PEOPLE4S DEMOCRATIC REBUPLIC OF ALGERIA

MMINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCHE

UNIVERSITY MOHAMED BOUDIAF - M'SILA

FACULTY: MATHIMATICS AND COMPUTER SCIENCE

DEPARTEMENT: COMPUTER SCIENCE

N° :……..

DOMAINE : MATHIMATICS AND COMPUTER SCIENCE

BRANCH: COMPUTER SCIENCE

OPTION : RTIC

Dissertation submitted to obtain Master degree

By:

- **HOUD MOHAMED**
- **DJEHICHE RAID**

SUBJECT

WEB BROWSER EXTENSION FOR DETECTING MALICIOUS WEB CONTENT

Publicly defended before the jury composed of

Ms .SAOUDI LALIA	University of M'sila	Supervisor

Academic year 2019/2020

ACKNOWLEDGMENT

FIRST OF ALL, I'D LOVE TO SHOW MY APPRECIATION AND GRATITUDE TO ALLAH THE ALMIGHTY WHO MADE THE PERFECT CONDITION FOR THIS WORK TO SEE THE LIGHT.
ANXIETY AND DEPRESSION WERE OVERWHELMING WHILE TRYING TO FINISH THIS PROJECT AND IT WAS ONLY POSSIBLE TO FIGHT THROUGH IT WITH THE SUPPORT OF SOME PEOPLE THAT I WANT TO THANK INDIVIDUALLY.
I SHALL GIVE THE CREDIT OF LEADING THIS PROJECT TO MY SUPERVISOR MS. SAOUDI LALIA, WHO OFFERED ME THE OPPORTUNITY TO WORK ON THIS PROJECT AND HAD ENOUGH PATIENCE TO REVIEW AND CORRECT MY WORK OVER AND OVER. LET ALONE SHARING HER WISE GUIDANCE AND UNCLOUDED THOUGHTS TO MAKE THIS PROJECT GO ON THE RIGHT PATH. SO THANK YOU, IT WAS PLEASURE TO WORK WITH YOU EVEN THOUGH I COULDN'T GIVE MY 100%.
THANKS TO MY FAMILY MEMBERS WHO FOR THE SUPPORT DURING MY EDUCATIONAL CAREER OVER THE YEARS.
THANKS TO MY TEAMMATE DJEHICHE RAID FOR BEING FRIEND AND LIFE MENTOR, THANKS TO MY CLASS MATES WHO MADE JUST THE PERFECT ATMOSPHERE TO MAKE THE PAST TWO YEARS BEARABLE.
THANK YOU ALL.

HOUD MOHAMED

ACKNOWLEDGMENT

FIRST OF ALL, I'D LOVE TO SHOW MY APPRECIATION AND GRATITUDE TO ALLAH THE ALMIGHTY WHO MADE THE PERFECT CONDITION FOR THIS WORK TO SEE THE LIGHT.
ANXIETY AND DEPRESSION WERE OVERWHELMING WHILE TRYING TO FINISH THIS PROJECT AND IT WAS ONLY POSSIBLE TO FIGHT THROUGH IT WITH THE SUPPORT OF SOME PEOPLE THAT I WANT TO THANK INDIVIDUALLY.
I SHALL GIVE THE CREDIT OF LEADING THIS PROJECT TO MY SUPERVISOR MS. SAOUDI LALIA, WHO OFFERED ME THE OPPORTUNITY TO WORK ON THIS PROJECT AND HAD ENOUGH PATIENCE TO REVIEW AND CORRECT MY WORK OVER AND OVER. LET ALONE SHARING HER WISE GUIDANCE AND UNCLOUDED THOUGHTS TO MAKE THIS PROJECT GO ON THE RIGHT PATH. SO THANK YOU, IT WAS PLEASURE TO WORK WITH YOU EVEN THOUGH I COULDN'T GIVE MY 100%.
THANKS TO MY FAMILY MEMBERS WHO FOR THE SUPPORT DURING MY EDUCATIONAL CAREER OVER THE YEARS.
THANKS TO MY TEAMMATE HOUD MOHMED FOR BEING FRIEND AND LIFE MENTOR, THANKS TO MY CLASS MATES WHO MADE JUST THE PERFECT ATMOSPHERE TO MAKE THE PAST TWO YEARS BEARABLE.
THANK YOU ALL.

DJHICHE RAID

Table of content

List of figures and tables

List of figures

List of tables

List of equations

General Introduction

GENERAL INTRODUCTION

Context of the Study

Nowadays, the web became a more essential part of human life, it meets all the needs of people of communicating, business, entertainment, banking, shopping and many other activities which done via web applications.

The content of the web is evolved from simple static HTML web pages to complex dynamic web application which is the result of the evolution on the web activities.

With this huge and fast evolution on the Internet and the exchange of sensitive data, the risk of steal or lose of user information is increasing day by day due to the increasing of malicious content on the web and the evolution of Cyber-attacks. This makes the security of web application more important than any time before.

This risk made people worry about their information and data which lead them to find methods or techniques to protect it.

Statement of Problem

The detection of malicious web content is a very sensitive topic which is always in research to keep pace with development of the web, to prevent this content there are many techniques have been developed. We can distinguish two categories of defense techniques:

Server-side: those techniques are implemented on the website most of them related to the configuration of files and source code to prevent the execution of scripts or explore the database.

Client side: on the client side the developers implement methods to protect clients from the harmful content on server there are two main methods:

Proxies or firewalls: These techniques require the installation and keep updating the used proxy or firewall which affect at the performance of the browser.

Browser extensions: it refers to the detection techniques implemented on the browser extensions.

The majorities of these extensions detect a limit type of malicious content and suffer from low rate of precision.

Objective

In order to prevent the malicious web content we propose a Google Chrome browser extension (WebGuard) to defeat the malicious content by detecting all types of malicious content.

We have minimize the features list and choose a high impact ones in order to accelerate the detection process and inform the user as soon as possible

WebGuard also uses Google safe browsing service to check the malicious web content we have collected a data set of 600 web sites between benign and malicious one.

Motivation

Malicious content and Cyber attacks are spreading crazily in the Internet due to the value of information and financial transactions.
According to Cyber statistics report presented in 201946% of web applications have critical vulnerabilities. And due to thycotic.com 73% of black hat hackers said traditional firewall and antivirus security is irrelevant or obsolete. Which mean most of web users are exposed to web attacks.[1]

Stealing users accounts and violate their privacy are one of the main reasons of Cyber attacks due to the wide spreading use of social media.

This high risk call to find how to detect the malicious content on the we band prevent it to offer a secure environment to web users and make them more comfortable about their information shared on the internet.

Report outline

This document was divided into four chapters:

- **The First chapter:** revolved around explaining chrome's extension architecture, URL architecture and website's request and response architecture and different malicious web content types.

- **The Second chapter**: focused on overview evaluating the already existing approaches used to detect malicious content and how they do it, closed by a comparison of previous works.
- **The Third chapter:** Introduced our claimed approach equipped by our conceptual model, techniques and tweaked methodologies to detect malicious web content.
- **The Fourth chapter:** contained the implementation and experimentation measures we made to make our approach with scores we achieved.
- And finally, a general conclusion for cloture.

CHAPTER 01

WEB BROWSER ARCHITECTURE& MALICIOUS WEB CONTENT

1.1. Introduction

Nowadays almost of people are surfing on the internet which it has huge number of websites. As we know the main way to access to the internet is with browser which allow to view and navigate the web pages and with this huge evolution of internet, many malicious web sites appeared that contains a harmful content which affect users privacy, this motivates developers to create systems and tools that help user detecting those malicious sites, among those tools we present web browser extensions.

1.2. Web browser extension

Web browser extensions are small software programs that customize the browsing experience. They enable users to tailor Chrome functionality and behavior to individual needs or preferences. They are built on web technologies such as HTML, JavaScript, and CSS[2].

An extension must fulfill a single purpose that is narrowly defined and easy to understand. A single extension can include multiple components and a range of functionality, as long as everything contributes towards a common purpose.

Extensions must have an icon that sits in the browser toolbar. Toolbar icons allow easy access and keep users aware of which extensions are installed. Most users will interact with an extension that uses a popup by clicking on the icon.

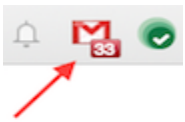

Figure 1.1: Extension icon[2].

1.2.1 Architecture

An extension architecture will depend on its functionality, but many robust extensions will include multiple components:

- **Manifest**
- **Background Script**
- **UI Elements**
- **Content Script**
- **Options Page**

1.2.1.1Manifest file

The manifest file, titled manifest.json, gives the browser information about the extension, such as the most important files and the capabilities the extension might use.

manifest.json

```
{
  "name": "My Extension",
  "version": "2.1",
  "description": "Gets information from C
  "icons": {
    "128": "icon_16.png",
    "128": "icon_32.png",
    "128": "icon_48.png",
    "128": "icon_128.png"
  },
  "background": {
    "persistent": false,
    "scripts": ["background_script.js"]
  },
  "permissions": ["https://*.google.com/'
  "browser_action": {
    "default_icon": "icon_16.png",
    "default_popup": "popup.html"
  }
}
```

Figure 1.2: Manifest file code[2].

1.2.1.2 Background Script

The background script is the extension's event handler; it contains listeners for browser events that are important to the extension. It lies dormant until an event is fired then performs the instructed logic. An effective background script is only loaded when it is needed and unloaded when it goes idle.

1.2.1.3 UI Elements

An extension's user interface should be purposeful and minimal. The UI should customize or enhance the browsing experience without distracting from it. Most

extensions have a browser action or page action, but can contain other forms of UI, such as context menus, use of the omnibox, or creation of a keyboard shortcut.

Extension UI pages, such as a popup, can contain ordinary HTML pages with JavaScript logic. Extensions can also call tabs, create or open additional HTML files present in the extension.

An extension using a page action and a popup can use the declarative content API to set rules in the background script for when the popup is available to users. When the conditions are met, the background script communicates with the popup to make its icon clickable to users.

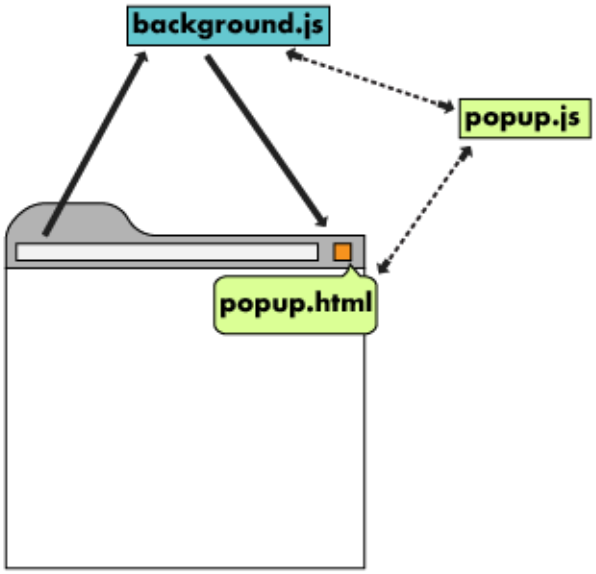

Figure 1.3:A browser window containing a page action displaying a popup[2].

1.2.1.4 Content scripts

Extensions that read or write to web pages utilize a content script. The content script contains JavaScript that executes in the contexts of a page that has been loaded into the browser. Content scripts read and modify the DOM of web pages the browser visits.

