A Mobile Application to avoid Online Deception

COMP4801 – Final Report

FYP21013
Lai Wai Yuet (3035568453) (Author)
Lai Filbert Gibson (3035568013)

Supervisor: Dr. Chim, Tat Wing

18th April 2022
Abstract

The use of internet is becoming more and more common in the 21st century. However, the safety risk of internet increase with the growth of the internet use. The number of cybercrimes has an increasing trend, which leads to monetary loss and leakage of personal information. This paper introduces a mobile application project that aims to provide users protection on the internet environment and increase their awareness of cybersecurity. This project serves as a free solution to cybersecurity for the public. Three main functions are provided by the application. The first function is to block the access towards reported phishing websites using Google Safe Browsing Service and APIs. The second function is detecting and blocking access to unknown phishing websites by APIs and machine learning model. Experiments and justification were conducted to train a model with the best performance. The last function is to provide the latest information on cybersecurity to users to help users get updated on how to protect themselves.
Acknowledgement

We would like to express our great appreciation to the project supervisor, Dr. Chim, Tat Wing, for his valuable suggestions and guidance on this project.
# Table of Content

1 Introduction .......................................................................................................................... 10  
1.1 Background ......................................................................................................................... 10  
1.2 Objectives ........................................................................................................................... 10  
1.3 Motivations .......................................................................................................................... 11  
1.4 Scope .................................................................................................................................. 12  
1.5 Report Outline ...................................................................................................................... 12  

2 Methodology .......................................................................................................................... 13  
2.1 System design ...................................................................................................................... 13  
2.1.1 Database ........................................................................................................................... 14  
2.2 Blocking reported phishing websites .................................................................................... 14  
2.2.1 External Services ................................................................................................................. 14  
2.2.2 Self-defined Blocklist .......................................................................................................... 15  
2.3 Scanning unknown phishing websites .................................................................................... 16  
2.3.1 APIs .................................................................................................................................. 16  
2.3.2 Machine Learning model .................................................................................................... 17  
2.4 Information regarding cybersecurity ....................................................................................... 18  

