Proxies have been observed injecting malicious JavaScript codes mining cryptocurrencies or fingerprinting and tracking data [32]. In addition, proxies have modified content fetched through them, such as injecting advertisements or requesting unsafe websites [39]. Further, Mani et al. [23] have detected man-in-the-middle (MITM) attacks operated by proxies and binaries, including malwares injected through proxies. Many proxies are blacklisted for spamming or other malicious behaviour [8]. Tsirantonakis et al. [39] stated that around 38% of working proxies modify the page somehow, and 5 - 10% of the responsive and operating proxies perform some suspicious behaviour [32].

By detecting unwanted behaviour in open proxies and continuous growth of their ecosystem alongside the rising demand to use proxies for different purposes, this study ventures to understand whether it is possible to proactively determine a possibility of changes by proxy and characterize specific types of content manipulations.

The importance of determining the possibility of manipulation by proxies relies on the fact that a considerable amount of manipulations made by proxies can not be detected by any antivirus or blocking tools,

i.e. tools that block ads injected in a webpage, such as injections in JavaScript files that will lead to unwanted behaviour on the user's system. Thus, the best solution against proxy manipulations is a preventive approach by which users can determine a possible behaviour of proxy before connecting to it that can be either malicious or benign.

Our work complements earlier studies by Tsirantonakis et al. [39] and Perino et al. [32]. Similar to these works, we analyze content manipulations performed by open proxies. We, however, take a step beyond detecting and analyzing the level of modifications and offer a prediction analysis of these modifications based on the technical characteristics of the website.

Manipulation is defined as any alteration and change made to the content of a website by proxies, which can be divided into three main behaviours: injection, modification, and deletion. An injection is defined as the addition of extra files in a connection through a proxy. The extra file is only received in a connection through a proxy and is not retrieved from the website in the direct connection. Modification is defined as any change or alteration applied to the content of a file in which the content of the file is modified by a proxy and is not identical to the original content of the file. Deletion is the behaviour by which some files are not sent through proxy and the user does not receive those files.

This work collects the content of websites from 1028 domains through 1293 open web proxies. Then to understand the misbehaviour of proxies, by leveraging machine learning and clustering techniques, the modifications introduced by proxies are detected and categorized into 13 detailed types, e.g., random placement of code, injection of trackers, injection of comments. We offer an analysis over modified content along with their type to give insight into proxy behaviour.

Within our collection process, beside the website content, the technical, network, server characteristics of domains and network information of proxies are retrieved to investigate their effect on potential content manipulations by proxies.

Our analysis is split into two parts to investigate manipulations applied by proxies. First, we investigate the modifications types performed by proxies. We leverage several machine learning models to predict the type of content modification applied to web page contents. We utilize the technical information collected about domain and proxy in predicting analysis. We further analyze the modification types and technical features to learn more about the proxy behaviour. By predicting the modification type applied by a proxy, we can find the answer to *What does determine the types of content modifications introduced by proxies?* question. In addition, we can have an insight into the behaviour that a proxy will have while connecting to a website without fetching the webpage content through the proxy.

Second, we investigate the injections made by proxies. Similar to modification analysis, several machine learning models are utilized to predict the injections. The technical characteristics collected for domains and proxies are used in the analysis. The key point in this analysis is that not only we do not need to fetch content through a proxy, but also we do not need to fetch content at all. This experiment only utilizes domain and proxy technical information and does not use any file-related features. We execute multiple prediction

experiments to predict injected files' presence, type, and source.

1.1 Contribution

Ultimately, this work presents two contributions that expand our understanding of proxy ecosystem behaviour and the direction of potential defences against content modification.

- **Predicting the content manipulation applied by open proxies:** we present a novel approach that allows us to proactively predict the content modification behaviour based on technical characteristics of a domain, server, and proxy. Our results illustrate that our approach can accurately (with around 90.0% accuracy) and, what is more important proactively (without a need to fetch data through a proxy) determine the types of changes that may be introduced by proxy. Our approach can predict whether a proxy will inject additional files and indicate the source and type of these files with 83.9% to 99.0% accuracy.
- **Open proxies alter domain contents based on the website and its web server's technical information:** through our analysis, we illuminate how proxies differentiate domains for possible content manipulations. Our study reveals an important observation that the majority of proxies manipulate website content depending on the technical information of the website and its web server. Our approach is the first attempt to understand the tactics of open proxies and their strategies for altering website content.

1.2 Thesis Structure

We organized our research's content in seven chapters as follows:

- **Chapter 1: Introduction**: This chapter introduces our research's general overview while highlighting its importance and specific contributions to the scientific field.
- **Chapter 2: Background**: This chapter provides a short introduction to proxy concept and briefly explains some of HTTP headers discussed in this study.
- **Chapter 3: Related Work**: This chapter provides a brief literature review on several studies about the proxy environment and potential behaviour of open proxies.
- **Chapter 4: Analysis Methodology**: This chapter gives a step-by-step explanation of our data collection process and how we find and extract manipulations injected by proxies. Moreover, we explain the detailed procedure of analyzing modifications and labelling them as well as the prediction process.
- **Chapter 5: Data**: This chapter explains the tools and the experimental setup, in addition to the source and amount of collected data set.

- **Chapter 6: Experimental Results**: This chapter presents the results of our work, which shows that we can determine the type of content modification by proxies. Moreover, we show that we can perfectly predict if there will be any file injection to a specific domain through a particular proxy. We can further predict the source and type of the injected file.

- **Chapter 7: Conclusion**: This chapter presents the conclusions drawn after parsing, analyzing and attributing cryptographic keys originating libraries and operating systems. Additionally, we suggest new and exciting research opportunities where our approach can be employed to derive or extend our research.

2 Background

2.1 Proxy

2.1.1 What is a proxy

The proxy server is simply a computer connected to the internet that serves the clients by receiving and forwarding their requests to the destination server. Proxy operates as a gateway that relays traffic between the end-user and the internet. Similar to any other device connected to the internet, the proxy server has its own IP address. It acts as an intermediary between the user and targeted destination (i.e. server or website) [37]. When a user connects to a proxy, that proxy turns to the client's representative over the internet, and the proxy performs all user's activity.

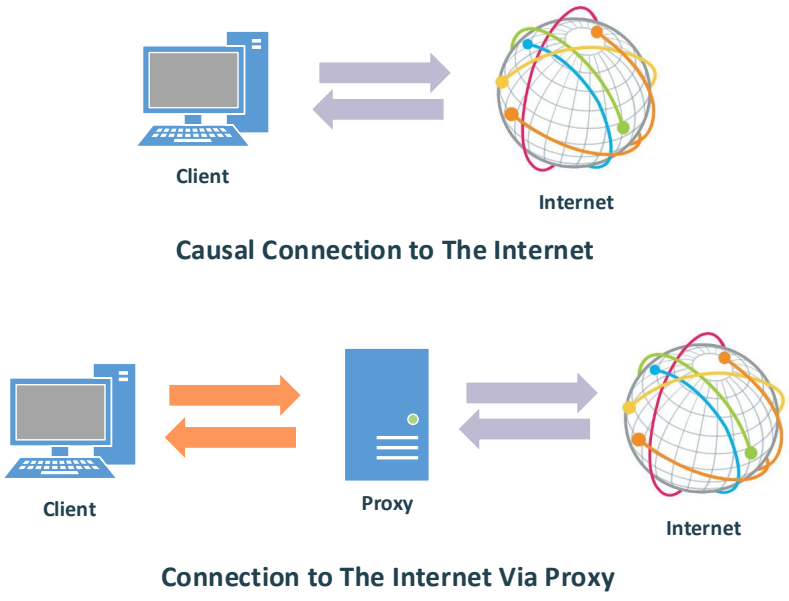


Figure 2.1: Proxy mechanism

2.1.2 Proxy Types

There are different types of proxies based on their functionality, protocols used, and their host. Each of these categories has its own types of proxies. This work focuses on forward data center HTTP and HTTPS proxies.

The two types of proxies based on their functionality are as follows.

- **Forward Proxy**: A forward proxy is the typical type of proxy that provides services on the client side. A forward proxy is an intermediary that receives the client requests, and then the proxy sends the requests to the destination server. The proxy first receives the response of the request and then forwards it to the client. Thus, the proxy is known as a client for the server and a server for the client [3].

- **Reverse Proxy**: Unlike the forward proxy, the reverse proxy provides services on the server side. The client requests are first received by proxy; then, if the requests are validated by proxy, the requests are forwarded to the server. Reverse proxies prevent direct and not validated access to critical data stored on servers. Varnish and Squid are two types of the reverse proxy [3].

Besides, proxies can be categorized based on the communication protocol they utilize.

- **SOCKS Proxy** [37]: A SOCKS proxy utilizes SOCKS protocol in communication. SOCKS, which stands for SOCKet Secure, is designed to route any type of traffic generated by any protocol or program. SOCKS proxy server establishes a TCP connection to communicate with another server. The SOCKS proxy server can bypass potential restrictions applied by a firewall monitoring HTTP ports.This kind of proxy is mainly used to bypass a firewall.

- **HTTP Proxy** [37]: An HTTP proxy leverages HTTP protocol for communication, which makes it suitable for web scraping. An HTTP proxy establishes a connection on the client's behalf by creating a tunnel between the browser (client) and a web server. HTTP proxy filters content by using HTTP protocol and have high performance, it also provides anonymity.

- **HTTPS Proxy**: An HTTPS proxy utilizes both HTTP and transport layer security (TLS) protocols for communication. The proxy establishes a secured HTTP connection by encrypting the data with the TLS protocol in the transport layer. HTTPS proxies allow users to communicate over the internet securely and anonymously.

- **FTP Proxy**: An FTP proxy acts as a gateway for FTP protocol and provides control over connections to the server. The source and destination of connections can be controlled and limited; user authentication can also manage and limit user access.

Proxies can also be grouped based on the IP address.

- **Data Center Proxy**: Data center proxy is the most common type of proxy that resides in a data center. The IP addresses of data center proxies are provided by cloud services such as AWS and Digital Ocean, so they are not associated with ISPs. The data center proxy is only an IP address in the cloud service that is used by many users at the same time. This type of proxy provides high performance

and speed and allows clients to hide their actual IP address, which is easily triggered as a proxy over the network. The IP address of data center proxies are not static and rapidly change [31].

- **Residential Proxy**: Residential proxy is associated with an IP address that belongs to an actual device; thus, residential proxies are affiliated with ISPs. The client's traffic is passed through a server and then a device in a connection over a residential proxy. Residential proxies are more challenging to detect as a proxy, and they are less likely to be banned [31].

- **ISP Proxy**: IP addresses of ISP proxies are provided by ISPs, but they are hosted on data centers and does not associate with an end user. ISP proxies have a combination of data center and residential proxy benefits. It is difficult to detect and ban these IPs as proxies, and these proxies allow high performance and speed [31].

- **Mobile Proxy**: Mobile proxy is associated with a mobile IP address provided by ISPs for mobile cellular internet connection. The mobile proxy can be on any device that provides a cellular internet connection. Mobile proxy is always associated with a device, and getting triggered as a proxy is even more difficult than the residential proxy. This kind of proxy can be used for managing multiple social accounts [2].

In addition, web proxies are one of the most common anonymizers among internet users. One of the main reasons for users to take advantage of proxies is hiding their identity and providing themselves with some privacy.

- **Transparent Proxy**: Transparent proxy provides no level of anonymity and reveals the user's IP address. The transparent proxy is not configured on end user devices and acts as an intermediary between user and server for authentication and monitoring purposes. The user IP address is sent to the server in the X-FORWARDED-FOR HTTP header [19].

- **Anonymous Proxy**: An anonymous proxy provides some level of anonymity by hiding the user's IP address from the server, but the server knows the connection is made through a proxy. Anonymous proxy provides such information in the VIA_PROXY HTTP header [19].

- **High Anonymity Proxy**: High anonymity (elite) does not reveal the user's IP address and the fact that the connection is through a proxy. It passes neither of the X-FORWARDED-FOR nor VIA_PROXY HTTP headers to the server [19].

- **Distorting Proxy**: Distorting proxy leverages a different method in keeping the user anonymous. It changes the the user IP to false IP address in the X-FORWARDED-FOR HTTP header and identifies itself as proxy in the VIA_PROXY HTTP header.

2.1.3 Proxy application

Proxies are widely used around the world due to the several advantages provided by using them. The main benefits are explained.

- **Anonymity and Privacy**: By communicating over a proxy, the client's IP address is masked which helps the user hide the personal information. Because the proxy traffic will be tracked and not the user's actual IP address, the user is seen as another IP over the internet.
- **Bypass restriction**: The user's IP address changes as a result of connecting to a proxy, and the user can have access to the content that is blocked for his original IP address.
- **Performance**: In cases that a proxy uses to cache data, when a user connects to a website for the first time, proxy caches the necessary information. Then proxy will use the cache data in the second connection and send the information faster than the regular connection.
- **Control**: Organizations can use proxy to restrict access to specific websites or limit their traffic as well as track their activity. Similarly, parents can limit the children's access to particular websites or track their activity and limit it.
- **Security**: Having an indirect connection to suspicious websites prevents users from getting infected by potential malware. In addition, tracking personal information will be more difficult by connecting to a website through proxy.