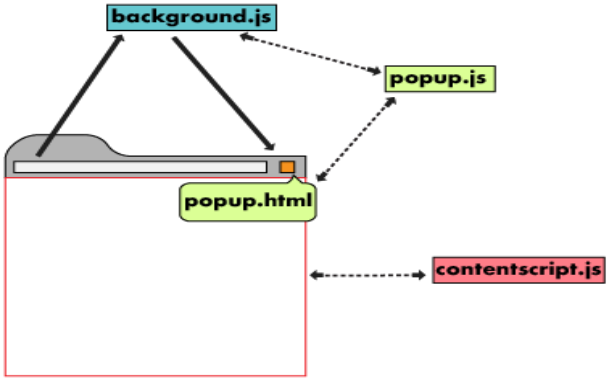

Figure 1.4: A browser window with a page action and a content script[2].

Content scripts can communicate with their parent extension by exchanging messages and storing values using the storage API.

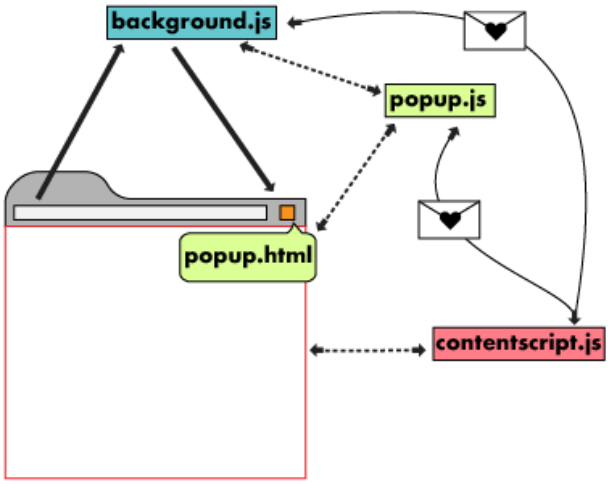

Figure 1.5: communication path between the content script and the parent extension[2].

1.2.1.5 Options Page

Just as extensions allow users to customize the Chrome browser, the options page enables customization of the extension. Options can be used to enable features and allow users to choose what functionality is relevant to their needs.

1.3 URL Architecture

1.3.1 What is a URL (Unified Resource Locator)

URL (or URL-address) is a special form of individual address of a certain resource on the Internet. It can refer to the website, some particular document, or an image. The Internet user just needs to insert this code into the location bar to find the needed

website, document, folder, or image. In plain language, it means the following: due to the URL address, the user gets information about where the needed information is located.[3]

1.3.2 What does URL stand for

A Uniform Resource Locator (URL), otherwise known as a Universal Resource Locator, is the address of a resource on the Internet and the protocol used to access it. It indicates the location of a web resource like a street address indicates where a person lives physically — because of this, an URL is often referred to as: “web address[4].

1.3.3 URL COMPONENTS

The major components of URL :

Figure1.6: simple URL components[3].

1.3.3.1 Protocol

The protocol declares how your web browser should communicate with a web server when sending or fetching a web page or document. The most common protocol is HTTP which stands for Hypertext Transfer Protocol.[3]

Another common protocol is HTTPS which stands for Hypertext Transfer Protocol Secure. You'll see this on secure pages, like shopping sites and log in pages. If you're visiting a site where you need to enter sensitive information, like bank details and passwords, make sure the protocol is declared as https. This means your web browser encrypts any information you provide so it can't be understood by any phishers who try to intercept the page during transfer.

Some protocols you're less likely to see includeFTP (File Transfer Protocol) POP (Post Office Protocol), SMTP(Simple Mail Transfer Protocol) and IMAP (Internet Message Access Protocol).

1.3.3.2 Subdomain

A subdomain is an additional part to your main domain name. Subdomains are created to organize and navigate to different sections of your website. You can create multiple subdomains or child domains on your main domain.

For example:store.yourwebsite.com

In this example, ‘store’ is the subdomain, ‘yourwebsite’ is the primary domain and ‘.com’ is the top level domain (TLD). You can use any text as your subdomain, but you want to make sure it’s easy to type and remember[25].

1.3.3.3 Domain name

Domain names are used to identify one or more IP address For example, the domain name **microsoft.com** represents about a dozen IP addresses. Domain names are used in URLs to identify particular Web pages. For example, in the URL: **http://www.pcwebopedia.com/index.html,** the domain name is **pcwebopedia.com[26].**

1.3.3.4 Port

The port number is rarely visible in URLs but always required. When declared in a URL it comes right after the TLD, separated by a colon. When it's not declared and in most cases where the protocol is http, port 80 is used. For https (secure) requests port 443 is used [3].

1.3.3.5 Path

The path refers to the exact location of a page, post, file, or other asset. It is often analogous to the underlying file structure of the website. The path resides after the hostname and is separated by “/” (forward slash). The path/file also consists of any asset file extension, such as images (.jpg or .png, etc.), documents (.pdf or .docx), and more[27].

1.3.3.6 Query

The query string contains data to be passed to server-side scripts, running on the web server. For example, parameters for a search. The query string preceded by a question mark (?), is usually a string of name and value pairs separated by ampersand (&), for example, ?first_name=John&last_name=Corner, q=mobile+phone, and so on[28].

1.3.3.7 Parameters

Query parameters are a defined set of parameters attached to the end of a URL. They are extensions of the URL that are used to help define specific content or actions based on the data being passed. To append query parameters to the end of a URL, a '?' Is added followed immediately by a query parameter.

To add multiple parameters, an '&' is added in between each. These can be created by any variation of object types or lengths such as String, Arrays and Numbers. The following is an example [29]:

http://example.com/path?name=Branch&products=[Journeys,Email,Universal%20Ads]

1.3.3.8 Fragment

is an anchor to another part of the resource itself. An anchor represents a sort of "bookmark" inside the resource, giving the browser the directions to show the content located at that "bookmarked" spot. On an HTML document, for example, the browser will scroll to the point where the anchor is defined; on a video or audio document, the browser will try to go to the time the anchor represents. It is worth noting that the part after the #, also known as the fragment identifier, is never sent to the server with the request[30].

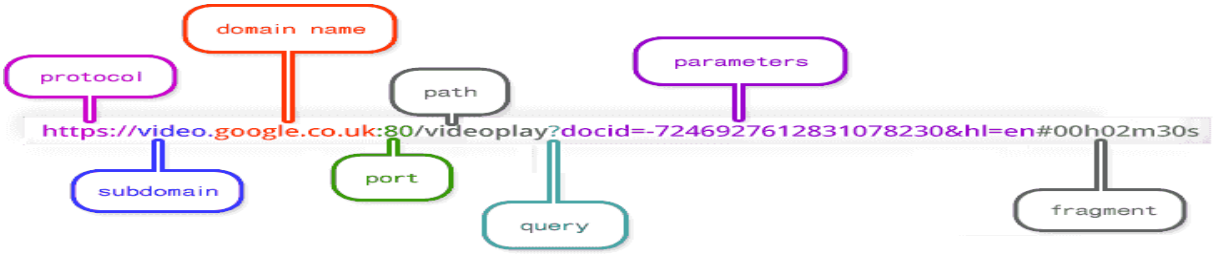

Figure 1.7: Complex URL components[4].

1.4 Web page

A Web page is a document for the World Wide Web that is identified by a unique uniform resource locator (URL) [5].

A Web page can be accessed and displayed on a monitor or mobile device through a Web browser. The data found in a Web page is usually in HTML or XHTML format. The Web pages usually also contain other resources such as style sheets, scripts and images for presentation. Users may be able to navigate to other pages through hypertext links.

1.4.1 Web page Structure

There are two structures : HTTP REQUESTS and HTTP RESPONSES[6].

HTTP requests, and responses [5]share similar structure and are composed of:

- A start-line describing the requests to be implemented, or its status of whether successful or a failure. This start-line is always a single line.
- An optional set of HTTP headers specifying the request, or describing the body included in the message.
- A blank line indicating all meta-information for the request has been sent.
- An optional body containing data associated with the request (like content of an HTML form), or the document associated with a response. The presence of the body and its size is specified by the start-line and HTTP headers.

The start-line and HTTP headers of the HTTP message are collectively known as the head of the requests, whereas its payload is known as the body

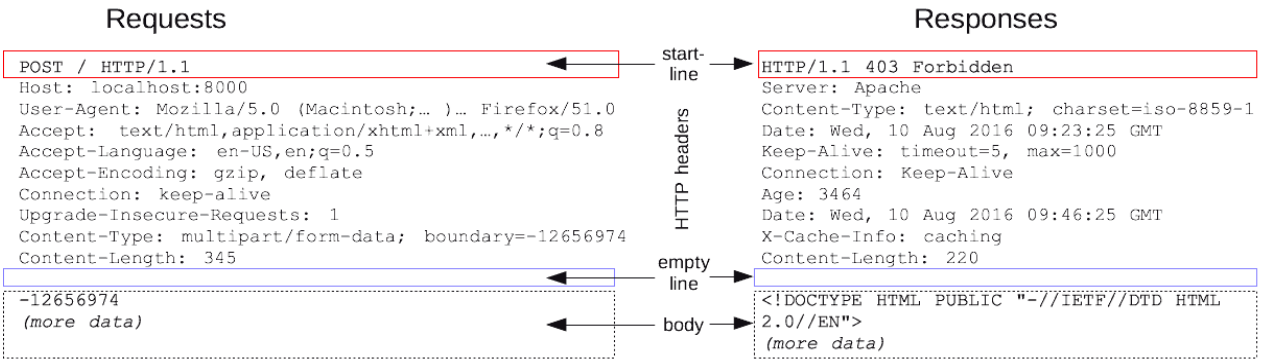

Figure 1.8: HTTP REQUESTS and HTTP RESPONSES Structure Similarity[5].

1.4.1.1 HTTP Requests

- Start line

HTTP requests are messages sent by the client to initiate an action on the server. Their start-line contains three elements:

- ❖ An HTTP method, a verb (like GET, PUT or POST) or a noun (like HEAD or OPTIONS), that describes the action to be performed. For example, GET indicates that a resource should be fetched or POST means that data is pushed to the server (creating or modifying a resource, or generating a temporary document to send back).
- ❖ The request target, usually a URL, or the absolute path of the protocol, port, and domain are usually characterized by the request context. The format of this request target varies between different HTTP methods. It can be:
  - o An absolute path, ultimately followed by a '?' and query string. This is the most common form, known as the origin form, and is used with GET, POST, HEAD, and OPTIONS methods.

    POST / HTTP/1.1

    GET /background.png HTTP/1.0

    HEAD /test.html?query=alibaba HTTP/1.1

    OPTIONS /anypage.html HTTP/1.0

  - o A complete URL, known as the absolute form, is mostly used with GET when connected to a proxy.

- The authority component of a URL, consisting of the domain name and optionally the port (prefixed by a ':'), is called the authority form. It is only used with CONNECT when setting up an HTTP tunnel.
    - Ex: CONNECT developer.mozilla.org:80 HTTP/1.1
- The asterisk form, a simple asterisk ('*') is used with OPTIONS, representing the server as a whole.
    - OPTIONS * HTTP/1.1

❖ The HTTP version, which defines the structure of the remaining message, acting as an indicator of the expected version to use for the response**.**

- Headers

A case-insensitive string followed by a colon (':') and a value whose structure depends upon the header. The whole headers, including the value, consist of one single line, which can be quite long.

There are numerous request headers available. They can be divided in several groups:

❖ General headers, like Via(is used to inform the server of proxies through which the request was sent), apply to the message as a whole.

❖ Request headers, like User-Agent, Accept-Type, modify the request by specifying it further (like Accept-Language), by giving context (like Referer), or by conditionally restricting it (like If-None).

❖ Entity headers, like Content-Length which apply to the body of the request. Obviously, there is no such header transmitted if there is no body in the request**.**

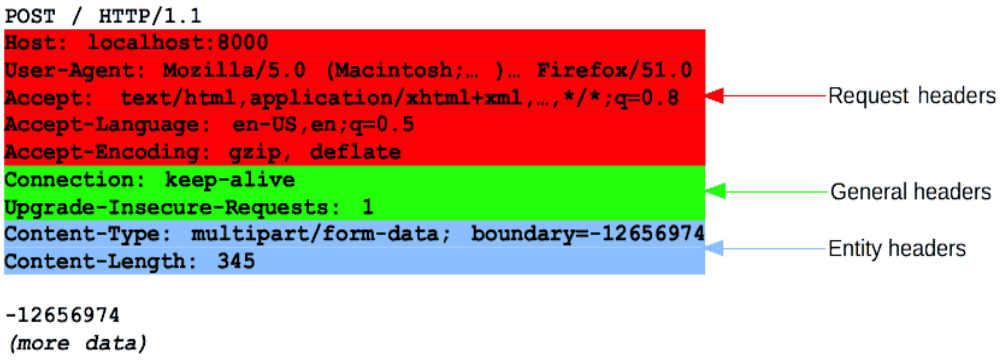

Figure 1.9: Example of headers in an HTTP request[5].

- Body

The final part of the request is its body. Not all requests have one: requests fetching resources, like GET, HEAD, DELETE, or OPTIONS, usually don't need one. Some requests send data to the server in order to update it: as often the case with POST requests (containing HTML form dat**a).**

Bodies can be broadly divided into two categories:

- ❖ Single-resource bodies, consisting of one single file, defined by the two headers: Content-Type and Content-Length.
- ❖ Multiple-resource bodies, consisting of a multipart body, each containing a different bit of information. This is typically associated with HTML Forms.

4.1.2 HTTP Responses

- Status line

The start line of an HTTP response, called the status line, contains the following information:

- ❖ The protocol version, usually HTTP/1.1.
- ❖ A status code, indicating success or failure of the request. Common status codes are 200, 404, or 302.

- ❖ A status text. A brief, purely informational, textual description of the status code to help a human understand the HTTP message.
- ❖ A typical status line looks like: HTTP/1.1 404 Not Fou**nd.**

- Headers

HTTP headers for responses follow the same structure as any other header: a case-insensitive string followed by a colon (':') and a value whose structure depends upon the type of the header. The whole header, including its value, presents as a single line.

There are numerous response headers available. These can be divided into several groups:

- ❖ General headers, like Via(is used to inform the server of proxies through which the request was sent), apply to the whole message.
- ❖ Response headers, like Vary and Accept-Ranges, give additional information about the server which doesn't fit in the status line.
- ❖ Entity headers, like Content-Length, apply to the body of the response. Typically, no such headers are transmitted when there is no body in the response**.**

Figure 1.10: Example of headers in an HTTP response[5].

- Body

The last part of a response is the body. Not all responses have one: responses with a status code, like 201 or 204, usually don't.

Bodies can be broadly divided into three categories:

- Single-resource bodies, consisting of a single file of unknown length, encoded by chunks with Transfer-Encoding set to chunked.
- Multiple-resource bodies, consisting of a multipart body, each containing a different section

1.5 Malicious web content

A malicious website is a site that attempts to install malware (a general term for anything that will disrupt computer operation, gather your personal information or, in a worst-case scenario, gain total access to your machine) onto your device. This usually requires some action on your part [24].

Malicious web content is divided into many several categories or types; in this section we picked the most common types of malicious content

1.5.1 Redirections

When hackers compromise a website, they can add malicious code to redirect specific users to another website. Some common methods used by attackers include modifying web server configuration rules , adding server-side scripts, or even including client-side JavaScript to create these malicious redirect[31].

To accomplish the conditional elements, attackers regularly limit redirects to referrers or user agents to target specific visitors and avoid detection. The final destinations are often infected with malware or set up for phishing, while the original website gets blacklisted by search engine authorities.

1.5.2 SEO (Search Engine Optimization)Spam

Hackers gain entry using some vulnerability that’s present on the vulnerable site, like a weak password or a security gap in an outdated plug-in.
Once inside, they start doing things to hijack your SEO accomplishments. They find your top-ranking pages and insert their own hyperlinks and spam keywords.

Ranking on Google takes a lot of effort but with it comes great benefits. These hackers would rather let you do all the hard work of SEO and digital marketing, and then use your website to promote their product/service. This is why SEO spam is also known as spamdexing or search engine poisoning (SEP).
The hack is so popular because it can target WordPress websites of all sizes and not necessarily just the large ones**.** The most common victims are small websites, NGOs, and WordPress blogs, that are not secured by SSL certificates or do not have any security measures in place.
This hack is well-disguised and hidden away from plain-sight of the owner. Therefore, it is one of the most difficult ones to detect. You could be hacked for a long time without even knowing it[32].

1.5.3 Malicious JavaScript

JavaScript is not an insecure programming language. It's just that code bugs or improper implementations can create backdoors which attackers can exploit.[33]

When you're browsing a website, a series of JavaScript (.js) files are downloaded on your PC automatically. These files are executed through your browser, so you can:

- see the content of the website you're on
- perform various actions (example: fill out a form or download a file from a website)
- see the online ads (banners) on that website, etc.

There are 8 main ways in which JavaScript is used to spread malware in current cyber attacks:

- ***Malicious JavaScript code injections in legitimate websites*** – used to redirect users to malware-laden websites or to exploit servers that trigger malware infections.

- ***Hidden iFrames*** – that load JavaScript malware from compromised sites, malware which then tries to execute code in the browser to infect the PC.

- ***Malicious JavaScript code injections in online advertising networks*** – which appears in online banner ads and also silently redirect users to malicious web locations.

- ***Drive-by downloads –*** which use infected JavaScript files to launch malware infections.

- ***Malicious JavaScript attachments*** – which are ran through a Windows program and can trigger insidious infections outside the browser.

- ***Infected downloads triggered through compromised JavaScript code injects*** – such as fake antivirus products, which are one of the **most common** scams on the Internet. These can compromise your system beyond the point of no return.

- ***Browser add-ons and plugins*** – these can be either infected or they can load external content loaded with malware from external sources.

- ***Fake software pop-up messages –*** that cyber crooks can easily forge to look real and convincing.

1.5.4 Phishing

Phishing is a cybercrime in which targets are contacted by email, telephone or text message by someone posing as a legitimate institution to lure individuals into providing sensitive data such as personally identifiable information, banking and credit card details, and passwords.[34]

There several ways to perform a phishing attack such as:

- **Website fabrication.** Someone may mock up a website that looks like the login page for an existing software company, but with a slightly different URL. If an end-user is fooled, they may enter their username and password without forethought.
- **Email spoofing.** Phishing attempts are also common via email. A hacker may send a message to someone, pretending to be an authority figure at a well-known website or a trusted partner institution, and request information from their recipient.

- **Direct social engineering.** Other attempts are more direct and specific; for example, someone may try to initiate a conversation via instant messenger.

1.5.5 Backdoors

A backdoor is a malware type that negates normal authentication procedures to access a system. As a result, remote access is granted to resources within an application, such as databases and file servers, giving perpetrators the ability to remotely issue system commands and update malware. Backdoor installation is achieved by taking advantage of vulnerable components in a web application. Once installed, detection is difficult as files tend to be highly obfuscated [35].

Webserver backdoors are used for a number of malicious activities, including:

- Data theft
- Website defacing
- Server hijacking
- The launching of distributed denial of service (DDoS) attacks
- Infecting website visitors (watering hole attacks)

Backdoor Trojan injection is often done in a two-step process to bypass security rules preventing the upload of files above a certain size. The first phase involves installation of a dropper—a small file whose sole function is to retrieve a bigger file from a remote location. It initiates the second phase—the downloading and installation of the backdoor script on the server.

1.6 Conclusion

On this chapter we talked about three major concepts which are Google Chrome extension structure, URL structure and webpage structure and different malicious web content types. Those concepts are the base of our work.

CHAPTER 02

RELATED WORKS

2.1. Introduction

In recent years, web content security has become the main target of developers. In this chapter we will present the most recent work concerning the development of extensions to secure the content received by web browsers.

2.2. Approach 1

Siragelding and all [7] proposed an approach to secure web content based on a model of three components: Feature Extractor, Learning & Model Selector, and the Detector. It begins with the extraction of characteristics, then training on a data set, and finally the detection of malicious content

.

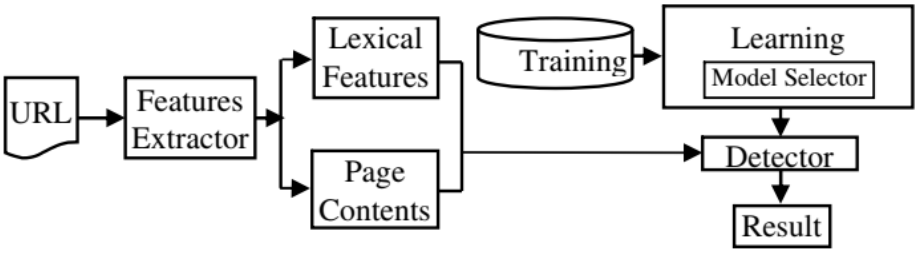

Figure 2.1: .approach general structure[7].

2.2.1 Feature Selector

This is the first component; it is responsible for extracting the Features of malicious web content.

The main step in this work is the selection of features, this model uses 39 features, 21 of them are new or modified and the others are obtained from previous works, the set of features is divided into two categories:

2.2.1.1 URL Lexical Features

The model uses the URL for the detection of malicious content, the dataset contains 10 Features, 6 are obtained from previous work and 4 are new, such as: number of

tokens, number of directories, length of host, length of URL, number of digits in the host, and number of parameters.

2.2.1.2 Web page content features

In general, malware loads web pages with tags; this model uses 29 characteristics such as: number of local links, number of external links, number of tags, number of applets, number of objects, number frames, number of shapes, and abnormal visibility. Also they use other features against dangerous Javascripts like: escape, unscape, evale, exec and unbound, obfuscation and encoding, length and number of internal and external scripts, number of methods .

2.2.2 Learning & Model Selector

It needs a data set to train the model and select the classification algorithm that gives the best results.

2.2.3 Detector

It uses the training model from the previous step to make the classification.

And for the classification they use 5 main algorithms are: Support Vector Machine (SVM), Decision Tree (DT), Naive Bayes (NB), K Nearest Neighbor (KNN), Artificial Neural Network (ANN).

2.3. Approach 2

Anand Desai and all [8] proposed another approach to secure web content against phishing; they use the UCI Dataset (University of California Irvine) to train a classifier after the extraction of features from the URL. This approach is based on three major steps: Obtaining Dataset, extraction of features and classification.

2.3.1 Obtaining Dataset

Dataset is obtained from UCI-Machine Learning Repository which contains 11055 sites which are classified as phishing sites and benign sites, and each site has 30 features.

And because some features uses standard database and some of them are impossible to extract the dataset restructured to contain 22 features

2.3.2 Extraction of the features

The model is based on 22 major Features such as: the length of the URL if the length of the URL is more than 52 characters the site is considered as a suspect phishing site. Google Index it checks whether the site is found in Google Index or not, generally Phishing sites are not indexed by Google.