2.2 HTTP Headers

When a browser tries to load the website's content, it establishes a connection to the webserver. The communication between web server and browser is over HTTP protocol. HTTP headers allow clients and servers to define a structure/set of specific rules for each HTTP connection. An HTTP header filed is comprised of its name followed by a colon and its value. There are numerous HTTP header fields and they can be categorized based on their functionality and effect in the connection. Some of the main groups of headers are security, authentication, caching, cookies. The security headers discussed in this work are explained below.

- **Cross Origin Resource Policy**: This response-type HTTP header controls access to resources of the website. This header helps the server protect against cross-origin or cross-site embedding of the returned source, which is about loading resources from other origins such as other domains or sites.
- **Content Security Report Only**: The content security report only response-type header is used for testing the policies. This header allows developers to keep an eye on the effects of the tests. The

violation reports are documented in JSON format and sent as an HTTP POST message to the specified URI in this header.

- **Except CT**: This response-type header prevents using misissued certificates for a site. It also allows sites to enforce the certificate transparency(CT) requirements.
- **Feature Policy**: The feature policy response header allows websites to manage additional features and APIs of web browsers. These features can be toggled on or off while surfing a website.
- **X Permitted Cross-Domain Policies**: This response header allows the domain to give clients permission to load content from their domain or restrict access to its content.

3 Related work

The content of this chapter is presented in four sections, and each section gives a brief summary of previous related works. We have separated the related studies into proxy behaviour analysis, network interception studies, web injection and content manipulation analysis subsections, and malicious URL detection.

3.1 Proxy Behaviour Analysis

The proliferation of open web proxies on the internet has necessitated several studies to characterize the nature and behaviour of proxy ecosystem. One of the most related studies to our work that explored misbehaviour of proxies was accomplished by Tsirantonakis et al. [39]. They introduced an approach for the detection of malicious content modification by open web proxies based on the expected DOM (Document Object Model) elements of the website DOM tree. They analyzed DOM tree in HTML file of the honey sites designed for this study and *bbc.com* as a realistic website sample. Their analysis of over 66K proxies revealed that 5.15% of proxies engage in suspicious content manipulation activities. Their study was the first to outline general types of malicious misbehaviour by proxies: advertisements, tracking, fingerprinting, privacy leakage, malware, and unclassified behaviour. Although they did a deep analysis on HTTP open proxies' behaviour, they excluded HTTPS proxies and conducted their experiment on honeysites and only one real website. consequently, the proxies behaviour might differ when analyzed over large number of real websites.

Mani et al. [23] broadly investigated availability, diversity, success rate, and misbehaviour of 107,000 open proxies over a period of 50 days. 92% of the collected proxies were unresponsive to proxy requests. Researchers observed proxy misbehaviour by fetching various files from geographically spread webservers that resulted in having 8% of proxies providing unexpected content at least once and 4% of proxies constantly sending unexpected content. The expected content was defined as the correct files that were originally uploaded on webservers and unexpected contents are the returned files by proxy that does not match the original content. In addition, they observed content manipulation of files focusing only on HTML files where 2.2% of HTML content manipulations were considered as malicious. Malicious misbehaviour of proxies was defined as ad injections, cryptojacking, and potential eavesdropping in this study.

A similar study was conducted by Perino et al. [33], which offered an insight into the proxy ecosystem by monitoring the behaviour of 230K free proxies over a period of almost two years. They mostly confirmed some of Mani et al. [23]'s findings, indicating that around 5%–10% of the working proxies exhibit suspicious behaviour. Their experiment setup crawled realistic websites and a honey site designed by the team. Not

only HTML file content manipulation was inspected, but also Java Script files and SSL certificates were taken into account. Their result showed that 1% of all HTTPS proxies interfered in TLS handshake by replacing the original certificate with a self-signed one. While Perino et al. [33] analyzed both HTTP/S proxies over real websites, their explanation was general and did not give details.

Perta et al. [34] focused on privacy and anonymity provided by commercial VPN(Virtual Private Network) services. This work investigated the 14 most popular commercial VPNs. Their study showed that all analyzed VPN services suffer from a percentage of IPv6 traffic leakage. They further investigated the security of the VPNs by performing a DNS Hijacking attack over VPN traffic; this attack gave them access to all victims' traffic.

With a focus on a comparative analysis, Choi et al. [8] analyzed open and residential proxies. They analyzed proxies' reputation by observing their presence in 27 different blacklists, alongside analysis of the geological distribution of proxies at both country and city-level. They found that most open proxies (80%) were blacklisted, among which 28% were labelled as spam, and 7% of the proxies were considered to launch attacks. Similarly, 86% of residential proxies were blacklisted, with 17% of them recognized as spammer proxy, and a small number of residential proxies were found to associate with adversary attacks.

Mi et al. [28] conducted the first research on residential proxies' behaviour and ecosystem. They leveraged a novel method to automatically find residential proxy IPs which resulted in the detection of 6 million residential proxy IPs and 500K hosts across 230+ countries. This study found a surprising fact about residential proxies; although providers asserted the proxy hosts have willingly joined their network, many hosts were compromised, including IoT devices. In addition, they found illegal activity on residential proxy hosts such as phishing, malware hosting, and illegal advertisement. This work showed that residential proxies are no more reliable means of communication than open proxies.

Weaver et al. [45] leveraged the Netalyzr, a network connectivity test service, to measure the prevalence of HTTP proxies on the Internet. A number of 645,000 IP addresses, so called clients, were investigated among which 14% of the clients showed evidences of presence of web proxies. In addition, the observed hidden proxies showed different behaviour such as caching, 404-rewriting, content injection, local antivirus and spyware.

The works reviewed in this section analyzed various types of proxies and their behaviour, and these works are the most relevant studies to our research. However, our work goes a step forward and tries to predict the proxy behaviour. This work is inspired by previous studies [39, 23, 33] that investigated open proxy content modification, while we predict the proxy content manipulation behaviour and look for the reason and source of content manipulation. The techniques and approaches employed in this study are similar to Tsirantonakis et al.'s [39] approach since we use similar tools to collect data and similar techniques in comparing the collected data. The main difference between this study and reviewed works is that our focus is on predicting and finding the reason and source of content manipulation. In addition, our approach in collecting manipulated content through proxies is different from these works, and unlike others, we analyze

both HTTP and HTTPS proxies.

3.2 Web Injection and Content Manipulation Analysis

We also review similar studies that focus on detecting web content injection and manipulation. A major number of reviewed works in this part cover ad injection. The injection of fraudulent and malicious advertising (aka malvertising) is also a well-known instance of malicious content modification and injection. There are a large number of studies on the analysis and detection of malvertising network activities.

Thomas et al. [38] have carried out a broad study over ad injection, which is one of the common types of content injections over the internet. They developed a scanner that scans the DOM (Document Object Model) tree of a webpage to identify the presence of swindler ad elements in the DOM tree. Over 1 million Chrome extensions and 25 million Windows binaries were examined in this work that resulted in finding over 50,000 Chrome extensions and 34,500 Windows binaries injecting ads. 38% ad injector extensions and 17% of ad injector Windows binaries are found to be malicious. Besides, this research revealed that ad injectors financially profit from over 3,000 brands that have to pay for the imposed ad traffic on their websites by ad injectors.

Xing et al. [47] conducted a study similar to Thomas et al. [38] research, which specifically focused on ad injecting browser extensions and their malvertising activities. Over 18,000 Chrome extensions were examined by their tool that automatically detected ad injections and labelled them as benign or malicious based on their landing website. 292 ad injecting extensions were found, 56 of which were detected to participate in malvertising. Furthermore, researchers showed that some of the malicious extensions changed the visited web page's content to force users to install malware, while other ad injecting extensions injected ads that participate in malvertising.

Zhang et al. [50] observed web injection through recommender systems, sort of information filtering systems that try to predict user's rating or preference toward an item, by analyzing four real-world recommender systems and their activities using machine learning and the publicly available features. Their study revealed that recommender systems were vulnerable to web injections and malicious manipulations. The researchers conducted three different types of promotion attacks to test the viability of manipulations. In addition, they presented countermeasures against manipulation on recommender systems.

Arshad et al. [4] proposed a Chromium extension, EXCISION, which automatically detects and blocks malicious contents loaded from third-party sources into the user's browser. They examined the corresponding URLs of the sequence of resource inclusions by employing HMM classification. This work focused on analyzing the source URL of the included content, while injected codes or malicious modifications made to benign files can not be detected and requires content analysis.

Alrwais et al. [1] conducted an extensive study on details of a massive ad fraud while it was in action. Also, The paper recommended some approaches to mitigate such attacks. The significant difference of this ad

fraud was its focus on human clicks for the fraud instead of click bots, programs that are designed to conduct click fraud. They reported central part of the attack was based on a malicious DNS changer, a malware that changed the DNS resolver setting of victims to DNS resolvers controlled by attackers. This study reported that the attack stole revenue of ad clicks and ad impressions from 20 different ad networks by misdirecting humans. In addition, this work recommended investigating open recursive resolvers to mitigate such attacks.

Zarras et al. [49] crawled more than 600,000 web advertisements using Selenium WebDriver and studied security aspect of advertisements. They demonstrated different circumstances that users were more prone to malvertisement by analyzing both web advertisements and the websites publishing them. Their work found that websites not having an exclusive agreement with an advertiser can potentially publish malicious ads.

Vratonjic et al. [42] observed multiple vulnerabilities of ad serving systems caused by on-the-fly modification. Then they showed how these vulnerabilities can be leveraged to conduct an attack and how feasible such attacks are. They used the idea of hash-chain to develop a method to provide data integrity via collaborating with the parties involved in advertising systems. They claimed the proposed method was more efficient than HTTPS.

Beside analyzing malvertising and related attacks Li et al. [20] presented MadTracer, which is a malvertizing detection tool with novel approach compared to similar works. MadTracer detected malvertizing cases 15 times as many as Google Safe Browsing and Microsoft Forefront did together and detected newly found attacks. They crawled the first 90,000 home pages of the Alexa list and found that 1% of them were exploited to have malvertizing such as fraudulent clicks or delivering malicious content.

Masri and Aldwairi [25] investigated the accuracy of VirusTotal, URLVoid, and TrendMicro, the cybersecurity online services that provide URL reputation checking service, in detecting malicious advertisements. They collected data from 600 websites, extracted advertisements, and submitted URLs of ads into the mentioned services to find out about the maliciousness of the ads. They reported that URLVoid had the most accuracy in detecting malicious ads among VirusTotal, URLVoid, and TrendMicro services.

Besides the detection of malvertisement, Cai et al. [6] observed the security and privacy aspect of online advertising in survey form. The threats to online advertisements and the technical cause of threats were covered in this survey. The threats analyzed in this work were fraud, ad injection, privacy theft, and malvertising. All threats shared the weak client-side security as a cause of the mentioned threats; also, weak regulations in the online advertising industry and violation of the internet principles were considered as other reasons.

Content manipulation and web injections are not limited to advertisement, Tyson et al. [40] investigated the content of HTTP header manipulation and its frequency in different types of networks. They reported that hosts from 25% of measured ASes have received modified headers at least once. In addition, they claimed that some HTTP headers were configured based on the region, such as abandoning cache applied in network of advanced countries. They also observed some behaviours of middleboxes such as injecting cookies, disabling performance-enhancing features.

All the works reviewed in this section focused on detecting content injection or content manipulation, among which the majority investigated ad-related behaviour. The mutual point between this study and the reviewed ones is their goal to detect and observe web content manipulation, which leads to the employment of similar techniques and tools, such as utilization of the Selenium WebDriver in [49] study and our work to scrape webpages, or using a similar approach to [38] research in analyzing webpage contents.

3.3 Network Interception Studies

All the studies in previous section pointed out the extent of content monitoring and alteration on the Internet. A deeper look at end-to-end violation was conducted by Chung et al. [9] who investigated end-to-end violations in DNS, HTTP, and HTTPS protocols aiming to identify the party responsible for the violations. They reported that up to 4.8% of nodes were subject to some end-to-end connectivity violation. They investigated 1.2m nodes across 14k ASes in 172 countries. Around 1% of nodes passed modified HTML files in HTTP analysis, and in HTTPS analysis, certificate replacement was observed in 0.5% of nodes. Moreover, they inspected content monitoring mainly conducted by anti-virus software and ISP-level middleboxes. This work's approach was based on using Luminati proxy service, which enabled them to route HTTP/S traffic via many Hola nodes and gain visibility into their networks.

Durumeic et al. [13] investigated HTTPS interception based on TLS handshake characteristics. Their large-scale analysis primarily focused on the prevalence and impact of HTTPS interception. Their study showed that most intercepted connections became less secure using weaker cryptographic algorithms. They also showed that network middleboxes were the reason for most broken connections and not client-side security.

Similarly, O'Neil et al. [30] explored the prevalence of TLS proxies reporting cases of negligent or malicious behaviour. By probing TLS connections, they found one TLS-proxied connection per 250. They reported more than 1000 cases in which malwares used TLS-proxies, where these malicious TLS-proxies dynamically inserted advertisements on secure websites. They employed a flash app for certificate mismatch detection. Observing from another point of view, C. de Carnavalet and Mannan [12] studied the security of TLS interception tools, such as anti-viruses and parental-control software that interpose a TLS proxy. They examined 14 leading TLS interception tools on windows by their proposed framework and investigated the certificate generation process and security of the private key in the softwares. This study uncovered vulnerabilities introduced by 14 software. Most softwares were vulnerable to MITM attacks, where one of these attacks was full server impersonation.

Waked et al. [43] developed a framework to investigate TLS inspecting appliances and uncovered multiple security issues relating to TLS version and certificate parameters. They also reported that all tested appliances change TLS parameters compared to the proxy-to-server parameters, such as the TLS versions, hashing algorithms, and RSA key sizes. Some of the revealed vulnerabilities in this work were: insufficient private

key protection, improperly validating or not validating certificate parameters resulting in MITM attacks.

The works reviewed in this section investigate different types of manipulations. Our study is based on the same concept of content injection and content modification as these papers, while we cover this concept from a proxy point of view and move it one step further to predict such behaviour and not only detect it. Since our work analyzes proxy behaviour, network behaviour, over web contents, the findings are similar to previous studies separately focusing on networks interceptions.

3.4 Malicious URL Detection

While we observe the behaviour of open proxies over webpages and their effects on the client side, some previous studies have investigated the webpage's behaviour and effect on the client side by a focus on detecting malicious URL and webpages.

Hou et al. [17] investigated the classification of malicious web pages with the help of machine learning approaches. They collected the dataset's dynamic HTML code of 1141 benign and malicious web pages. They analyzed each webpage and extracted lexical features, besides calculating n-grams of Java Script functions as attributes. They set the label of each webpage as benign or malicious and the exact type of attack observed in the webpage, then used as the target in the classification method. Leveraged machine learning models in this study were Decision Tree, Naive Bayes, SVM, Boosted Decision Tree, among which Boosted Decision Tree gained the best results with 96% of accuracy. They also reported that the most practical features were the frequency of Java Script functions.

Mamun et al. [22] conducted a study on detecting and classifying malicious URLs based on their attack types using machine learning methods. They created a data set of 114,000 malicious and benign URLs from 4 different attack types. Lexical features were extracted for the analysis. Information gain and CFSSubsetEval feature selection algorithms were applied to the dataset prior to machine learning models. The leveraged models were k-n neighbour, random forest and a tree-based classifier named C4.5. Two different experiments were accomplished in this work. In the first approach, the dataset was split based on the attack type, and malicious URL detection was applied on each dataset separately, which resulted in 97 to 99% accuracy. In the second approach, the entire dataset was leveraged with five classes indicating benign URL and four malicious URLs with their specific attack type, which gained 94% accuracy.

Vanhoenshoven et al. [41] leveraged several machine learning models to detect malicious URLs, such as namely Naïve Bayes, Support Vector Machines, and Random Forest. They used a dataset containing 2.4 million URLs, with 3.2 million features. Having many attributes added a high calculation overhead to feature selection. Thus, they created three different feature sets with different groups of attributes; the first dataset contained all features with a correlation coefficient higher than a random value of 0.2. The correlation coefficient was calculated using the Pearson correlation algorithm. The second dataset contained only binary attributes that have a Pearson correlation coefficient of more than 0.1. The last dataset contained only actual

value features without any reduction in features. Their classification results showed that most methods have 90% accuracy, and random forest had the best results with 97.7% accuracy over the first dataset.

While our work observes the proxy behaviour and the manipulations made by proxy on the webpages and detects the type of content manipulations by applying machine learning techniques, the studies reviewed in this section mainly try to detect malicious webpages and URLs with their specific type of attacks by leveraging machine learning models.

3.4.1 Summary

Most of the studies reviewed in this chapter analyze different sorts of network interception and content manipulations from various points of view. A subset of these works investigated network interception in TCP and application layer network connections. Another group of studies conducted more specific research and analyzed website content manipulations which mostly covered ad-related behaviour. More specifically, some works focused on HTTP proxy behaviour and content manipulations. They analyzed the proxy behaviour and detected content manipulations. Our study is based on the content manipulation concept but focuses on predicting the potential content modification and injection executed by HTTP and HTTPS proxies, and finding the source manipulations.

4 Analysis methodology

The overview of our processing pipeline is presented in Figure 4.1. Our approach comprises four modules: data collection, content modification detection, characterizing modifications, and prediction module. Given a list of open web proxies and domains to analyze, we begin the collection process that retrieves website content, website technology, website and proxy network information and web server information. Once data is collected, website content is parsed, and the modified content snippets are extracted for analysis. We leverage clustering to determine the types of modifications present in our data. This effectively gives us labels that we can further use in prediction analysis.

The ultimate goal of our approach is to predict the types of changes one can expect in website content when visiting a website through a web proxy. As such, our prediction analysis takes into account all contextual information gathered about the website, a server, a proxy, and a generated label *without modified content*.

4.1 Collection

Several studies previously reported the facts of content manipulation by proxies [39, 32, 23]. Due to the volatility of proxy ecosystem, it is challenging to reliably identify proxies that consistently make modifications. We thus venture to collect a set of proxies for our analysis and test their behaviour for possible content manipulation.

We collected proxy lists advertised by four different proxy sites: clarktem[1], sunny9577[2], proxyscan.io[3], and augmented the lists with additional results collected with ProxyBroker. [10] We contacted these sources four times throughout August 2021. The received lists were deduplicated and tested for availability. The operating proxies were retained for further data collection.

To have a baseline for comparing proxy behaviour, we contacted a selected set of web domains through the tested proxies and directly without a proxy. The flow of data collection is shown in Figure 4.2. Given a list of proxies and domains, our data collection module randomly picks a domain URL from the list and first establishes a connection to verify its availability. If the domain is operational and the website is available (response code: HTTP 200 OK), we collect server and technology information and load the website's contents directly.

[1] https://raw.githubusercontent.com/clarketm/proxy-list/master/proxy-list.txt
[2] https://sunny9577.github.io/proxy-scraper/proxies.json
[3] https://www.proxyscan.io/api/proxy?type=http,httpslimit=600

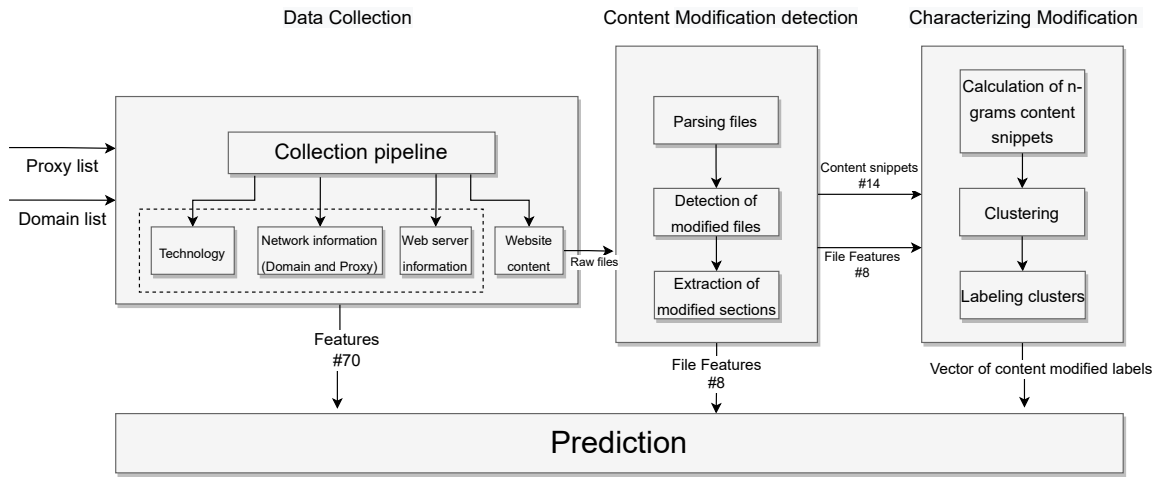


Figure 4.1: The overview of the analysis pipeline

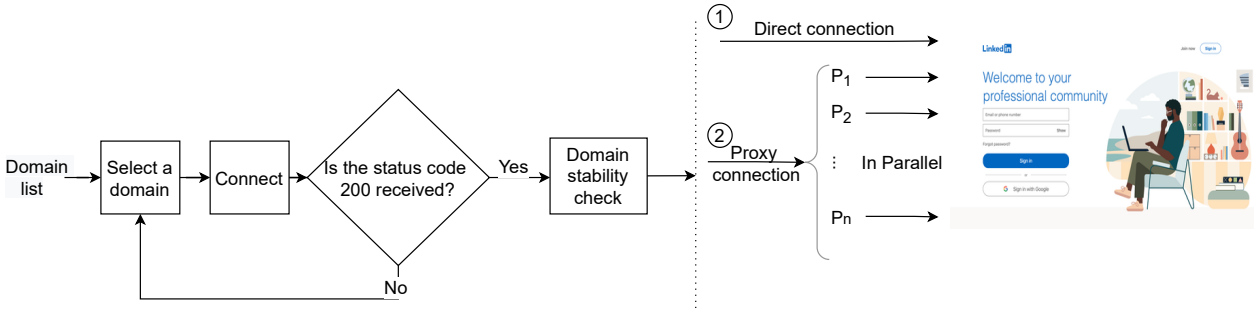


Figure 4.2: The overview of the data collection pipeline

Due to the highly volatile nature of some websites, in this work, we focus specifically on the ones that consistently provide the same content across multiple requests. Although this potentially limits the numbers of eligible for our study domains, this provides us a stable set to test our hypothesis. To confirm stability, we downloaded content of each website four times over a period of six weeks. If the content remains stable over this time, we deem this website to be stable and access it again directly and through all operational proxies in parallel. We ensure that all downloads through proxies are completed immediately after the direct access. If this is not possible (e.g., due to network connection), the content download for a given website is restarted.

For all operational domains, the module collects the following information:

a) Website content The corresponding homepage of each domain is downloaded associated with all of its content such as HTML, CSS and JavaScript files, images/video/audio content, fonts, unexpected random files loaded by browser such as CSV, XML, archives, and any other files.

The HTML file is the main source code of a web page containing images and other files and content shown on the web page as well as required libraries for the web page's functionality. JavaScript files contain JavaScript instructions executed in web pages to carry out functions such as validation of form fields. CSS files define the appearance of the web page. Subsequently, our module derives sitemap structure extracts and

downloads all hyperlinks present in loaded content.

b) Technology In addition to website content, we gather additional information about the technology employed by the website. This includes all client-side and server-side libraries, content management systems, statistical/analytical packages used by the website (e.g., ASP.Net, PHP, HTML5, Drupal, JQuery, Google Analytics, etc.). JQuery and HTML5 are samples of client-side libraries, and PHP and ASP.NET are server-side libraries. Drupal is a sample of content management systems alongside WordPress, the most popular CMS. To understand whether content manipulation depends on the vulnerability of employed by domain libraries, we separately label vulnerable software libraries using JavaScript library vulnerability scanner, DSJS. [36]

Technology group consists of 20 features needed to clearly differentiate between the various technological means employed by servers and application to communicate with clients and with other applications.

c) Network information For each domain, we also collect network characteristics at the time of our data collection. These network characteristics of the domain contain the ASN number, the prefix of the website's IP address, and RIR name, handle, register date, country of the domain. Besides domain, network attributes are collected for proxies as well. Proxy network features consist of ASN number, the prefix of the proxy's IP address, RIR name, the protocol of the proxy, and anonymity. Among these features, the anonymity of proxies shows how anonymous the connection via a proxy is and is demonstrated with three levels (High, Low, Transparent). Protocol of the proxy also indicates whether this proxy establishes a connection over HTTP/S protocol or both.

This group consists of 18 features needed to identify the data and traffic sources to allow for the detection of proxies and domains and for the differentiation between content distribution networks and static sources.

d) Web server information Besides technology, we also obtain information related to a web server hosting the corresponding domain during data collection, such as HTTP security headers, webserver applications (e.g., Apache, Nginx), and OS of the underlying web server with its version if possible. To further increase accuracy, some features have been engineered to detect specific versions of web servers, e.g., webserver metadata like "server" has been further characterized into "Nginx," "Apache," and "Microsoft_IIS" features. This group consists of 32 features that are relevant to the identification of web-related aspects, such as HTTP headers, security headers, communication protocols, and domain metadata.

A total of 92 unique features are engineered to ensure an adequate representation of features related to identification and prediction of content manipulation. The list of features within each category is given in Table 4.1.