2.3.3 Classification

For site classification the approach is based on three Machine Learning Algorithms which are: K-Nearest neighbor (KNN), Support Vector Machines (SVM) and Random Forest.

2.4. Approach 3

This approach is proposed by Immadisetti and all[10] based on machine learning to detect malicious content in URLs to classify them as malicious or benign.

This approach takes two steps:

2.4.1. Extraction of the features

It is the first stage which concerns the selection of the features which are used to classify the URLs. Here are the features used in this approach illustrated in the table 2.1.

2.4.2 Classifier

It is a Machine Learning model which is based on the Convolutional Neural Networks (CNN) algorithm, to analyze the extracted Features and classify the sites.

S1.No	Features and Types		
	Feature	**Type**	**Category a**
1	Token Count	Float	Lex Type
2	Average Path Token	Float	Lex Type
3	Largest Path	Float	Lex Type
4	Largest Token	Float	Lex Type
5	IP Address Presence	Binary	Lex Type
6	Largest Domain Length	Float	Lex Type
7	Number of Dots	Float	Lex Type
8	Length of the URL	Float	Lex Type
9	Path Token Count	Float	Lex Type
10	Domain Token Count	Float	Lex Type
11	AverageTokenLength	Float	Lex Type
12	Average Domaine Token Length	Float	Lex Type
13	Length of the Host	Float	Lex Type
14	Security Sensitive Words	Float	Behavior
15	Autonomous System Number	Float	Network
16	SafeBrowsing	Rank Real Float	3edParty[2]
17	Rank Country	Int	Geo[3]
18	Rank Host	Binary	Relative[4]
19	Malicious[4]	Float	Result[5]

Table 2.1 : URLfeatures

2.5MalURLs

This extension is proposed by M. Aldwairi and all [9], It is an extension which detects malicious websites based on the lexical Features of the URL and the Features of the host, it uses genetic algorithms and the Naive Bayes classifier to classify the sites.

2.5.1 Features extraction

The features used in this approach are divided into three categories:

2.5.1.1 The lexical features of URL

The lexical features are the properties of the URL, the properties used in this approach are: the length of the main domain, domain, hostname, length of the URL, number of dots in the URL, also tokens in hostnames and tokens in the URL path.

2.5.1.2 The features of the host:

These features are derived from the host such as: IP address, geographic properties, DNS properties, time to live (TTL), DNSA, DNS PTR, DNS MX records, WHOIS information and dates. These lexical features are very important to increase the accuracy of the classifier.

2.5.1.3 Special features

They are news that report good results such as: JS enable / disable, HTML Title tag content (<title></title>), 3-4-5 grames, Term Frequency and Invers Document Frequency (TF -IDF)

2.5.2 Classifier

It is a component based on the Naive Bayes algorithm, it uses the features extracted from the URL of the previous step to classify the sites are malicious or not.

2.5.3 Genetic algorithm

The genetic algorithm is used to generate the appropriate dataset from the initial dataset.

2.6. MWPT

M. Anantha Raman and all [11] developed an extension for firefox mobile, which is used to detect malicious web pages, this extension uses a Backend server to extract the features of the URL and send them to a classifier based on Machine Learning which classifies the web page malicious or benign, if the URL is malicious a warning message is displayed to notify the user not to consult this site, otherwise the page is loaded automatically in the web browser.

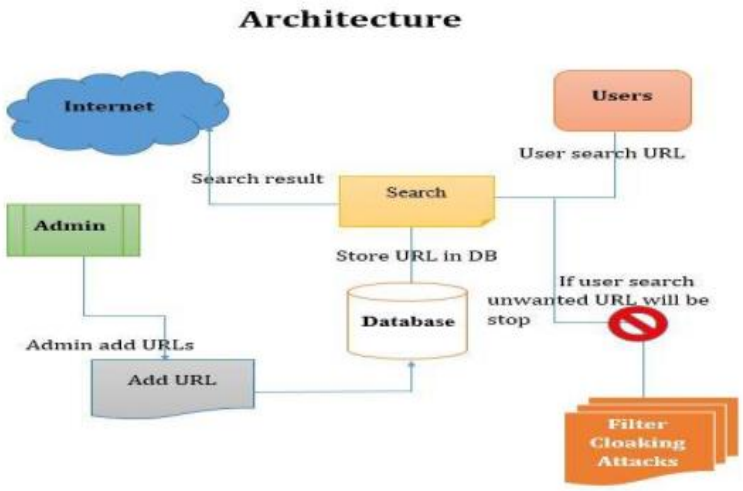

Figure 2.2: MWPT architecture[11]

2.7 Comparison of previous works

In this section we would make a comparison of the works that we reviewed in this chapter. Our comparison based on the common characteristics of those works.

	Approach [1]	**Approach[2]**	**Approach [3]**	**MALURLS**	**MWPT**
URL analysis	×	×	×	×	×
Web page analysis	×				
Type	Framework	Google Chrome extension	Classification model	Lightweight System	Mozilla Firefox mobile extension
Detected content type	Web pages	Fishing	Websites	Websites	Webpage

Table 2.2: Approaches Comparison

MWPT is an extension developed for Mozilla Firefox mobile and it based only on URL analysis.

MALURLS is a lightweight system also based only on URL analysis.

Approach [3] a classification model based on URL features.

Approach [2] also based on URL features but it is a Google Chrome extension.

Approach [1] is the only work uses webpage content features in the classification.

2.8. Conclusion

In this chapter we talked about the existent works, explained their characteristics and functionalities and finally make a comparison between them to propose our own extension in the next chapter.

CHAPTER 03: WEBGUARD EXTENTION

3.1 Introduction

After studying the previous works to detect malicious web content, in this chapter we propose our method to develop chrome extension that detects malicious web content. Our extension is based on machine learning to identify the malicious websites.

3.2 Our work

Most of users suffer from malicious web content while surfing the net as we see in the previous chapter, the available used tools detect a limit types of malicious content.

So in order to make the experience of surfing more secure on the client side, we propose a web browser extension (Web Guard) for Google Chrome, that analyses requests and responses from websites to detect malicious web content.

3.3 Web Guard Conception

Web Guard is a Google Chrome extension that intercepts any traffic between web server and Chrome browser. WebGuard analyses the requests and responses from web sites in order to detect the malicious content of websites.

WebGuard has three major components: Feature Extractor,, training module and the classifier (figure1):

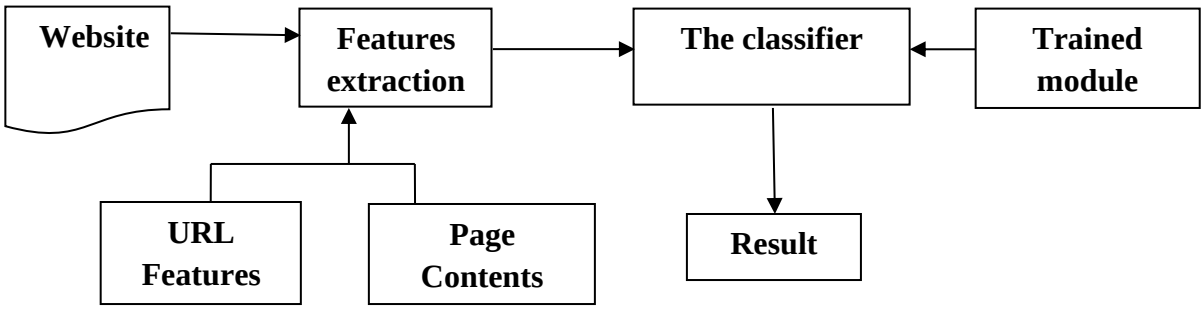

Figure 3.1: General structure of WebGuard extension

3.3.1 Features Extraction

We choose specific features extracted from URL and Web Page that help our classifier to classify the web site, the web sites have a huge number of features so we reduce the number of features in order to optimize the execution time of the extension.

The collection of features is divided into categories as bellow [12]:

3.3.1.1 URL Lexical Feature

Lexical feature is the textual properties of the URL itself. This feature list [2, 4] includes the length of the hostname, the length ofthe entire URL, the number of dots in the URL and token in the hostname (delimited by '.'), in the path URL (strings delimited by'/', '?', '.', '=', '-' and '_'). All these features are integer valued and their values range helps in identifying the malicious web URLs.For example: In Phishing attack the malicious URL has mimic appearance to the benign web page and only differs by these bag-of-words.

1. http://83.16.123.18/www.paypal.com/update.htm?=
2. http://signin.paypal.com@10.19.32.3/
3. www.pay.pal.com Host Based Feature:

The above Phishing URL appears similar to that of benign URL https://www.paypal.com/home .All three URLs above contain "Pay", "Pal", "com" string which makes the malicious URL look alike to benign one and user can easily be trapped by the attacker.

3.3.1.2. Page Content Based Feature

Source code of the web page is important source of feature for identifying malicious web pages. The contents based feature of web pages includes number of functions, objects, Frames, data streams, hyperlinks, In-bound, Out-bound, #f hidden elements.

These features are based on the vulnerabilities and loopholes in the language used for scripting purpose. The scripting languages include the DHTML, CSS, JavaScript.

3.3.1.3 .DHTML or HTML features: DHTML or HTML feature includes word count, Average word length, distinct word count, and size of I-frame. These features are exploited by the attacker for obfuscating the malicious code or script into the web page.

3.3.1.4 JavaScript features: JavaScript is one of the popular scripting languages which is used as the validation code for the web page. Some functions in the JavaScript are quite vulnerable and are exploited by the attacker for injecting the malicious script. These malicious scripts get executed just by clicking on the page. Some of JavaScript functions which are most exploited are eval(), escape(), unescape(),exec(), ubound() etc.

3.3.2 Google safe browsing

Safe Browsing is a Google service that lets client applications check URLs against Google's constantly updated lists of unsafe web resources. Examples of unsafe web resources are social engineering sites (phishing and deceptive sites) and sites that host malware or unwanted software. Come see what's possible.

With Safe Browsing you can:

- Check pages against our Safe Browsing lists based on platform and threat types.
- Warn users before they click links in your site that may lead to infected pages.
- Prevent users from posting links to known infected pages from your site.

The table below present the list of extracted features:

Features	Explanation
Double slash redirection	Detecting if there are double slash redirecting in the URL
Longest domain length	Calculate the Length of the Domain
Number of dots	Calculate the number of dots in the URL
URL length	Calculate the Length of the URL
host length	Calculate the length of the Host
URL Scripts	Detect the scripts and commands in the URL
sub-domain(www)	Check the type of the Sub-Domain
HTTPS	Check the HTTPS protocol
domain (.com,.net…)	Check type of the Domain
numbers in URL	Detect if there is numbers in the URL

count number	Count numbers in The URL
count letter	Count letters in the URL
rank host	Check the rank of the Host
Googlesearch engine	Check the URL by Google Search Engine
JavaScript in URL	Detect JAVA Script in the URL
malicious script in content	Detect the malicious scripts in the web page
external links	Detect if there are external links in the web page
iframe count	Count the I frames in webpage

Table 3.1: List of selected features

3.3.3 Features Selection

In our study we select a group of relevant features for determinate the malicious content. The selection is done according to the impact of features on the obtained results.

The importance of features is calculated with Mean Decrease Accuracy method which calculate the impact of each feature on the accuracy of the classifier

Mean Decrease Accuracy method is to directly measure the impact of each feature on accuracy of the model. The general idea is to permute the values of each feature and measure how much the permutation decreases the accuracy of the model. Clearly, for unimportant variables, the permutation should have little to no effect on model accuracy, while permuting important variables should significantly decrease it[5].