Table 4.1: List of features

Type	Features
File features	magictype, mimetype, encoding, link_type, is_code, is_unique, ext, filesize_gp
Content snippet features	charnum, ch_avg_len, std_dev, added_changes, reduced_changes, num_file_chars, change_ratio, num_spaces, num_uppercase, num_lowercase, num_digit, num_enters, value_change, tag_change
Technology	MetaGenerator, WWW_Authenticate, PHP, HTML5, Drupal, Frame, SVN, JQuery, Google_Analytics, ShareThis, Modernizr, Open_Graph_Protocol, ASP_NET, Microsoft_HTTPAPI, bootstrap, jquery_migrate, jquery, ckeditor, mustache.js, ckeditor
Webserver	Cross_Origin_Embedder_Policy, Cross_Origin_Opener_Policy, Cross_Origin_Resource_Policy, Content_Security_Policy, X_Content_Type_Options, Content_Security_Policy_Report_Only, Expect_CT, Feature_Policy, Origin_Isolation, Strict_Transport_Security, Upgrade_Insecure_Requests, X_Download_Options, X_Frame_Options, X_Permitted_Cross_Domain_Policies, X_Powered_By, X_XSS_Protection, Public_Key_Pins, starting_domain, dom_protocol, Parked_Domain, Microsoft_IIS, Apache, nginx, server, server_ver, OS, PoweredBy, HTTPServer, Via_Proxy, Check_Point_SSL_Network_Extender, dom, init_status
Network	proxy,IP_dom, handle_dom, asn_dom, asn_name_dom, rir_dom, reg_date_dom, prefix_dom, cc_dom, domain_dom, isp_dom, proxy_type, anonymity, prefix, asn, country, rir, source

4.2 Content manipulation detection

Once data is collected, manipulations are detected and extracted following the three-step process:

Step 1: Parse files All collected files are parsed to make them digestible for the later stages. All corrupted files that can not be opened and files with file size zero are discarded. At this stage, the following file features are extracted to guide further analysis: "file magic number," and "mime type," which indicate the type of the file, giving some clues about what the file contains. The "extension" shows what type the file pretends to be (The term pretend is used, since a file extension can be chosen regardless to its real file/magic type.). Type of the file's "source URL link" indicates whether it is an internal link of the domain address or an external link. "Is_code" indicates whether a file is a source code or other type of files such as image, binary, etc. "Is_unique" shows whether the file is unique among the files received from this domain in a given connection or there is more than one file with the same file hash. The "encoding" and "size" of the file are also extracted.

In addition to extracting file features for the next steps, the hash of the file is calculated to clean the dataset from duplicated files. By comparing the hash of files downloaded in a single connection, duplicated files in the connection are removed to reduce complexity and extra computations.

Step 2: Manipulation detection All files collected through the proxy and without it are paired based on the source download link which includes the name of the file. The files are hashed and compared. When the hash of a file downloaded through proxy differs from the hash of the original file (we will refer to it as the *base file*), we consider this file modified. In addition, there are some other files downloaded through proxies that do not match with a base file, this group of files are extra files loaded only when the connection is made over proxy. These files are considered as injected files. The term *manipulation* implies both *file modification* and *file injection*.

Step 3: Extraction of modifications Content modifications applied to a file typically come in several forms: content injection, modification, and deletion. In this analysis, we consider all three types of manipulations. For each pair of files, altered content is extracted using Google's high-performance DMP library [4] that implements the Myers difference algorithm [29]. These extracted code/content snippets are saved for further analysis.

In the extraction process, only pairs of source code files are taken into account. Each pair of source code files are compared using the DMP library to extract the manipulations made by a proxy. The result of the DMP library can be one of the three different types of *deletion, addition, no change* that are explained in the Data chapter. If results show that a code snippet is returned as a deleted line and a similar code is returned as added code, the change is considered *modified*. However, if a code snippet is only returned as a deleted snippet, the change is *deletion*; similarly, the *addition* change is defined by a piece of code returned as an added snippet.

To enrich the further analysis, at this stage, we also calculate features characterizing extracted content snippets that include a summation of characters, length of each snippet added to the file, number of added snippets, number of removed code snippets, etc. (Table 4.1).

4.3 Characterizing modifications

Previous studies reported different types of content manipulations [39, 23]. We employ a clustering approach to investigate the types of changes present in our collected content. In order to categorize modifications, first, the modified code snippets are extracted from the manipulated file then the character-level 4-grams of code snippets are generated and their frequency is counted. Beside extracting 4-grams, we use content snippet features in clustering, consisting of numerical features about the code snippet.

[4]Diff-Match-Patch (DMP) https://opensource.google/projects/diff-match-patch

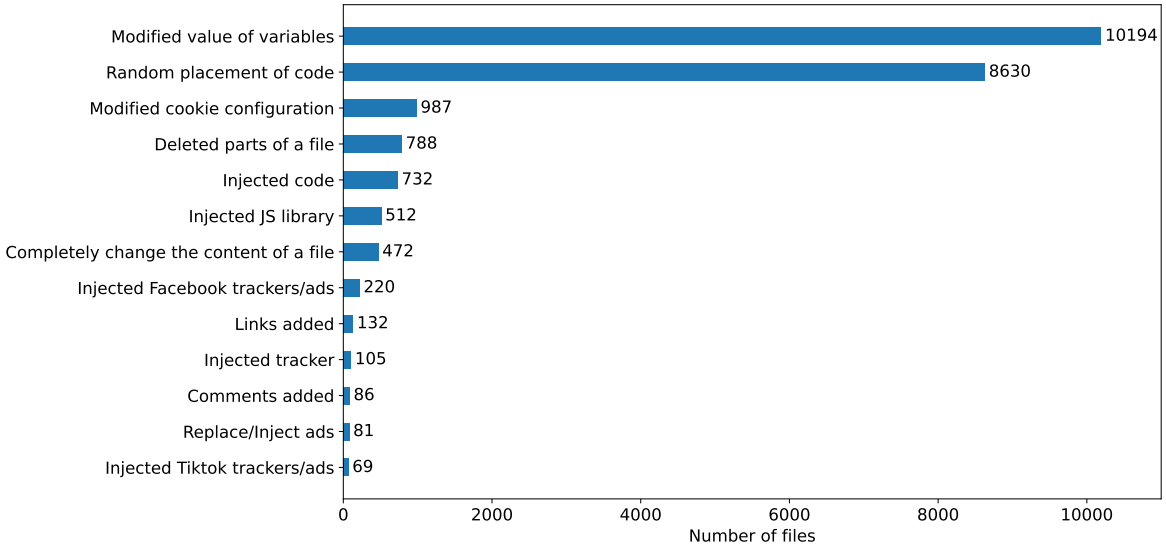


Figure 4.3: Classification of content modifications by types

N-grams is a sequence of n contiguous samples from a given text. It can be a sequence of n words, characters, phrases, lines of the given text. We leverage character-level 4-grams in our project to find all sequences of 4 characters, this will engage the content of code snippets in the next steps of our analysis. N-grams are used to extract features for machine learning models that are applied on text or any sequential data, such as natural language processing, identification of genre of a website [24], or in cyber security field it is being used in code authorship attribution [18] or malware detection [35].

We need to remove the unwanted files from our dataset, so that the result of the clustering is promising. By extracting 4-grams we can automatically remove the unwanted files using their specific 4-grams or combination of them. For example, code snippets that have returned 404 network error or internal server errors are considered as unwanted changes and are removed automatically.

To group code snippets into categories of modifications types, we have to identify the different modification types. We utilize clustering techniques over all generated features of code snippets. Before employing a clustering algorithm, feature engineering techniques are applied to the dataset.

Firstly, cells with null values of categorical columns are filled with 'unknown,' and numerical cells are filled with '0'. Afterwards, columns containing one value are removed and duplicated rows are removed. Then, columns containing categorical values are coded into numerical values.

Clustering is applied to the dataset that consists of snippets containing at least five characters in length (so that 4-grams can be calculated). In addition, to reduce the amount of noisy and irrelevant n-grams, we thus select the top 20 most frequent 4-grams for each file. The produced n-gram frequency of code snippets showed that most files have almost 20 n-grams seen at least two times in the snippet, and the rest of the n-grams are seen once.

Having top 20 frequent 4-grams per each code snippet as distinct features in the dataset makes it a high dimensional one. Applying a clustering algorithm to a high-dimensional dataset will not result in a desirable answer. Consequently, t-distributed stochastic neighbour embedding (t-SNE) algorithm [15] is chosen to support clustering. Yang et al. [48] utilizes t-SNE algorithm to visualize the malware dataset in order to find out the number of malware families, and then they apply clustering methods to cluster malwares into their families. Similarly, we visualize our dataset of code snippets and find out the number of modifications, then apply the k-means clustering method to the our dataset, configuring K-means algorithm with the number of clusters observed in the previous step.

T-SNE is an unsupervised, non-linear technique that is used in visualizing high-dimensional data and data dimension reduction. In other words, t-SNE can show how the data is arranged in the high-dimensional space. T-SNE algorithm consists of two main steps. In the first step a similarity measure between pairs of high-dimensional data in calculated. Then the same measure is calculated over pairs in low-dimensional space. Afterwards, the similarity measures are optimized with a cost function [15].

K-means is an unsupervised distance-based algorithm, in which the distance is calculated to assign a data point to a cluster. The number of clusters is pre-defined before running the algorithm and each cluster is associated with a centroid. The main goal of k-means is to minimize the sum of distances between the points and their corresponding cluster [21].

The results of clustering are examined manually to assign each cluster a modification type. The reason for manual examination is as followed.

1. Clustering algorithm only groups similar modifications into a cluster and does not assign a type to the cluster.

2. Some defined modification types are subsets of some other types due to mutual 4-grams, thus they can be in the same cluster.

3. Some clusters can be created based on the content of the web page rather than keywords showing the modification type.

Since observing a large dataset manually is not possible, the whole process of visualizing the dataset, clustering and manual observation is operated over chunks of data. The main goal of the process is to label the modified snippets, so the configuration of the t-SNE and k-means algorithm varies based on the chunk that the algorithms are applied to. The distribution of data points will be different in chunks, thus the perplexity parameter of the t-SNE algorithm will be set to various values in each chunk. In addition, the number of modification types in chunks will not necessarily be the same, so the k-means model will be configured with different values as the number of clusters each time.

As mentioned in previous paragraphs, clustering code snippets does not define the exact modification type and simply clusters the modifications. Hence, the corresponding code snippets of data points in the clusters are manually analyzed to define the applied modification on them. The defined type of modifications are:

- **Modified cookie configuration**: adding or removing a piece of code related to the configuration of cookie attributes.
- **Random placement of code**: moves the existing line(s) of code within a file, e.g., 2 lines in the JavaScript library can be swapped.
- **Modified value of variables**: modified the value of a variable, e.g., ID, is changed.
- **Injected tracker**: added URL links to tracking code or a website.
- **Injected JS library**: a script tag with a URL link to an additional JS library is added to the code.
- **Injected Tiktok trackers/ads**: links to Tiktok ads or events or various Tiktok tracker libraries are added to the file.
- **Injected Facebook trackers/ads**: similar to the injected Tiktok trackers/ads category, links to Facebook ads or events or Facebook tracker libraries are added to the code.
- **Replace/Inject ads**: the original advertisement present in the website content downloaded directly is replaced with another ad., or new ads are added to the code.
- **Injected code**: new chunks of code are added to the original file, however, their functionality is not obvious.
- **Deleted parts of a file**: a chunk of code or content is deleted.
- **Completely change the content of a file**: in such case, the entire content of the file is replaced, yet the original name is retained.
- **Links added**: new URL links that do not fit into either of the above tracker/ads categories are added to the code.
- **Comments added**: some comments are added to the code.

Although some of these modifications can be generalized into more broad categories, we preferred fine granular categorization to distinguish between different modifications. Figure 4.3 shows the distribution of these types of modifications in our collected data. It is interesting to note that the majority of changes include modification of variable value (e.g., converting session cookie to path prefix) and random placement of code, while deletion of file portion is equally present as injection of some code.

4.4 Prediction

When the modification types are extracted, each file is tagged with its corresponding modification type. Then the file dataset is ready for modification prediction. The prediction is applied on three distinct datasets, that

differ in the type of files they contain. One dataset only consists of JavaScript collected files, the other dataset contains only HTML files, and the last dataset covers the combination of HTML and JavaScript files.

The dataset used in this experiment consists of 3 groups of features related to files, domains, proxies and all features are categorical. Domain-related features include network, web server, and technology information collected in the data collection section. The proxy part includes network features, and the dataset contains various characteristics of files.

The dataset requires preparation for better results before applying any machine learning (ML) model. Thus, feature engineering methods are applied to the dataset. First, the duplicated rows are removed, and columns with only one value were deleted. Then null valued cells were filled with *unknown* and the categorical values were encoded to numerical values. Finally, the dataset is standardized to prevent any bias.