After the tests we have eliminate 6 features which have no Impact at the results of the classification. Table3.2 shows the eliminated features after the selection phase.

List of Features
ip length
average path token
number of internal links
number of @
number of 0
longest word length in URL

Table 3.2: eliminated features

3.3.4 Features Preprocessing

In this phase, the unstructured information about the URL (e.g. textual description) is appropriately formatted, and converted to a numerical vector so that it can be fed into machine learning algorithms. For example, the numerical information can be used as is, and Bag-of-words is often used for representing textual or lexical content[2].

This phase has four major steps:

1. Read the URL as string from the dataset file
2. Create new file to store the URL as numeric vector
3. .Extract features from the URL by calling the extraction function which represent each feature as a numeric value and convert the URL into a numeric vector
4. .Store the numeric vector in the created file.

3.3.5 Learning phase

In order to train the classifier we have prepared a dataset composed of 600 site .

Our dataset contain 300 safe web sites which extracted from Alexa top sites and 300 malicious web sites extracted from previous researches.

The structure of the data set is shown in the figures below

```
@attribute Double_slash_redirection {1,,0,-1}
@attribute Longest_domain_length {1,,0,-1}
@attribute Number_of_dots {1,,0,-1}
@attribute URL_length {1,,0,-1}
@attribute host_length {1,,0,-1}
@attribute URL_Scripts {1,,0,-1}
@attribute domain {1,,0,-1}
@attribute HTTPS {1,,0,-1}
@attribute sub-domain {1,,0,-1}
@attribute numbers_in_URL {1,,0,-1}
@attribute count_number {1,,0,-1}
@attribute count_letter {1,,0,-1}
@attribute rank__host {1,,0,-1}
@attribute Googlesearch_engine {1,,0,-1}
@attribute malicious_script_content {1,,0,-1}
@attribute external_links {1,,0,-1}
@attribute JavaScript_in_URL {1,,0,-1}
@attribute Google_Safe_Browsing {1,,0,-1}
@attribute iframe_count {1,,0,-1}
@attribute Class {1,-1}

@data
1,1,1,1,1,1,1,1,1,1,1,1,1,1,-1,1,-1,-1,-1,1
1,1,1,1,0,1,1,1,1,1,1,1,1,1,-1,1,-1,-1,-1,1
1,1,1,1,1,1,1,1,1,-1,-1,-1,1,1,-1,1,-1,1,1,1
1,1,1,1,1,1,1,1,1,1,1,1,-1,1,1,-1,1,-1,-1,-1,1
1,0,1,0,-1,1,1,1,1,1,1,1,1,1,-1,1,1,-1,1,1
1,1,1,0,-1,1,1,1,1,1,1,1,1,1,-1,1,-1,-1,1,1
1,1,1,0,-1,1,1,1,1,1,1,1,1,1,-1,1,-1,-1,-1,1
```

Figure 3.2: Structure of the dataset

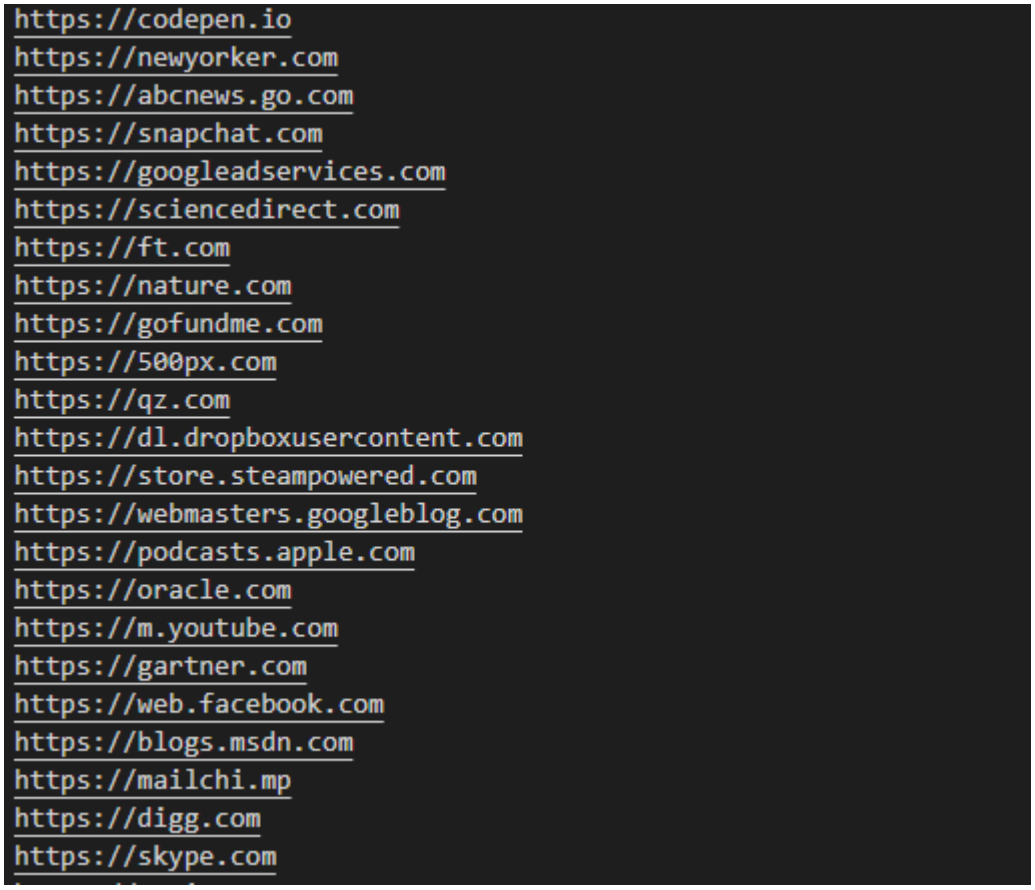

Figure 3.3: some websites of the dataset

3.3.6 Trained module generation

Our work is based on analyzing both HTTP request and response, which make it able to detect the advanced malicious content cached on the web page if the URL analyzing result is benign. As described in the following flow chart (figure 3.7):

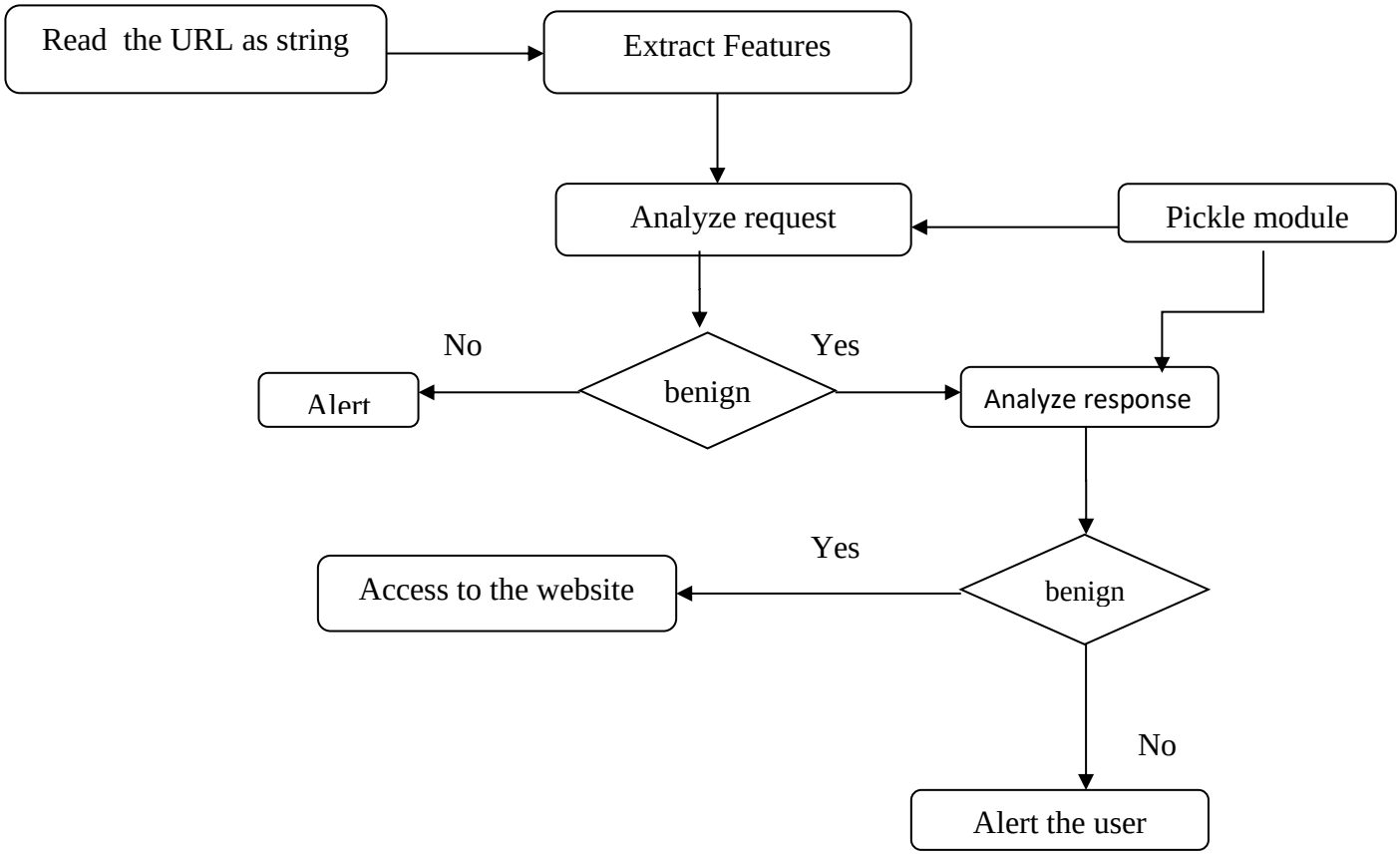

Figure 3.4 WebGuard flowchart

Unlike the previous works our classification approach is based on a module generated in training phase as .PKL file .

The PKL file type is primarily associated with Pickle by Python. Pickle is a module of Python that serializes objects so they can be saved to a file, and loaded again when called by the program. Serialization of files is called pickling and deserialization is called unpickling.. Files that are serialised by the Pickle module are stored in the PKL format [23].

This module stores the data of the training phase so in new classification operation the classifier predicts directly from the module doesn't need a new training session which is a positive point to reduce the process time.

3.3.6 The classifier

For good classification results we test our extension with four machine learning algorithms which are: Support Vector Machine (SVM),Naive Bayes(NB),K Nearest Neighbors(KNN),Random Forest Algorithm(RF).The selection based on the best results of precision sorted by the previous algorithms.

3.3.6.1 Support Vector Machine (SVM)

A support vector machine or SVM [14]is a supervised method ofmachine learning for classification. It tries to determine a decisionboundary that separates the web pages into malicious and benignclass.SVM maximizes the distance of the hyper plane and theresulting decision boundaries are robust to slight change of thefeature vectors.

When input data is not separable by the linear function, SVMsuse a kernel function to map the data into a higher dimensionalspace, and separate the data on the mapped dimension.

3.3.6.2Naive Bayes

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature[15].