For our prediction analysis, we explored the performance of six classification algorithms: Multi-layer Perceptron, Decision Trees, Linear Discriminant Analysis, Random Forest, Logistic Regression, and Support Vector Machine analysis. The selected machine learning algorithms were chosen based on previous studies [44, 22, 41, 17] that leveraged machine learning techniques to detect malicious URLs by collecting website features of the URLs.

Multi-layer Perceptron (MLP) [26] algorithm is a version of the neural network algorithm that is a modelling inspired by brain cells. Each computational element in MLP networks is called neuron and includes a linear matrix operation between the weight parameter and the input, then a non-linear operation is applied to the result of the previous calculation. Multiple neurons create a layer, and a network consists of layers of neurons.

The combination of linear and non-linear operations makes the network suitable for non-linear problems as well as linear problems. MLP is a supervised model used for both classification and regression problems which makes it suitable for our classification problem. The MLP network leveraged in our experiment is implemented via the scikit-learn library which provides a simple function to create an MLP network with our own specific size.

Decision Tree (DT) [5] is a non-parametric algorithm that tries to predict a value based on some decision rules derived from data features. Decision trees are simple supervised machine learning algorithms that have a low timing cost and are used for regression and classification problems. The leaned rules can be modelled to a tree in which internal nodes correspond to attributes and leaf nodes correspond to class labels. The model builds such a tree by selecting the attribute that best splits the training examples into their proper classes.

Discriminant Analysis [46] algorithm develops discriminant functions that are linear combinations of independent variables that will discriminate between the categories of the dependent variable. It enables us to examine whether significant differences exist among the groups regarding the predictor variables. Linear discriminant analysis (LDA) is easy to compute based on their closed-form solutions, and they provide multi-class classification. LDAs have easy configuration since there are no hyperparameters to configure and they

are known to work well in practice. As a result, this supervised model is chosen for our analysis.

Random forest (RF) [16] classifier is an ensemble of multitude decision trees, and in prediction time, it chooses the class that is predicted by most of the trees in the forest. The decision trees in the forest are created in a way to have the least correlation. Thus, the prediction result of trees can differ and the crowd decides the final prediction. Random forest's advantage over decision tree is the point that the result of a wrong tree can be neglected by the correct decision of many other trees.

Logistic Regression (LR) [11] is similar to linear regression algorithm with the main difference in which the target contains categorical variables. Logistic regression is a statistical and linear model that calculates the probability of key modulus x belonging to a target class. We use multinomial logistic regression classification as we have more than two classes in our dataset.

Support Vector Machine (SVM) [7] algorithm tries to find a line or hyperplane between classes of data to separate them. The dimension of the separator is dependent on the number of features. This supervised method is used for classification, regression, and outlier detection. In general, SVM models are efficient in high-dimensional spaces and they are effective for datasets with a larger number of features compared to the number of samples. The SVM algorithm can be configured with versatile kernels and form a linear or non-linear model. As the accuracy of the linear SVM on the test set is close to the non-linear SVM, such a method is more efficient and much faster in high-dimensional data applications. Linear SVM is a collection of problem formulations, solvers, and optimization algorithms that can be utilized to find an efficient solution.

Table 4.2: The parameters of the classification algorithms

Algorithm	**Hyperparameters**	**Classifier**
Multi-layer Perceptron	hidden_layer_sizes=(100), max_iter=10000, learning_rate="adaptive", solver="adam", alpha=0.001, random_state=42	Nonlinear
Decision Tree	max_depth=50	Nonlinear
Linear Discriminant Analysis	solver="svd", shrinkage=None, tol=1.0e-4	Linear
Random Forest	n_estimators=100, criterion="entropy", bootstrap=True, min_samples_split=2, min_samples_leaf = 1, oob_score = True, max_features="auto", min_impurity_decrease=0.0, max_depth=50	Nonlinear
Logistic Regression	penalty="l2", max_iter=100000, dual=False, dual=False, fit_intercept=True, intercept_scaling=1.0, random_state=42, solver="lbfgs", multi_class="multinomial"	Linear
Support Vector Machine	kernel="linear", C=0.025	Linear

Some machine learning models have specific configuration settings. Multi-layer perceptron is set to *one*

hidden layer with 100 neurons, with *adam* optimization method and an *adaptive* learning rate. The decision tree is set to have at most *50* nodes in depth to prevent the model from overfitting since we have around 80 features in the datasets. In LDA model is configured to have *svd* solver since it is suitable for a dataset with a large number of features. The random forest model is set to have the same maximum depth as decision tree,*50*, and it has *100* trees. The logistic regression model is set to use *lbfgs* optimization algorithm with *L2* regularization method. Table 4.2 has gathered the detail of all model configurations.

4.5 Summary

In this chapter, the approach utilized in this study was explained. First, we collected the technical features of proxies and websites and the content of webpages through proxies. Then, the collected files were parsed and processed to find the manipulated files. The modified files were processed in the next step to extract the modified code snippets. We categorized the modified code snippets based on their type of modification, which resulted in 13 types of modification. We leveraged six machine learning algorithms to predict the type of modification applied to the retrieved files of webpages through proxies.

5 Data Collection and Its Analysis

As shown in our analysis pipeline in Figure 4.1, we have several steps and we created a setup for each one of steps. We used some tools and Python for our setup, explained below. In addition, statistics of the collected dataset is given in this section.

5.1 Experimental setup

5.1.1 Data collection setup

Our data collection setup consists of three parts: connecting to the domain, loading the domain and its content, downloading the loaded content of the domain.

The connecting setup leverages WhatWeb tool v0.5.5[1]. WhatWeb helps us recognize the website's web server features, e.g., the OS of the web server and technologies a website uses, such as statistic/analytics packages and JavaScript libraries. We can get the version number of technologies. We use WhatWeb to make a connection to the selected domain; then, by receiving the status code, we find out about the availability of the domain. If the status code is 200, the domain is available for scraping. Besides, while we try to connect to the domain using WhatWeb, we collect various information that we can retrieve from WhatWeb about the domain.

If the website is available online, we load its content using our setup, which is implemented using the Python language (v 3.8.5) and Selenium WebDriver tool. Selenium WebDriver is a web framework that controls a browser and loads a website or web application test. [14] In our setup, Selenium WebDriver loads the domain's content over a chrome browser (v 91.0.4472.77).

When the content is loaded, their source URL links are grabbed from the web driver and given to the requests-futures (v 0.9.1) Python library to download content from captured source URL links. [27] The website technology is also identified by the WhatWeb tool.

As shown in Figure 4.2, direct and proxy connections were done in parallel. This paralleling is done using Linux GNU parallel (stable version released on April 2021) software.

[1] https://github.com/urbanadventurer/WhatWeb

5.1.2 Manipulation detection setup

The manipulation detection process starts with parsing downloaded files for which some features are calculated, such as the size of the file and SHA1 hash of the file. As explained in the approach chapter, these features are used to find manipulated files. The parser is implemented in Python language (v 3.8.5), and the SHA1 hash is calculated using the *hashlib* Python library.

When manipulated files are found, the modified sections of each manipulated file are extracted for characterization. This procedure uses the Google diff-match-patch (DMP) tool, which compares the same directly downloaded file with over proxy downloaded version, and finds the differences.

The result returned by the Google DMP tool is a list of tuples, wherein each tuple first element is one of these {-1, 0, 1} values representing an operation and the second element of the tuple is the content snippet of the compared file that the mentioned operation is done over it. *-1* represents deletion, i.e., if a tuple starts with -1, the snippet shown as the second element of the tuple has been removed from the file, *1* represents an addition, i.e., if a tuple starts with 1, the snippet in the second element of the tuple has been added to the file, *0* represents no change, i.e., if a tuple starts with 0, the snippet in the second element of the tuple has remained the same with any changes.

5.1.3 Characterizing manipulation setup

In order to characterize and label modifications, we need to parse and extract some features from modification code snippets. Thus, we calculated character level 4-grams using n-gram (v 4.0.3) Python library. Then code snippets are clustered using these features. The clustering module is implemented using the scikit-learn library (v 0.23.2).

5.1.4 Prediction setup

Our prediction setup is implemented using the Python language (v 3.8.5) with the scikit-learn library (v 0.23.2). Scikit-learn is a machine learning library for Python that allows programmers to use various machine learning algorithms with different attributes. We have defined multiple machine learning models using this library to predict the target in several experiments on the dataset. A summary of the classification algorithms' parameters used in the prediction module is given in Table 4.2. A 5-fold cross-validation was employed to measure the accuracy of all machine learning models.

Table 5.1: The collected data

	Total	**Responsive**				
		Number	**Countries**	**ASNs**	**Prefixes**	**IPs**
Domains	1200	1028	48	263	563	985
Proxies	3088	1293	73	720	1009	-

5.2 Collected proxies and domains

5.2.1 Proxies

Proxies are collected from github lists clarktem[2] and sunny9577[3], in addition to proxyscan.io[4] API, and augmented the lists with additional results collected with ProxyBroker[10].

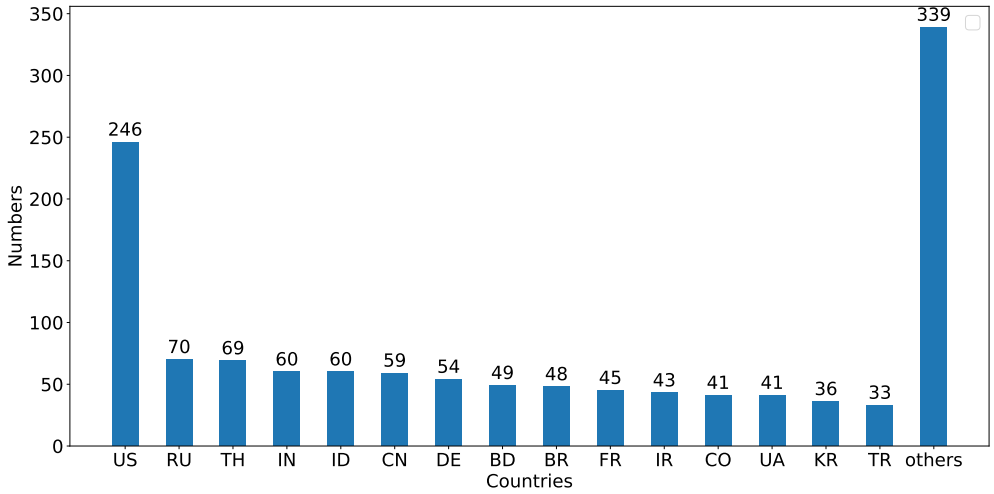
We have identified 3088 proxies for our experiments, among which approximately 41% were alive and responsive. The majority of the operational proxies are located in the US (254), while the rest are distributed across 72 countries (Figure 5.1 (a)). The most common ASNs among proxies are ASN numbers *14061, 24940* which contain 39 and 35 proxies, respectively. 82% (1042) of our proxies support both HTTP and HTTPS protocols, 125 proxies support only HTTP, and 126 support only HTTPS protocol.

5.2.2 Domains

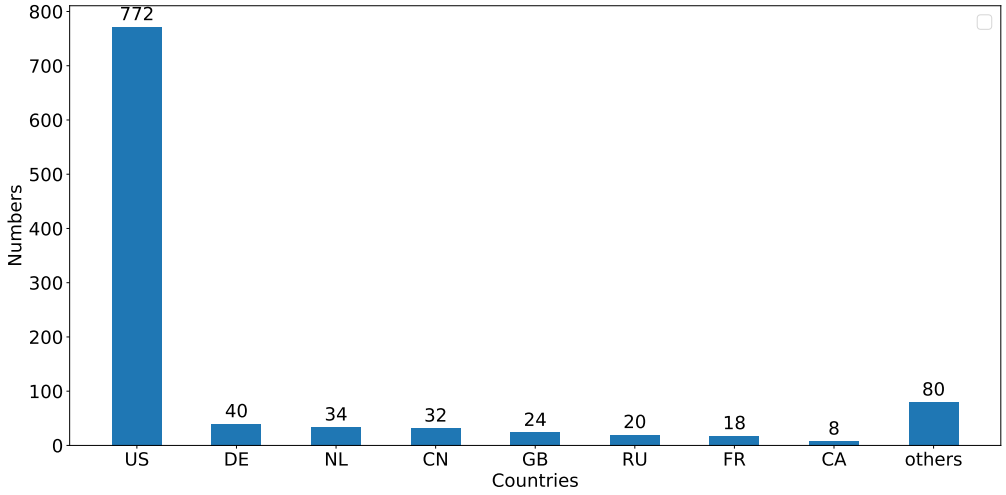
A selection of domain names has been extracted from a list of the top 1 million domains called Majestic Million list [5]. A set of 1200 domains has been selected randomly from the 20,000 first domains of the lists for our experiments.