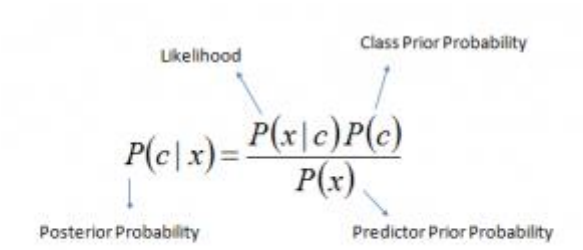

Figure 3.5:.Bayes' Theorem equation

3.3.6.3 K Nearest Neighbors(KNN)

K Nearest-Neighbor (KNN): [13]is a simple algorithm for predicting a class of an example. This classifier is supervised learning based on the distance of the example.The training stage simply stores all training examples with their labels. To predict theclass for a new test example, first it computes its distance to every training exampleand then, keeps the k closest training examples, where $k \geq 1$. Finally, it looks for thelabel that is most common among these examples. This label is assigned to this testexample as the predicted result

3.3.6.4 Random Forest Algorithm

Random Forest[16] is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting.

The below diagram explains the working of the Random Forest algorithm:

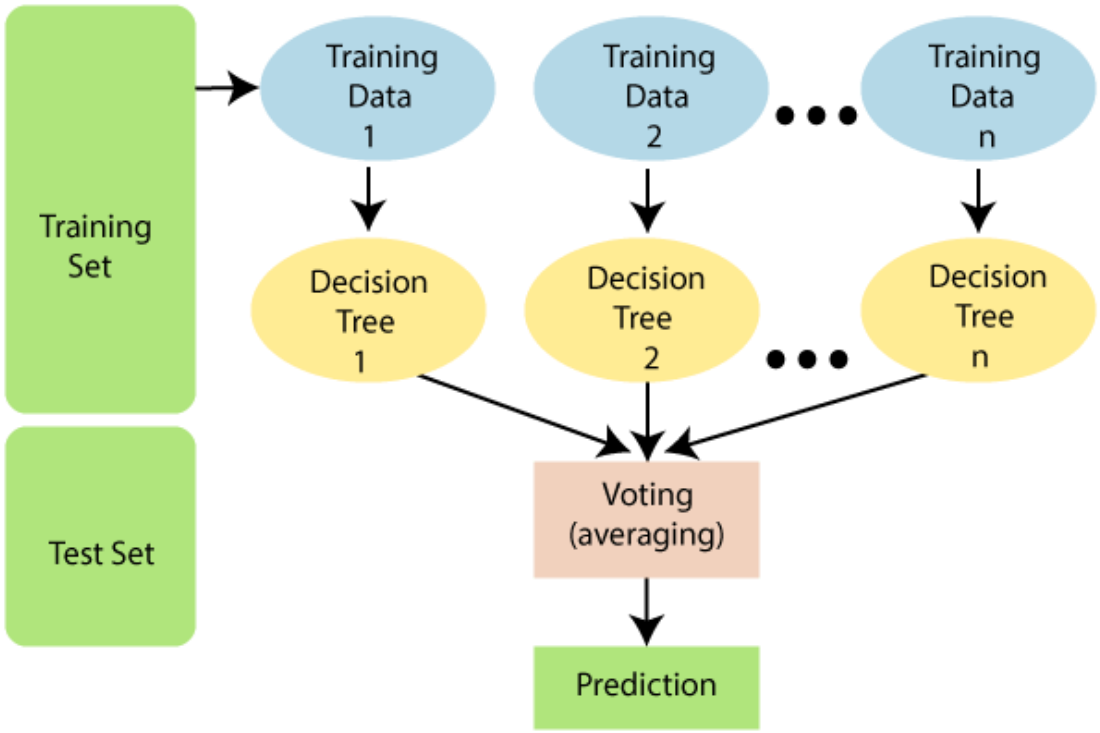

Figure 3.6: Structure of Random Forest Tree Algorithm[16].

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

❖ Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

- It takes less training time as compared to other algorithms.

- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

3.4 Conclusion

In this chapter we present the conception of Webguard and explain its components, we present also the used methodology to protect web users from the malicious content of the websites.

In the next chapter we will present the steps how we implement WebGuard, the used tools in implementation and discuss the results of the experiments

CHAPTER 04

IMPLEMENTATION ANDEXPERIMENTATION

4.1. INTRODUCTION

In this chapter we present the steps of implementation and experimentation of our work that we explained in chapter 3 which is developing a web browser extension to detect malicious web content .We have so many experiments to test and measure the performance of our extension.

4.2 Experimentation setup

We run all our tests on a custom built computer while disabling all sort of processes that has may affect the performance one way or another.

- CPU: Intel i5 9400F @4.1Ghz
- RAM: 8 GB @3000Hz
- Operating System: Microsoft Windows 10 Pro 64 bit
- Built in monitoring tool: Google dev tool
- Data mining tool: Weka 3.8
- IDE: visual studio code x64-1.48.2
- Web browser: Google Chrome
- Server environment: Apache PHP 5.6.18
- Programming language: Python 3 ,PHP , JavaScript

4.3 Developing tools

We use to develop our web extensions the following tools:

4.3.1. Google chrome browser

We choose Google Chrome browser to do our study because chrome is the most used browser according to global statistics (in figure 4.1),also because it is an open source browser.

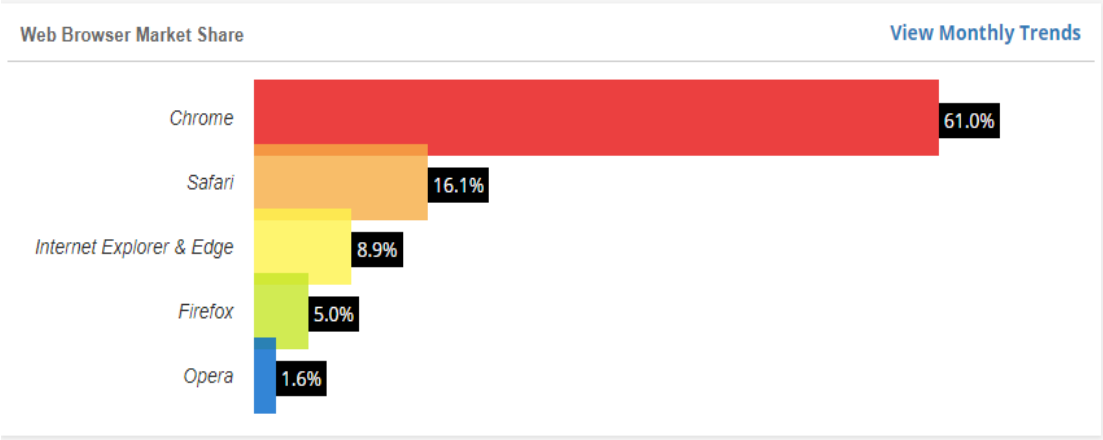

Figure 4.1: ranke of web browsers according to w3counter global stats[17].

4.3.2.Weka

Weka [18] is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

4.3.3.SerpWow API

SerpWow [19] is a robust, real-time API to return Google Search Results , it is easy to understand JSON & HTML output. SerpWow provides data from organic results, maps, news stories, images, videos, related searches, the knowledge graph and more. Location-specific results; with SerpWow you can target your results to specific locations or countries. You'll receive accurate geo-located results just like you would when searching in a browser [2].

We used SerpWowAPI to check the rank of the tested website. If the domain is in the first pages on the search engine. We use Google and Bing search results.

```
#google_engin
def google_Search_Engine(a):
  serpwow = GoogleSearchResults("074C72450A2947D2B25B8AD1A4900C05")
  params = {
    "q" : a,
    "time_period" : "last_year",
    "sort_by" : "highest_rating",
    "flatten_results" : "false"
    }
  result = serpwow.get_json(params)
  thislist= []
  if 'organic_results' not in result:
      return -1
  else:
      m=len(result['organic_results'])
      for i in range(m):
          s=json.dumps(result['organic_results'][i]['domain'])
          thislist.append(result['organic_results'][i]['domain'])
          nmb = thislist.count(a)
          if(nmb>3):
              return 1
          else:
              if(nmb>=1):
                  return 0
              else:
                  return -1
```

Figure 4.2: SperpWow Google search engine.

```
#bing_engin
def ping_Search_Engine(a):
  serpwow = BingSearchResults("074C72450A2947D2B25B8AD1A4900C05")
  params = {
    "q" : a,
    "engine" : "bing",
    "no_cache" : "true"
    }
  result = serpwow.get_json(params)
  thislist= []
  m=len(result['organic_results'])
  for i in range(m):
    s=json.dumps(result['organic_results'][i]['domain'])
    thislist.append(result['organic_results'][i]['domain'])
  #print(thislist)
  nmb = thislist.count(a)
  #re=print(nmb)
  if(nmb>3):
    return 1
  else:
    if(nmb>=1):
        return 0
    else:
      return -1
```

Figure 4.3: . SperpWow Bing search engine

4.3.4 Alexa rank

Alexa rank [20] is a measure of website popularity. It ranks millions of websites in order of popularity, with an Alexa Rank of 1 being the most popular. Alexa Rank reveals how a website is doing relative to all other sites, which makes it a great KPI for benchmarking and competitive analysis. Alexa rank is calculated using a proprietary methodology that combines a site's estimated traffic and visitor engagement over the past three months. Traffic and engagement are estimated from the browsing behavior of people in our global panel, which is a sample of all Internet users.

4.4.WebGuard Interface

WebGuard is an extension which work on the browser's background with a simple interface, as it is illustrated in the figures bellow:

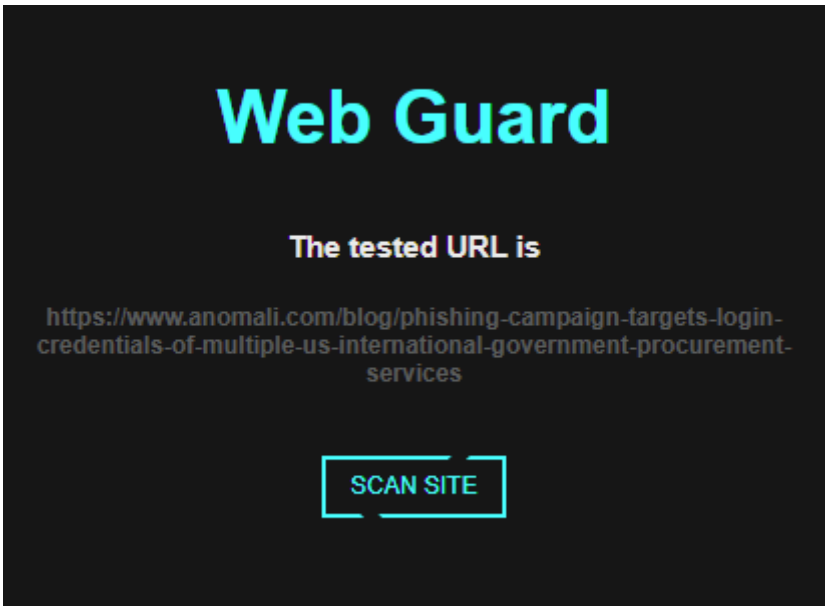

Figure 4.4:.General Interface of Web Guard.

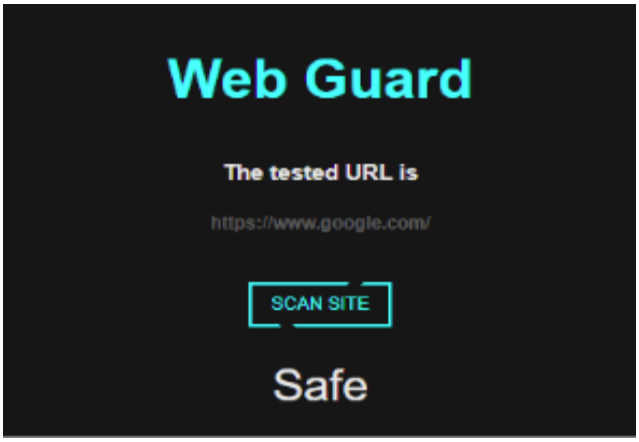

Figure 4.5: Alert when the website is safe

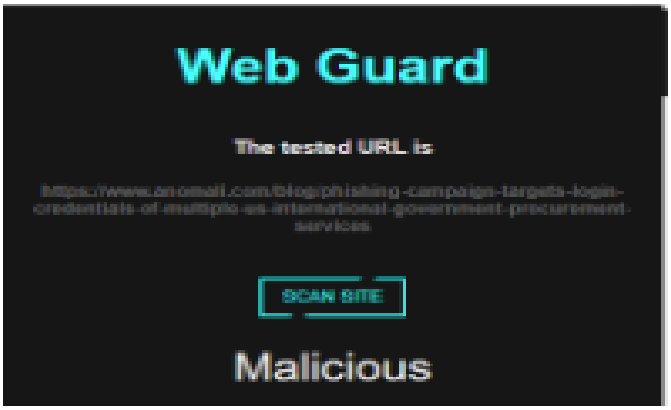

Figure 4.6: **Alert** when the website is malicious

4.5 Experimentation

In this section, we will measure the performance of our extension by testing the classifier with our generated dataset.

4.5.2 Experiment results

We used our generated dataset (contain 300 benign website and 300 malicious ones) to test the performance of our module using the choosed classifiers .

We used the method of Cross-validation 10Folds for our tests. Which is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it.

In k-fold cross-validation (we select K=10), the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once[36].

Figure 4.7 shows the result obtained by Weka. We obtained an accuracy of 99.5 % with 0.5% error ratio using Random Forest Tree.

While using Naïve Bayes, SVM and KNN reached less results shown in figure 4.8, figure 4.9, figure 4.10 and figure 4.10 respectively.

After the tests we conclude that Random Tree works better with a big dataset,with higher accuracy more than the other classifiers.

```
Time taken to build model: 0.06 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         597               99.5    %
Incorrectly Classified Instances         3                0.5    %
Kappa statistic                          0.99
Mean absolute error                      0.015
Root mean squared error                  0.072
Relative absolute error                  3.0043 %
Root relative squared error             14.391  %
Total Number of Instances              600

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,997    0,007    0,993      0,997    0,995      0,990    1,000     1,000     1
                 0,993    0,003    0,997      0,993    0,995      0,990    1,000     1,000     -1
Weighted Avg.    0,995    0,005    0,995      0,995    0,995      0,990    1,000     1,000

=== Confusion Matrix ===

   a   b   <-- classified as
 299   1 |   a = 1
   2 298 |   b = -1
```

Figure 4.7: Weka's Random forest Tree Resaults

```
Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         591               98.5    %
Incorrectly Classified Instances         9                1.5    %
Kappa statistic                          0.97
Mean absolute error                      0.0158
Root mean squared error                  0.1152
Relative absolute error                  3.1566 %
Root relative squared error             23.0459 %
Total Number of Instances              600

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,987    0,017    0,983      0,987    0,985      0,970    0,997     0,995     1
                 0,983    0,013    0,987      0,983    0,985      0,970    0,997     0,998     -1
Weighted Avg.    0,985    0,015    0,985      0,985    0,985      0,970    0,997     0,997

=== Confusion Matrix ===

   a   b   <-- classified as
 296   4 |   a = 1
   5 295 |   b = -1
```

Figure 4.8: Weka's Naïve Bayes Results

```
Time taken to build model: 0.25 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         593               98.8333 %
Incorrectly Classified Instances         7                1.1667 %
Kappa statistic                          0.9767
Mean absolute error                      0.0117
Root mean squared error                  0.108
Relative absolute error                  2.3333 %
Root relative squared error             21.6025 %
Total Number of Instances              600

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,993    0,017    0,983      0,993    0,988      0,977    0,988     0,980     1
                 0,983    0,007    0,993      0,983    0,988      0,977    0,988     0,985     -1
Weighted Avg.    0,988    0,012    0,988      0,988    0,988      0,977    0,988     0,983

=== Confusion Matrix ===

   a   b   <-- classified as
 298   2 |   a = 1
   5 295 |   b = -1
```

Figure 4.9: Weka's SVM Results

```
Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         591               98.5    %
Incorrectly Classified Instances         9                1.5    %
Kappa statistic                          0.97
Mean absolute error                      0.0142
Root mean squared error                  0.1127
Relative absolute error                  2.8319 %
Root relative squared error             22.5334 %
Total Number of Instances              600

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               1,000    0,030    0,971      1,000    0,985      0,970    0,995     0,983     1
               0,970    0,000    1,000      0,970    0,985      0,970    0,995     0,996     -1
Weighted Avg.  0,985    0,015    0,985      0,985    0,985      0,970    0,995     0,990

=== Confusion Matrix ===

   a   b   <-- classified as
 300   0 |   a = 1
   9 291 |   b = -1
```

Figure 4.10: Weka's KNN Results

4.5.3 Evaluation

There are three main parameters or criteria that show the performance of our classifier which are:

- Precision

Precision evaluates how precise a model is in predicting positive labels. Precision answers the question, out of the number of times a model predicted positive, how often was it correct? Precision is the percentage of your results which are relevant [22].

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Figure 4.11: Precision formula[21].

- Accuracy

Accuracy is an evaluation metric that allows you to measure the total number of predictions a model gets right.

Accuracy will answer the question, what percent of the models predictions were correct? Accuracy looks at True Positives and True Negatives [21].

$$Accuracy = \frac{TruePositive + TrueNegatives}{TruePositive + FaLsePositive + TrueNegative + FalseNegative}$$

Equation 4.1: Accuracy Formula.

- Recall

Recall calculates the percentage of actual positives a model correctly identified (True Positive). When the cost of a false negative is high, you should use recall[22].

$$Recall = \frac{TruePositive}{TruePositive + FaLseNegative}$$

Equation 4.2: Recall formula

In the following table 4.3 we present the scores obtained of the accuracy, recall and precision of our extension. Using four algorithms of classification which are the Random Tree, Naïve Bayes, SVM,KNN.

Classifier	Accuracy	Precision	Recall
Random Tree	99.5%	99.7%	99.3%
SVM	98.8%	99.3%	98.3%
KNN	98.5%	100%	97%
NB	98.5%	98.7%	98.3%

Table 4.3: Results metric of our extension (Accuracy, Precision , Recall).

4.5.4 comparison

The following charts in figure 4.11 figure 4.12 figure 4.13 shows comparison between our extension and the closest one from the previous works (Approach 1) using the common classifiers between them by testing the same dataset on Appproch1.

.