Out of 1200, 1028 (85%) were operational and available for content download. The operational domains originate from 985 unique IP addresses from 563 prefixes and represent 48 countries worldwide. In addition, domains are from 263 distinct ASNs, and ASN with number *13335* is the most frequent ASN seen among domains, from where 275 domains are. There are only 11 ASNs that include more than ten domains, and the 252 other ASNs have less than 11 domains and mostly have only one domain. The distribution of domains across countries is given in Figure 5.1 (b). The majority of the domains reside in the US (772), besides seven other countries containing more than eight domains each and 40 remaining countries have less than eight domains. The summary of the domains and proxies used in our analysis are provided in Table 5.1.

[2]https://raw.githubusercontent.com/clarketm/proxy-list/master/proxy-list.txt
[3]https://sunny9577.github.io/proxy-scraper/proxies.json
[4]https://www.proxyscan.io/api/proxy?type=http,httpslimit=600
[5]https://majestic.com/reports/majestic-million



(a) Distribution of proxies



(b) Distribution of domains

Figure 5.1: Distribution of proxies and domains over countries.

5.2.3 Collected Files of Dataset

As a result of running our experiment setup over 1028 domains and 1293 working proxies, we have collected 4,502,397 files in total. These files have been analyzed in terms of the number of files, their types, and their size.

The details of the number of collected files are given in Table 6.3 which shows the distribution of files by their type. The majority of the collected files were compressed files by 37.6%, and among the rest, we have received images and JavaScript files the most with 26.6% and 22.9%, respectively. By decompressing the compressed files, 1,690,966 were added to image, application, HTML, JS (JavaScript), text files, most of which are JS files.

The general text and application groups shown in Table 6.3 include various specific types. Text category contains *.XML, .CSS, .asm* and other type of files where .asm files that contain assembly code are actual .css files that are mistakenly recognized as .asm files, .xml files contain site map, while .css files contain style information that has effect on the appearance of the website. Application group contains *JSON, CSV, OCTET-STREAM* files among which the octet-stream files are documents that are open in spreadsheet or word processor applications.

The analysis of types of files is executed based on their mime type and magic number. Our downloaded files are of 54 distinct magic numbers and 32 distinct mime types. Since the focus is on the text files in the analysis, the types of the text files are also inspected. Text files consist of 25 different magic types and 13 mime types. Among all manipulated files and manipulated text files, variability is of mime type and magic number is less compared to collected files and injected files have a similar trend, the details are shown in Table 5.2. While it was expected to receive files of several types such as HTML, JavaScript, and media types, the received files are of a large variety of mime types with unexpected types such as CSV assembly files.

Table 5.2: The number of types of files.

	Downloaded files		**Manipulated files**		**Injected files**	
	All files	**Text files**	**All files**	**Text files**	**All files**	**Text files**
Magic number	54	25	43	17	35	11
Mime type	32	13	24	7	22	6

In addition, the size of the files has been inspected by dividing files based on their mime type and calculating the average size of each mime type in the defined subsets. Among all subsets of files, the largest files with average sizes of 7 Mb, 4.8 Mb, and 4.1 Mb from all downloaded, manipulated and injected files respectively belong to *videos/mp4* mime type.

5.3 Summary

In this chapter, the technical tools utilized in this study are explained. We explained the programming language used for scripting and related tools and libraries for our experiment. In addition, the source of collected proxy and domains and distribution of them over countries are demonstrated. Also, an analysis of the collected files regarding size, types, and the number of them is given to show the variability of the collected files.

6 Experimental results

In this chapter, the conducted experiments and their results will be explained. We executed two sets of machine learning experiments to predict content modification types defined in the previous chapter and predict content injections. Moreover, the relation between file dataset samples and their features is analyzed, and interesting results are achieved.

6.1 Collected Data Analysis

We have run a setup to collect files from web pages directly and through open web proxies to provide a proper dataset for our research. Hence, the outcome of this setup and the collected files is explained first.

The website requests routed through a proxy are generally expected to return identical results without it. This statement, however, is not always true. In our experiment, we observe that the content of 70-90% of domains received from proxies was different from the base content, content received directly from the website, in most cases(Table 6.1). Similarly, we received different content from almost all domains fetched through 58 proxies. The manipulated content received through proxies includes manipulations made by proxies and network and web server errors.

The analysis of manipulating proxies and their network shows that modifying proxies are distributed over various ASNs. We have grouped the proxies based on the number of domains they manipulate, and we show the list of the eight most frequent ASNs that proxies of each group belong to in Table 6.2. A total of 48 ASNs are given in the Table, among which nine have appeared in more than one category, and we have 32 unique ASNs as the most frequent ones among proxies. By further observation of the nine ASNs repeated

Table 6.1: Content manipulation by proxies

Number of proxies	Manipulated content
32	less than 10% of domains
92	10% - 30% of domains
31	30% - 50% of domains
226	50% - 70% of domains
831	70% - 90% of domains
58	90% - < 100% of domains
None	100% of domains

in multiple categories, we found that four are allocated by ARIN (Canada, United States, some Caribbean nations), three are allocated by APNIC (Asia-Pacific region), and two are allocated by RIPE (Europe, the Middle East and parts of Central Asia).

Table 6.2: Top ASNs of content manipulating proxies

Top ASNs	Proxies Manipulating content
61317, 45820, 58923, 135377, 45102, 132203, 16509, 8075	less than 10% of domains
24940, 23969, 61317, 45102, 14061, 131207, 3209, 8075	10% - 30% of domains
16509, 23969, 137311, 23383, 8308, 9341, 4766, 7018	30% - 50% of domains
14061, 14618, 24940, 8075, 31898, 17451, 262186, 16276	50% - 70% of domains
4766, 14061, 16276, 24940, 8075, 7713, 12876, 8400	70% - 90% of domains
31898, 45629, 133054, 55492, 45102, 24940, 15808, 37963	90% - < 100% of domains

We have collected more than four million files through proxies that form our dataset. Manipulated files share 13.8% (857,153) of the collected contents, and 29.7% (254,915) of manipulated files are identified as injected with the remaining of the manipulated files detected as modified files (Table 6.3).

The distribution of collected, manipulated, and injected content by various file types is given in Table 6.3. While the majority of the files fetched through proxies constitute compressed files (37.6%), image files (26.6%), and JavaScript codes (22.9%), the vast majority of changes were done to HTML and text files (67.4% and 57.2% respectively). Injections form a significant part of the manipulations of the font, audio, image, and JavaScript files.

In the previous paragraph, we discussed which files have the most manipulations compared to the overall number of files in the category. A significant portion of manipulated files is compressed HTML, JavaScript, and text files. Similarly, injected files mainly comprise JavaScript, compressed, image, and HTML files.

Injection analysis: We observe that 29.4% of files fetched through proxies are injected (Table 6.3). Surprisingly, video and audio files were injected more often than expected, with almost 40% of all video files (644) and 100% of audio files (22). Thus, injected video and audio files were further observed in order to find any embedded or malicious content. All injected media files were analyzed with Virus Total and ClamAV, but none was malicious. In addition, these files were analyzed to find embedded content, while none of the media files contained any suspicious embedded content. However, the further manual analysis indicated that injected ads delivered all injected audio and video content.

Malicious content modifications: We scanned all manipulated files for potential signs of malicious content. 403 files were detected by AVG and Avast anti-virus software as malicious. These detected files were fetched through 20 proxies. Note that we employed other anti-virus solutions such as ClamAV with no success primarily due to obfuscated nature of embedded scripts. We manually analyzed these files and

determined that they all contained ten unique Javascript (JS) scripts related to crypto mining behaviour.

Table 6.3: The number of collected and manipulated files in detail.

	Downloaded files*			Manipulated filesˆ			Injected filesˆ		
Type of file	**Num. of files**	**Num. of decom-pressed files**	**Total Num. of files**	**Num. of files**	**Num. of decom-pressed files**	**Total Num. of files**	**Num. of files**	**Num. of decom-pressed files**	**Total Num. of files**
Text (e.g., .asm, .xml)	171,994 (3.8%)	432,023 (25.5%)	604,017 (9.8%)	98,321 (57.2%)	56,141 (13.0%)	154,462 (25.6%)	4,599 (4.7%)	14,786 (26.3%)	19,385 (12.6%)
HTML	164,177 (3.6%)	141,069 (8.3%)	305,246 (4.9%)	110,714 (67.4%)	72,904 (51.7%)	183,618 (60.2%)	12,534 (11.3%)	14,599 (20.0%)	27,133 (14.8%)
JS	1,030,938 (22.9%)	874,777 (51.7%)	1,905,715 (30.8%)	102,768 (10%)	77,385 (8.8%)	180,153 (9.5%)	42,027 (40.9%)	32,325 (41.8%)	74,352 (41.3%)
Application (e.g., .json, .csv)	235,670 (5.2%)	44,969 (2.7%)	280,639 (4.5%)	19,647 (8.3%)	9,448 (21.0%)	29,095 (10.4%)	8,274 (42.1%)	2,243 (23.7%)	10,517 (36.1%)
Image	1,198,785 (26.6%)	192,915 (11.4%)	1,391,700 (22.5%)	74,169 (6.2%)	8,590 (4.5%)	82,759 (5.9%)	48,709 (65.7%)	4,647 (54.1%)	53,356 (64.5%)
Compressed	1,690,966 (37.6%)	-	1,690,966 (27.3%)	224,773 (13.3%)	-	224,773 (13.3%)	68,892 (30.6%)	-	68,892 (30.6%)
Font	4,939 (0.1%)	5,213 (0.3%)	10,152 (0.2%)	302 (6.1%)	305 (5.9%)	607 (6.0%)	302 (100%)	292 (95.7%)	594 (97.9%)
Video	4,819 (0.1%)	0	4,819 (0.1%)	1,664 (34.5%)	0	1,664 (34.5%)	664 (39.9%)	0	664 (39.9%)
Audio	109 (0.002%)	0	109 (0.001%)	22 (20.2%)	0	22 (20.2%)	22 (100%)	0	22 (100%)
Total	4,502,397	1,690,966	6,193,363	632,380 (14.0%)	224,773 (13.3%)	857,153 (13.8%)	186,023 (29.4%)	68,892 (30.6%)	254,915 (29.7%)

* : Percentages calculated in the columns of this section are based on the Total value on their column.

ˆ : Percentages calculated in the columns of these sections are based on the values of the same column in the previous section of the table.

Besides the text category, the vast majority of manipulations are performed on HTML and JS files. Hence, we focused on HTML and JS files as they are the primary source of content and code information for a domain. The corrupted JS and HTML files are discarded from the analysis. In addition, the extracted code snippets with a size of ≤ 60 characters are discarded. 60 character length of modification is our assumption for finding a meaningful and effective line of code in HTML and JavaScript, while less than this size is considered to be only an ID defined in the code.

Our observation shows that the number of manipulations (size of extracted code snippets) can vary from as small as 60 characters to 2,472,545 characters in JavaScript files and 3,322,605 characters in HTML files (Table 6.4).

Table 6.4: The size of extracted content snippets (# of characters)

Type	Average	Minimum	Maximum
JS	9,607.22	60	2,472,545
HTML	15,611.10	60	3,322,605

6.2 Prediction Analysis

Our findings show proxies' diversity of content manipulations (Figure 4.3). The close inspection revealed that proxies engage in these content changes seemingly randomly.

To understand the rationale for adding particular types of changes to domains, we conducted several experiments to address the following questions:

- What does determine the types of content modifications introduced by proxies?
- Can we predict file injections?
- Can we predict the source of injections?

What does determine the types of content modifications introduced by proxies? In order to investigate the content manipulation behaviour of proxies, we leverage all contextual information gathered during the data collection step, i.e., website technology, network information, webserver information and file features, with a target being the label indicating content modification type. Since these labels were derived through preliminary clustering, the feature extracted from manipulated code snippets is discarded.

The content modification prediction experiment was executed on three different sets: JavaScript only, HTML only, and the mixture of JavaScript and HTML files. Our results show that we can reasonably accurately identify the type of modifications that will be introduced by a proxy into HTML or JavaScript files.

Most machine learning models have resulted in an accuracy of at least 80% and more than 90% in three models in predicting the modifications applied on JavaScript files. The experiment over HTML files received pretty high accuracy with at most 87% and more than 76% in most cases. The details are given in Table 6.5. Furthermore, the accuracy of our prediction over the mix of HTML and JS datasets is more than 79% in most machine learning models, with the highest accuracy of 89.9% using the random forest model.

These results support our conjecture that *what types of modifications will be introduced in website content depends solely on proxy and domain, network, and web server characteristics and can be determined without the need to fetch data through a proxy.* Experiments have further confirmed assumptions with a more varied data set comprised of HTML and JS files.

The next step is to analyze feature impacts in our experiments to find the more compelling features in predicting the modification types. We decided to utilize information gain (IG) to identify the importance of

Table 6.5: Classification Accuracy of experiments using all features

	Algorithms					
Experiments	**LR**	**LDA**	**SVM**	**RF**	**DT**	**MLP**
Predicting type of modification based on JS and HTML	79.2%	70.1%	79.1%	**89.9%**	88.4%	88.7%
Predicting type of modification based on JS	83.8%	72.0%	80.5%	**92.0%**	90.2%	90.0%
Predicting type of modification based on HTML	76.8%	72.7%	77.6%	**87.0%**	85.7%	86.8%
Predicting presence of injected file	60.7%	80.0%	61.6%	**83.9%**	61.6%	83.5%
Predicting presence of injected file and its type	56.0%	41.9%	61.8%	100%	100%	**99.0%**
Predicting presence of injected file and its source	48.8%	48.8%	48.5%	100%	100%	**98.2%**

each feature. Tables 6.8 show the top selected features with Information gain (IG) $IG > 0.01$.

We can observe that the number of features selected as significant across experiments related to JS and HTML files is comparable: 29 features for prediction based on HTML and JS together and 31 features for JS and HTML when individually tested. Interestingly, file features seem to be more effective in predicting content modification based on JS files (6 out of 8 features are present) than based on HTML files (2 out of 8 features are considered essential).

Yet, the size of an HTML or a corresponding JS file is the top-ranked feature for all content type prediction experiments. We can hypothesize that the nature of content change mainly dictates this, i.e., smaller files have less content for modification. Indeed, in our data, larger files have more modified snippets.

While categories of features have different importance/ranking, they have a similar presence in all experiments. In all tests, Web server-related and Network-related features constitute the majority of features, i.e., they account for 30% to 42% and 35% to 49% of the total features, respectively. Technology-related features account for 10% to 13% of the total features.

The experiment was repeated by leveraging the features with $IG > 0.01$, which resulted in the same result as running all the features. In addition, we repeat our experiments excluding all proxy-related features, and the results are presented in Table 6.6. The accuracy drops significantly, indicating the critical role that proxy features play in the prediction process.

A close manual review of our prediction analysis results revealed several general observations. *First*, content snippets that were injected into files, such as injection of JS library, code, trackers, can be seen both on internal and external files. *Second*, the vulnerability of libraries employed by the domain does not play any role in the possibility of content manipulations by open proxies. *Third*, a group of modifications seen on

Table 6.6: Classification accuracy excluding proxy and its features from dataset

	Algorithms					
Experiments	**LR**	**LDA**	**SVM**	**RF**	**DT**	**MLP**
Predicting type of modification based on JS and HTML	46.6%	6.7%	**55.6%**	52.2%	43.1%	45.3%
Predicting type of modification based on JS	43.6%	14.2%	**57.6%**	49.3%	46.0%	48.2%
Predicting type of modification based on HTML	43.3%	36.7%	43.3%	**46.7%**	30.0%	33.0%
Predicting presence of injected file	44.4%	47.2%	**47.2%**	41.7%	33.3%	44.4%
Predicting presence of injected file and its type	37.8%	13.6%	**54.7%**	47.5%	33.3%	40.0%
Predicting presence of injected file and its source	40.3%	33.9%	**47.5%**	42.5%	45.0%	42.8%

domains have specific HTTP headers that are not configured correctly. For example, injection of JS libraries, links, advertisements or trackers is found on domains that do not have 'Content Security Report Only' among their HTTP headers or have not configured it.

In Table 6.7, we have summarized how the lack of these headers can impact the abilities of proxies to manipulate or inject traffic. When such headers are set to *None* or are not set at all, proxies are able to manipulate traffic freely, and client-side browsers have no technical means to prevent or mitigate such actions.

Most of the headers we have observed as not being set or properly configured are in the category of security headers that cover the loading, modification, injection and removal of content in transit or on the client-side. Most of these headers are directly connected with proxies' core functionalities and operational purposes.

As explained in background section, the *except CT* headers is used to prevent usage of wrongly issued certificates by a website. The lack of such header makes it possible for proxies to replace original security parameters and certificates with elements of their choosing, resulting in encrypted connections not being considered private and secure.

Furthermore, the header "Cross-Origin Resource Policy" (CORP) helps websites and applications to protect their content against requests from other origins. Misconfiguration of this header can lead to stealing the domain content and resources by requests sent from other websites.

When a website, for whatever reason, does not make use of security headers available in major web browsers, it exposes itself to the loading, modification, injection and removal of its content.

Can we predict file injections? We observe that 29.4% of files fetched through proxies are injected (Table 6.3). We try to predict such injections in this study. Hence, we create a dataset of all domains and proxies with their related features, and we predict whether each domain sample will have injected files fetched through a specific proxy. To investigate the feasibility of predicting file injections, we performed several machine learning models where results are shown in Table 6.5.

The results also confirm our second hypothesis, i.e., we can fairly accurately (83.9% with RF) identify if additional files may be injected by proxy, regardless of file type. Besides the random forest model, we have reached 83.5% and 80% on the multi-layer perceptron and linear discriminant analysis models.

Then we analyze the impact of features in this experiment to find the effective features in the results. Thus, information gain (IG) of the Random Forest model is calculated, and features with $IG > 0.01$ are listed in 6.9. We can observe that proxy-related features play an important role in this prediction: the four most significant features are proxy characteristics. Overall we have 23 significant features, among which network and webserver features cover the majority of features as well as proxy networks.

We further ran the same experiment on a dataset without any proxy-related features to assess the effect of proxy in injection prediction. Results show that proxies do have a vital role in injection to a domain, and details are given in Table 6.6.

Can we predict the source or type of injection? The last set of our experiments focused on predicting the source and type of additional files injected by proxy. None of the file-related features were included in the analysis in these experiments, and the target label was the type or source of files. In cases when a domain fetched through a proxy resulted in multiple files, the target label was cumulative of file types.

The results shown in Table 6.5 confirm our third hypothesis, i.e., *we can fairly accurately identify the source and type of files that a proxy may inject*, and this can also be done by using just the source website and the candidate proxy information. We achieved 98.2% accuracy (MLP algorithm), focusing not only on the probability of injection but also on the source of the possible injection and 99% accuracy (MLP) in further predicting the type of additional file that might be injected by proxy.

The importance of features was also calculated in these experiments and is shown in Table 6.9. The results show that domain-related characteristics are the most significant in identifying the type and source of file injection. To ensure the role of proxies in these experiments, the prediction is repeated with excluding proxy features, and as shown in Table 6.6, the proxy's role is not negligible.

6.3 Summary

This chapter showed how accurate our setup is in predicting content modification types and injection prediction with different additional information. We also showed that injection prediction does not require fetching content from websites through proxy. Moreover, the importance of HTTP header configuration role in content manipulation was discussed.

Table 6.7: The types of modification seen on domains with not configured HTTP headers

HTTP headers	Cross origin resource policy	Content security report only	Except CT	Feature policy	X permitted cross domain policies
Modification types	Inject code	Cookie configuration	Complete change of the content	Cookie configuration	Inject code
	Cookie configuration	Inject JS library	Inject ads	Inject JS library	Cookie configuration
	Inject JS library	Add links	-	Add links	Add links
	Add links	Complete change of the content	-	Complete change of the content	Complete change of the content
	Complete change of the content	Inject ads	-	Inject ads	Delete part of a file
	Inject ads	Delete part of a file	-	Delete part of a file	Inject tracker
	Delete part of a file	Inject tracker	-	Inject tracker	Inject Tiktok
	Inject tracker	Inject Tiktok	-	Inject Tiktok	Inject Facebook
	Inject Tiktok	-	-	Inject Facebook	-
	Inject Facebook	-	-	-	-

Table 6.8: The features (IG> 0.01) selected for modification type prediction experiments given Table 6.5 (shown in order of their ranking from most to less significant)

Prediction based on JS and HTML		**Prediction based on JS**		**Prediction based on HTML**	
Features	**Types**	**Features**	**Types**	**Features**	**Types**
filesize_gp	File	filesize_gp	File	filesize_gp	File
ext	File	magictype	File	dom	Webserver
magictype	File	dom	Webserver	IP_dom	Network
dom	Webserver	prefix_dom	Network	prefix_dom	Network
mimetype	File	IP_dom	Network	encoding	File
IP_dom	Network	mimetype	File	proxy	Proxy Network
prefix_dom	Network	asn_dom	Network	asn_dom	Network
asn_dom	Network	handle_dom	Network	Proxy_prefix	Proxy Network
reg_date_dom	Network	X_Content_Type_Options	Webserver	handle_dom	Network
handle_dom	Network	reg_date_dom	Network	reg_date_dom	Network
cc_dom	Network	Strict_Transport_Security	Webserver	Proxy_asn	Proxy Network
Strict_Transport_Security	Webserver	asn_name_dom	Network	asn_name_dom	Network
asn_name_dom	Network	JQuery	Technology	JQuery	Technology
proxy	Proxy Network	link_type	File	starting_domain	Webserver
link_type	File	starting_domain	Webserver	Strict_Transport_Security	Webserver
jquery	Technology	bootstrap	Technology	MetaGenerator	Technology
Proxy_prefix	Proxy Network	Frame	Technology	Proxy_country	Proxy Network
Proxy_asn	Proxy Network	X_Frame_Options	Webserver	bootstrap	Technology
encoding	File	ext	File	jquery	Technology
starting_domain	Webserver	cc_dom	Network	X_Frame_Options	Webserver
X_Frame_Options	Webserver	Content_Security_Policy	Webserver	anonymity	Proxy Network
X_XSS_Protection	Webserver	proxy	Network	PoweredBy	Webserver
Proxy_country	Proxy Network	Proxy_prefix	Proxy Network	cc_dom	Webserver
PoweredBy	Webserver	X_Powered_By	Webserver	X_XSS_Protection	Webserver
Content_Security_Policy	Webserver	PoweredBy	Webserver	nginx	Webserver
server	Webserver	Proxy_ASN	Proxy Network	Content_Security_Policy	Webserver
bootstrap	Technology	Proxy_country	Proxy Network	Proxy_rir	Proxy Network
X_Powered_By	Webserver	server	Webserver	server	Webserver
Frame	Technology	X_XSS_Protection	Webserver	X_Powered_By	Webserver
-		encoding	File	Apache	Webserver
-		HTTPServer	Webserver	MetaGenerator	Technology

Table 6.9: The features selected for prediction of injected files experiments given Table 6.5 (shown in order of their ranking from most to less significant)

Predicting presence of extra file		**Predicting source of extra file**		**Predicting type of extra file**	
Features	**Types**	**Features**	**Types**	**Features**	**Types**
proxy	Proxy Network	dom	Webserver	dom	Webserver
proxy_prefix	Proxy Network	IP_dom	Network	IP_dom	Network
proxy_asn	Proxy Network	prefix_dom	Network	prefix_dom	Network
dom	Webserver	asn_dom	Network	asn_dom	Network
proxy_country	Proxy Network	Strict_Transport_Security	Webserver	Strict_Transport_Security	Webserver
IP_dom	Network	starting_domain	Webserver	as_name_dom	Network
prefix_dom	Network	as_name_dom	Network	handle_dom	Network
anonymity	Proxy Network	handle_dom	Network	reg_date_dom	Network
asn_dom	Network	reg_date_dom	Network	starting_domain	Webserver
starting_domain	Webserver	X_Frame_Options	Webserver	X_Frame_Options	Webserver
as_name_dom	Network	X_Powered_By	Webserver	X_Powered_By	Webserver
Strict_Transport_Security	Webserver	Content_Security_Policy	Webserver	X_XSS_Protection	Webserver
Proxy_rir	Proxy Network	X_XSS_Protection	Webserver	Content_Security_Policy	Webserver
handle_dom	Network	PoweredBy	Webserver	X_Content_Type_Options	Webserver
reg_date_dom	Network	X_Content_Type_Options	Webserver	PoweredBy	Webserver
link_type	File	JQuery	Technology	Frame	Technology
X_Frame_Options	Webserver	Frame	Technology	Open_Graph_Protocol	Technology
X_Powered_By	Webserver	cc_dom	Network	JQuery	Technology
PoweredBy	Webserver	server	Webserver	cc_dom	Network
JQuery	Technology	dom_protocol	Webserver	server	Webserver
Content_Security_Policy	Webserver	Open_Graph_Protocol	Technology	dom_protocol	Webserver
cc_dom	Network	HTTPServer	Webserver	rir_dom	Technology
HTML5	Technology	MetaGenerator	Technology	OS	Technology
-		Google_Analytics	Technology	MetaGenerator	Technology
-		OS	Technology	HTML5	Technology
-		HTML5	Technology	Google_Analytics	Technology
-		rir_dom	Network	proxy_type	Proxy Network
-		proxy_source	Proxy Network	proxy_source	Proxy Network
-		anonymity	Proxy Network	anonymity	Proxy Network
-		proxy	Proxy Network	-	
-		proxy_prefix	Proxy Network	-	

7 Conclusion

This work presents a novel approach for assessing the feasibility of predicting the types and sources of content manipulation by open web proxies. While previous studies have focused on identifying malicious patterns of behaviour and general analysis of proxy's misbehaviour on the Internet, our work takes a step further. It offers a novel work to proactively determine potential changes, looking simultaneously at the presence, type, and source of content change. As opposed to existing reactive techniques in nature, our approach is designed to predict changes without the need to fetch data through a potentially untrusted proxy.