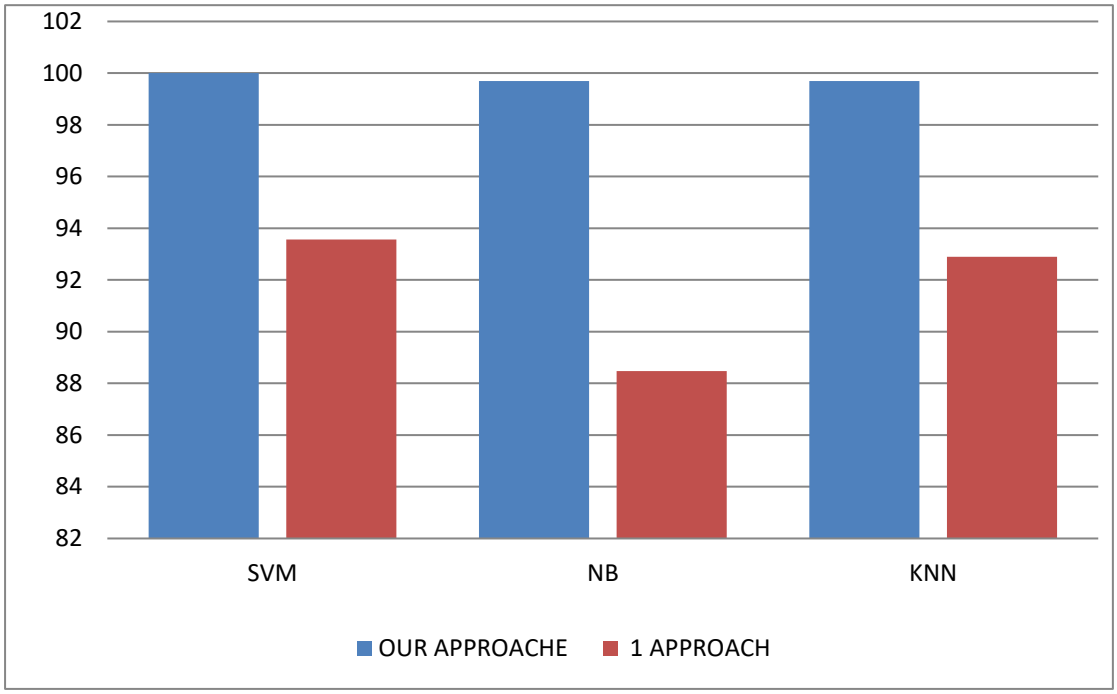

Figure 4.12: Accuracy comparison

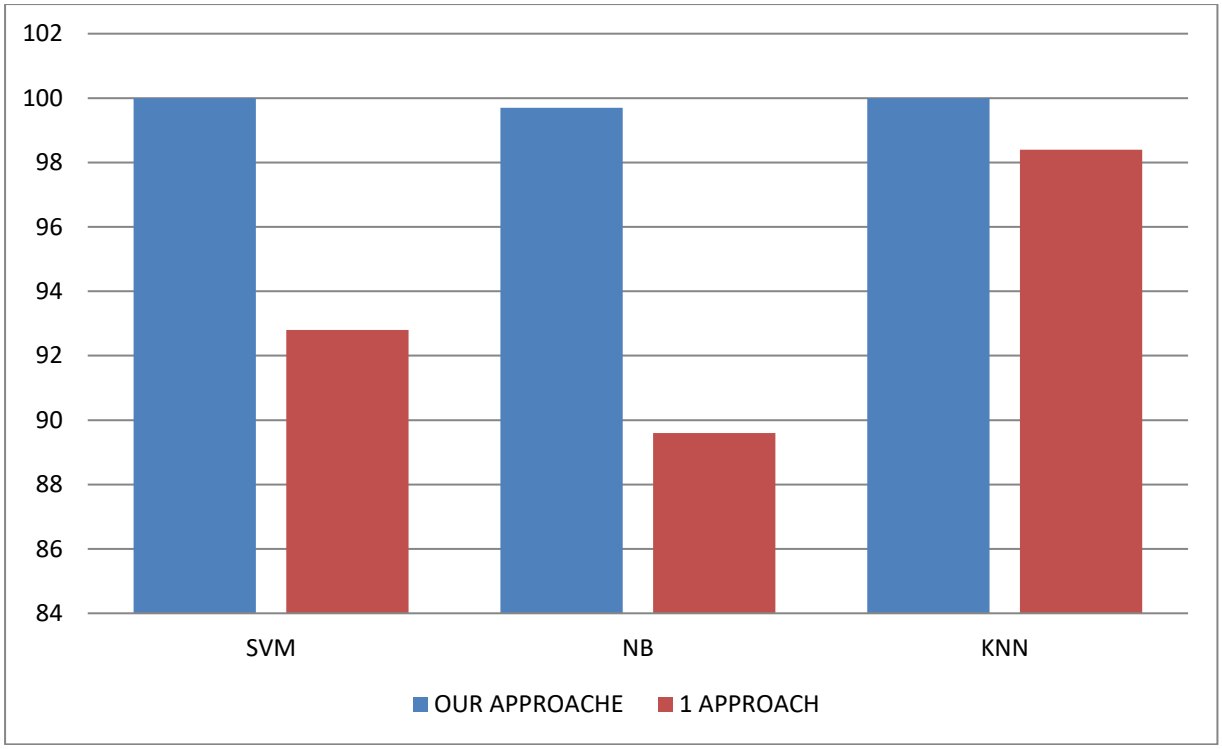

Figure 4.13: Precision comparison

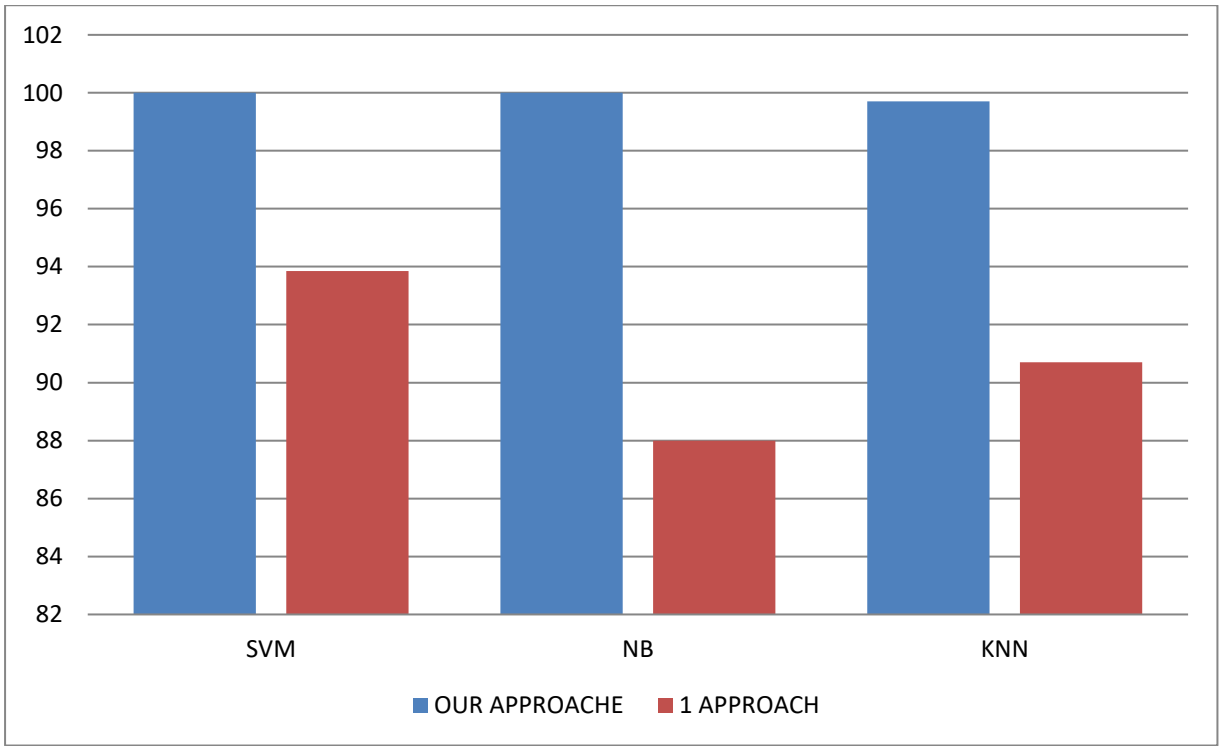

Figure 4.14: Recall comparison

4.6. Result Discussion

Our extension achieves perfect results on the test step. During the test our extension scored a high accuracy, precision and recall, these scores are result of the right selection of features set which improved this results and make our extension a powerful tool against the web ;malicious content.

4.7. Conclusion

In this chapter we present the implementation and experimentation of our WebGuard extension for detecting malicious web content. Our extension classification method based on a selected features set.

The experimental study clearly shows the efficiency of our extension in detection of malicious content, which proved by the high results obtained from classifiers.

GENERAL CONCLUSIN

General Conclusion

Due to the huge evolution of the computer science and the widely usage of the internet, malicious web content is spreading widely and quickly through the global websites which put the users data at high risk of losing or expose theme by different types of cyber-attacks. In our project we make a study at the existent approaches of detecting the malicious web content. the study shows that those approaches still incomplete and have several limits which led us to developed a Google chrome extension "Web Guard" to defeat the malicious content on the web by analyzing the requests and response from websites to detect the malicious web pages.

Our extension contributed through maintaining the following points:

- Almost new malicious features list that matches nowadays web content
- Pretended to detect the majority types of malicious content
- Analyze both request and web pages content
- Prevent the malicious content by blocking the malicious web pages
- Have a machine learning model to classify the websites into malicious and benign that is almost perfect.

In our project we have generated a dataset that includes malicious and benign websites which weren't' used in previous works.

During the experiments we tested our extension with the generated dataset (300 benign websites and 300 malicious ones)

The obtained experimental results demonstrate that our extension detects perfectly the majority of malicious websites during the test phase, which prove the effectiveness of our selected features list to detect malicious content .

Perspectives

Our approach focused on analyzing request more than response, it can be improved by focusing more at the analyzing of the response pages, and adding a database with user feedback to improve the process of classification.

REFERENCES

References

Articles

[7]Abubakr Sirageldin, Baharum B. Baharudin, and Low Tang Jung,Malicious Web Page Detection: A Machine Learning Approach,Computer & Information Science Department, University Technology Pertonas Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia,2014.

[8]Anand Desai,Janvi Jatakia,Rohit Naik and Nataasha Raul,Malicious Web Content Detection Using Machine Leaning,IEEE International Conference On Recent Trends in Electronics Information & Communication Technology (RTEICT), india,May 19-20, 2017.

[9]M. Aldwairi, R. Alsalman MALURLS: "A Lightweight Malicious Website Classification based on URL features, JOURNAL OF EMERGING TECHNOLOGIES IN WEB INTELLIGENCE, VOL. 4, NO. 2, MAY 2012

[10]Immadisetti Naga Venkata Durga Naveen, Manamohana K, Rohit Verma,Detection of Malicious URLs using Machine Learning Techniques,International Journal of Innovative Technology and Exploring Engineering (IJITEE),Volume-8 Issue-4S2 March, 2019

[11]M. Anantha Raman , R. Anil Kumar , S. Gowri Shankar , P. Deivendran ,Detecting Malicious Web pages in Real Time,International Journal of Innovative Research in Science, Engineering and Technology,india,Vol. 7, Special Issue 2, March 2018.

[12]DOYEN SAHOO, CHENGHAO LIU, STEVEN C.H. HOI, Malicious URL Detection using Machine Learning: A Survey, Vol. 1, No. 1, Article, August 2019.

[13]Elkan, C.: Nearest neighbor classification. University of California, San Diego (2007).

[14]Dr. Jitendra Agrawal, Dr. Shikha Agrawal, Anurag Awathe, Dr. Sanjeev Sharma,Malicious Web Page Detection through Classification Technique: A Survey, IJCST Vol. 8, Iss ue 1, Jan - March 2017.

References

Web sites

[1]https://www.webarxsecurity.com/website-hacking-statistics-2018-february/ visited on June 2020

[2]https://developer.chrome.com/extensions/overview visited on February 2020

[3]https://www.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.inter net.doc/topics/dfhtl_uricomp.html visited on mars 2020

[4]https://www.techopedia.com/definition/1352/uniform-resource-locator-url visited on Mars 2020

[5]https://www.techopedia.com/definition/4774/web-page-page visited on Mars 2020

[6]https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages visited on Mars 2020

[15]https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/ visited on August 2020

[16]https://www.javatpoint.com/machine-learning-random-forest-algorithm visited on August 2020

[17]https://www.w3counter.com/globalstats.php visited on July 2020

[18]https://www.analyticsvidhya.com/learning-paths-data-science-business-analytics-business-intelligence-big-data/weka-gui-learn-machine-learning/visited on July 2020

[19]https://www.programmableweb.com/api/serpwow visited on July 2020

[20]https://blog.alexa.com/marketing-research/alexa-rank/visited on August 2020

References

[21]https://lawtomated.com/accuracy-precision-recall-and-f1-scores-for-lawyers/ visited on August 2020

[22]https://medium.com/@erika.dauria/accuracy-recall-precision-80a5b6cbd28d visited on August 202

[23] https://filext.com/file-extension/PKL visited on September 2020.

[24] https://us.norton.com/internetsecurity-malware-what-are-malicious-websites.html visited on September 2020.

[25]https://www.wpbeginner.com/glossary/subdomain visited on September2020.

[26]https://www.webopedia.com/TERM/D/domain_name.html visited on September2020.

[27]https://zvelo.com/anatomy-of-full-path-url-hostname-protocol-path-more visited on September 2020.

[28]https://www.tutorialrepublic.com/html-tutorial/html-url.php visited on September 2020.

[29]https://branch.io/glossary/query-parameters/ visited on September 2020.

[30]https://developer.mozilla.org/enUS/docs/Learn/Common_questions/What_isaURL visited on September 2020.

[31]https://sucuri.net/guides/website-malware visited on September 2020.

[32]https://www.malcare.com/blog/seo-spam/ visited on September 2020.

[33]https://heimdalsecurity.com/blog/javascript-malware-explained visited on September 2020.

[34]https://www.phishing.org/what-is-phishing visited on September 2020 .

[35]https://www.imperva.com/learn/application-security/backdoor-shell-attack/visited on September 2020.

[36] https://www.openml.org/a/estimation-procedures/7 visited on September 2020

Abstract

Malicious content on the web became a global risk at web users, its huge spreading made users vulnerable to all types of cyber attacks that can performed behind websites. Many recent researches were proposed to detect malicious content.

in this work we propose a chrome extension to defeat the majority types of malicious web content by analyzing HTTP requests and responses. The extension is based on a relevant list of malicious features geared with a machine learning classifier also it integrated Google safe browsing for more protection. The obtained experimental results demonstrate the effectiveness of our extension on detecting the malicious content which proved by the perfect result scored on several classifiers with 100% accuracy

Keywords: Web Browser Extension, Malicious content. Machine learning, HTTP request and response analysis.

ملخص

أصبح المحتوى الضار على الويب خطرًا عالميًا على مستخدمي الويب ، حيث أدى انتشاره الضخم إلى جعل المستخدمين عرضة لجميع أنواع الهجمات الإلكترونية التي يمكن أن تتم خلف مواقع الويب. تم استخدام العديد من الأساليب والأبحاث الحديثة واكتشاف المحتوى الضار. وفي هذا اقترحنا امتداد كروم لهزيمة جميع أنواع محتوى الويب الضار من خلال تحليل طلبات و ردود الواب ، ويستند الامتداد إلى قائمة من ميزات المواقع الضارة الموجهة مع مصنف التعلم الآلي أيضًا ، قام بدمج التصفح الآمن من غوغل لمزيد من الحماية. أظهرت النتائج التجريبية التي تم الحصول عليها فاعلية امتدادنا في الكشف عن المحتوى الضار والذي ثبت بالنتيجة المثالية المسجلة على العديد من المصنفات بدقة 99.5٪.

الكلمات المفتاحية :امتداد متصفح الويب ، المحتوى الضار، التعلم الآلي ، طلب HTTP وتحليل الاستجابة.