This work presented a novel technique in observing modification types by leveraging clustering algorithms, contrary to previous studies [23, 39, 32] that analyzed the modifications with naive manual methods. Our machine learning-based method allowed us to derive 13 types of content alterations.

The identified modification types were leveraged in further analysis to proactively and accurately identify modification types applied to HTML and JS files, with 87% accuracy. The accuracy is increased by predicting modifications applied to JS and HTML files separately to over 90%. Besides, we recognized a relation between HTTP header settings of a website and modifying behaviour of the proxy toward the website via a detailed analysis of features and modification types. We found that improper configuration of HTTP headers allows proxies to freely apply manipulations to the website's content.

We have been able to design classification models that can proactively indicate with over 83.9% to 99.0% accuracy the probability of a proxy being able to add extra content not found in the original site, along with its type and source. The superiority of this experiment is that this method does not need any webpage content to be fetched through proxies, which reduces the risk of receiving unwanted content.

In conclusion, the presented work is able to accurately predict the specific type of modification that a proxy might apply to a web page content or whether the proxy will inject any extra file with its type. Thus, we can predict whether a proxy will show unwanted behaviour toward a webpage. In addition, we showed that misconfiguring HTTP headers can let proxies modify contents freely.

7.1 Future Work

This research work opens the door to many other interesting research paths around open web proxies and website configurations. Consequently, several research projects could be pursued based on the presented approach:

- **Collect a set of various domains**: In this study, we showed that the proxy behaviour is also based on the website configurations. Hence creating various categories for websites to compare the proxy behaviour toward categories of websites will result in interesting outcomes. These categories can include dynamic, static, popular, unknown, old (technology-wise), and new (technology-wise) domains.

- **Focus on malicious behaviour of proxy**: This work covers different behaviours of a proxy. Thus, it is interesting to have a similar approach focusing on the malicious behaviour of proxies. It can reveal details about malicious proxies and how they apply such changes.

- **Focus on the impact of HTTP header configurations**: To further mature our outcome that shows proxy behaviour is also dependent on website configuration, there can be research accomplished with a focus on the various configuration of HTTP headers for a single honey site. It can also analyze real-world websites with various HTTP header configurations. This analysis can precisely explain the impact of HTTP header configuration on proxy content manipulation.

- **Publish a proxy tester tool**: This study shows that the behaviour of the proxy toward a website is predictable. A general tool or platform can be created that warns of potentially malicious behaviour of a specific proxy while connecting to a particular domain. In addition, a blocking tool can be designed to block a connection if a potential malicious behaviour is predicted.

References

[1] Sumayah A Alrwais, Alexandre Gerber, Christopher W Dunn, Oliver Spatscheck, Minaxi Gupta, and Eric Osterweil. Dissecting ghost clicks: Ad fraud via misdirected human clicks. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 21–30, 2012.

[2] M. Alston. what are mobile proxies used for? [Online]. Available: `https://infatica.io/blog/all-you-need-to-know-about-mobile-proxies/`. [accessed: 17.03.2022].

[3] Apache. what are forward and reverse proxies. [Online]. Available: `https://httpd.apache.org/docs/2.0/mod/mod_proxy.html#forwardreverse`. [accessed: 17.03.2022].

[4] Sajjad Arshad, Amin Kharraz, and William Robertson. Include me out: In-browser detection of malicious third-party content inclusions. In *International Conference on Financial Cryptography and Data Security*, pages 441–459. Springer, 2016.

[5] Leo Breiman, Friedman Jerome, Stone Charles J., and Olshen R.A. *Classification and regression trees*. Wadsworth International Group, 1984.

[6] Yegui Cai, George OM Yee, Yuan Xiang Gu, and Chung-Horng Lung. Threats to online advertising and countermeasures: A technical survey. *Digital Threats: Research and Practice*, 1(2):1–27, 2020.

[7] Vinod Kumar Chauhan, Kalpana Dahiya, and Anuj Sharma. Problem formulations and solvers in linear svm: a review. *Artificial Intelligence Review*, 52(2):803–855, 2019.

[8] Jinchun Choi, Mohammed Abuhamad, Ahmed Abusnaina, Afsah Anwar, Sultan Alshamrani, Jeman Park, Daehun Nyang, and David Mohaisen. Understanding the proxy ecosystem: A comparative analysis of residential and open proxies on the internet. *IEEE Access*, 8:111368–111380, 2020.

[9] Taejoong Chung, David Choffnes, and Alan Mislove. Tunneling for transparency: A large-scale analysis of end-to-end violations in the internet. In *Proceedings of the 2016 Internet Measurement Conference*, IMC '16, page 199–213, New York, NY, USA, 2016. Association for Computing Machinery.

[10] D. constverum. Proxybroker, asynchronous public proxy finder. [Online]. Available: `https://proxybroker.readthedocs.io/en/latest/`. [accessed: 12.01.2022].

[11] David Roxbee Cox and E Joyce Snell. *Analysis of binary data*, volume 32. CRC press, 1989.

[12] X de Carné de Carnavalet and Mohammad Mannan. Killed by proxy: Analyzing client-end tls interception software. In *Network and Distributed System Security Symposium*, 2016.

[13] Zakir Durumeric, Zane Ma, Drew Springall, Richard Barnes, Nick Sullivan, Elie Bursztein, Michael Bailey, J Alex Halderman, and Vern Paxson. The security impact of https interception. In *NDSS*, 2017.

[14] T. Fortner. Selenium webdriver for automatic web crawling. [Online]. Available: `https://www.selenium.dev/documentation/webdriver/`. [accessed: 15.02.2022].

[15] Geoffrey Hinton and Sam T Roweis. Stochastic neighbor embedding. In *NIPS*, volume 15, pages 833–840. Citeseer, 2002.

[16] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282. IEEE, 1995.

[17] Yung-Tsung Hou, Yimeng Chang, Tsuhan Chen, Chi-Sung Laih, and Chia-Mei Chen. Malicious web content detection by machine learning. *Expert Systems with Applications*, 37(1):55–60, 2010.

[18] Vaibhavi Kalgutkar, Ratinder Kaur, Hugo Gonzalez, Natalia Stakhanova, and Alina Matyukhina. Code authorship attribution: Methods and challenges. *ACM Computing Surveys (CSUR)*, 52(1):1–36, 2019.

[19] J. Keenan. what is an anonymous proxy. [Online]. Available: `https://smartproxy.com/blog/what-is-an-anonymous-proxy`. [accessed: 17.03.2022].

[20] Zhou Li, Kehuan Zhang, Yinglian Xie, Fang Yu, and XiaoFeng Wang. Knowing your enemy: Understanding and detecting malicious web advertising. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 674–686, New York, NY, USA, 2012. Association for Computing Machinery.

[21] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[22] Mohammad Saiful Islam Mamun, Mohammad Ahmad Rathore, Arash Habibi Lashkari, Natalia Stakhanova, and Ali A Ghorbani. Detecting malicious urls using lexical analysis. In *International Conference on Network and System Security*, pages 467–482. Springer, 2016.

[23] Akshaya Mani, Tavish Vaidya, David Dworken, and Micah Sherr. An extensive evaluation of the internet's open proxies. In *Proceedings of the 34th Annual Computer Security Applications Conference*, ACSAC '18, page 252–265, New York, NY, USA, 2018. Association for Computing Machinery.

[24] J Mason, Michael Shepherd, and Jack Duffy. Feature selection for an n-gram approach to web page genre classification. In *Proc. Web Intelligence*, 2009.

[25] Rima Masri and Monther Aldwairi. Automated malicious advertisement detection using virustotal, urlvoid, and trendmicro. In *2017 8th International Conference on Information and Communication Systems (ICICS)*, pages 336–341. IEEE, 2017.

[26] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[27] R. McFarland. Asynchronous python http requests for humans using futures. [Online]. Available: `https://github.com/ross/requests-futures`. [accessed: 13.12.2021].

[28] Xianghang Mi, Xuan Feng, Xiaojing Liao, Baojun Liu, XiaoFeng Wang, Feng Qian, Zhou Li, Sumayah Alrwais, Limin Sun, and Ying Liu. Resident evil: Understanding residential ip proxy as a dark service. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1185–1201. IEEE, 2019.

[29] Eugene W. Myers. An o(nd) difference algorithm and its variations. *Algorithmica*, 1:251–266, 1986.

[30] Mark O'Neill, Scott Ruoti, Kent Seamons, and Daniel Zappala. Tls proxies: Friend or foe? In *Proceedings of the 2016 Internet Measurement Conference*, pages 551–557, 2016.

[31] A. Pathak. Different types of proxies based on their ip. [Online]. Available: `https://geekflare.com/understanding-proxy-types/`. [accessed: 17.03.2022].

[32] Diego Perino, Matteo Varvello, and Claudio Soriente. Long-term measurement and analysis of the free proxy ecosystem. volume 13, New York, NY, USA, November 2019. Association for Computing Machinery.

[33] Diego Perino, Matteo Varvello, and Claudio Soriente. Long-term measurement and analysis of the free proxy ecosystem. *ACM Transactions on the Web (TWEB)*, 13(4):1–22, 2019.

[34] Vasile Claudiu Perta, M Barbera, Gareth Tyson, Hamed Haddadi, Alessandro Mei, et al. A glance through the vpn looking glass: Ipv6 leakage and dns hijacking in commercial vpn clients. 2015.

[35] Igor Santos, Yoseba K Penya, Jaime Devesa, and Pablo Garcia Bringas. N-grams-based file signatures for malware detection. *ICEIS (2)*, 9:317–320, 2009.

[36] M. Stampar. Small tool used for finding out-dated js libraries. [Online]. Available: `https://github.com/stamparm/DSJS`. [accessed: 08.12.2021].

[37] Martin Sysel and Ondřej Doležal. An educational http proxy server. *Procedia Engineering*, 69:128–132, 2014.

[38] Kurt Thomas, Elie Bursztein, Chris Grier, Grant Ho, Nav Jagpal, Alexandros Kapravelos, Damon McCoy, Antonio Nappa, Vern Paxson, Paul Pearce, et al. Ad injection at scale: Assessing deceptive advertisement modifications. In *2015 IEEE Symposium on Security and Privacy*, pages 151–167. IEEE, 2015.

[39] Giorgos Tsirantonakis, Panagiotis Ilia, Sotiris Ioannidis, Elias Athanasopoulos, and Michalis Polychronakis. A large-scale analysis of content modification by open HTTP proxies. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[40] Gareth Tyson, Shan Huang, Felix Cuadrado, Ignacio Castro, Vasile C Perta, Arjuna Sathiaseelan, and Steve Uhlig. Exploring http header manipulation in-the-wild. In *Proceedings of the 26th International Conference on World Wide Web*, pages 451–458, 2017.

[41] Frank Vanhoenshoven, Gonzalo Nápoles, Rafael Falcon, Koen Vanhoof, and Mario Köppen. Detecting malicious urls using machine learning techniques. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016.

[42] Nevena Vratonjic, Julien Freudiger, and Jean-Pierre Hubaux. Integrity of the web content: The case of online advertising. In *2010 Workshop on Collaborative Methods for Security and Privacy (CollSec 10)*, Washington, DC, August 2010. USENIX Association.

[43] Louis Waked, Mohammad Mannan, and Amr Youssef. To intercept or not to intercept: Analyzing tls interception in network appliances. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 399–412, 2018.

[44] Zhongru Wang, Peixin Cong, and Weiqiang Yu. Malicious code detection technology based on metadata machine learning. In *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*, pages 394–399. IEEE, 2020.

[45] Nicholas Weaver, Christian Kreibich, Martin Dam, and Vern Paxson. Here be web proxies. In *Proceedings of the 15th International Conference on Passive and Active Measurement - Volume 8362*, PAM 2014, page 183–192, Berlin, Heidelberg, 2014. Springer-Verlag.

[46] Petros Xanthopoulos, Panos M Pardalos, and Theodore B Trafalis. Linear discriminant analysis. In *Robust data mining*, pages 27–33. Springer, 2013.

[47] Xinyu Xing, Wei Meng, Byoungyoung Lee, Udi Weinsberg, Anmol Sheth, Roberto Perdisci, and Wenke Lee. Understanding malvertising through ad-injecting browser extensions. In *Proceedings of the 24th international conference on world wide web*, pages 1286–1295, 2015.

[48] Hangfeng Yang, Shudong Li, Xiaobo Wu, Hui Lu, and Weihong Han. A novel solutions for malicious code detection and family clustering based on machine learning. *IEEE Access*, 7:148853–148860, 2019.

[49] Apostolis Zarras, Alexandros Kapravelos, Gianluca Stringhini, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. The dark alleys of madison avenue: Understanding malicious advertisements. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 373–380, 2014.

[50] Yubao Zhang, Jidong Xiao, Shuai Hao, Haining Wang, Sencun Zhu, and Sushil Jajodia. Understanding the manipulation on recommender systems through web injection. *IEEE Transactions on Information Forensics and Security*, 15:3807–3818, 2019.