3 Experiments and Results ......................................................................................................... 19  
3.1 HTTP Server ......................................................................................................................... 19  
3.2 Database ............................................................................................................................... 20  
3.3 Function1: Blocking Reported Phishing Websites ................................................................. 21  
3.3.1 External Services ............................................................................................................... 21  
3.3.2 User-Defined Blocklist ....................................................................................................... 23  
3.4 Function 2: Blocking Suspicious Websites ............................................................................ 26  
3.4.1 External Services ............................................................................................................... 27  
3.4.2 Machine Learning Model ................................................................................................... 28  
3.5 Function3: Information on Cybersecurity ............................................................................. 30  
3.5.1 News on Cybersecurity ....................................................................................................... 30  
3.5.2 Practice on prevent Online Frauds ..................................................................................... 32  
3.5.3 Quiz on Cybersecurity Knowledge ..................................................................................... 32  
3.6 Machine Learning Model ..................................................................................................... 33  
3.6.1 Training Dataset ............................................................................................................... 34  
3.6.2 Testing Dataset ................................................................................................................... 34  
3.6.3 Model Training .................................................................................................................. 35
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.3.1</td>
<td>Pricing of different Solutions of BlockSite</td>
<td>P. 11</td>
</tr>
<tr>
<td>Figure 2.1.1</td>
<td>Overall system design</td>
<td>P. 13</td>
</tr>
<tr>
<td>Figure 2.3.1</td>
<td>Security checking performed by URLVoid</td>
<td>P. 16</td>
</tr>
<tr>
<td>Figure 3.1.1</td>
<td>Workflow of the HTTP server</td>
<td>P. 19</td>
</tr>
<tr>
<td>Figure 3.2.1</td>
<td>Workflow of checking against reported phishing websites</td>
<td>P. 21</td>
</tr>
<tr>
<td>Figure 3.2.2</td>
<td>Screenshots of blocking reported phishing website by Google safe Browsing</td>
<td>P. 22</td>
</tr>
<tr>
<td>Figure 3.2.3</td>
<td>Screenshots of blocking reported phishing website by APIs</td>
<td>P. 23</td>
</tr>
<tr>
<td>Figure 3.2.4</td>
<td>Screenshot of blocklist page (left) and deleting item (right)</td>
<td>P. 24</td>
</tr>
<tr>
<td>Figure 3.2.5</td>
<td>Screenshot of blocking a website in Browser Page</td>
<td>P. 25</td>
</tr>
<tr>
<td>Figure 3.2.6</td>
<td>Screenshots of blocking website by blocklist</td>
<td>P. 26</td>
</tr>
<tr>
<td>Figure 3.3.1</td>
<td>Workflow of detecting potential fraud websites</td>
<td>P. 26</td>
</tr>
<tr>
<td>Figure 3.3.2</td>
<td>Screenshot of blocking suspicious websites by APIs</td>
<td>P. 27</td>
</tr>
<tr>
<td>Figure 3.3.3</td>
<td>Screenshot of blocking a suspicious website by machine learning model</td>
<td>P. 28</td>
</tr>
<tr>
<td>Figure 3.3.4</td>
<td>Screenshot of feedback example</td>
<td>P. 29</td>
</tr>
<tr>
<td>Figure 3.3.5</td>
<td>Workflow of feedback system</td>
<td>P. 30</td>
</tr>
<tr>
<td>Figure 3.5.1</td>
<td>Workflow of the news sharing function</td>
<td>P. 31</td>
</tr>
<tr>
<td>Figure 3.5.2</td>
<td>Screenshots of news page</td>
<td>P. 31</td>
</tr>
<tr>
<td>Figure 3.5.3</td>
<td>Screenshot of the info page</td>
<td>P. 32</td>
</tr>
<tr>
<td>Figure 3.5.4</td>
<td>Screenshots on the Quiz page</td>
<td>P. 33</td>
</tr>
<tr>
<td>Figure 3.6.1</td>
<td>Dividing a URL into different substrings</td>
<td>P. 34</td>
</tr>
<tr>
<td>Figure 3.6.2</td>
<td>Performance of XGBoost on testing dataset</td>
<td>P. 36</td>
</tr>
<tr>
<td>Figure 3.6.3</td>
<td>Feature importance measured in MDI of training dataset</td>
<td>P. 38</td>
</tr>
<tr>
<td>Figure 3.6.4</td>
<td>Feature importance measured in MDI after feature selection</td>
<td>P. 39</td>
</tr>
<tr>
<td>Figure 3.6.5</td>
<td>Performance on selected features in testing dataset</td>
<td>P. 39</td>
</tr>
<tr>
<td>Figure 3.6.6</td>
<td>Feature importance measured in MDI of testing dataset</td>
<td>P. 40</td>
</tr>
<tr>
<td>Figure 3.6.7</td>
<td>Performance of the final model</td>
<td>P. 42</td>
</tr>
<tr>
<td>Figure 4.1.1</td>
<td>General user interface design</td>
<td>P. 43</td>
</tr>
<tr>
<td>Figure 4.2.1</td>
<td>Screenshot of the browser page</td>
<td>P. 44</td>
</tr>
<tr>
<td>Figure 4.2.2</td>
<td>Screenshot of loading URL outside</td>
<td>P. 45</td>
</tr>
<tr>
<td>Figure 4.3.1</td>
<td>Screenshot of the setting page</td>
<td>P. 46</td>
</tr>
<tr>
<td>Figure 4.3.2</td>
<td>Block a website with two approaches</td>
<td>P. 46</td>
</tr>
<tr>
<td>Figure 4.4.1</td>
<td>Screenshot of the help page</td>
<td>P. 46</td>
</tr>
<tr>
<td>Figure 4.5.1</td>
<td>Examples on self-descriptive component</td>
<td>P. 48</td>
</tr>
<tr>
<td>Figure 4.5.2</td>
<td>Screenshot of a toast message</td>
<td>P. 49</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Feature importance on the UCI dataset</td>
<td>P. 50</td>
</tr>
</tbody>
</table>
List of Tables

Table 1  The structure of the ‘user_feedback’ collection  P. 20
Table 2  The structure of the ‘news’ collection  P. 20
Table 3  Performance of the three models on the validation set  P. 35
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Hypertext Transfer Protocol Secure</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>MDI</td>
<td>Mean Decrease in Impurity</td>
</tr>
<tr>
<td>SVM</td>
<td>Support vector machine</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>UX</td>
<td>User Experience</td>
</tr>
<tr>
<td>WSGI</td>
<td>Web Server Gateway Interface</td>
</tr>
<tr>
<td>XGBoost</td>
<td>Extreme Gradient Boosting</td>
</tr>
</tbody>
</table>
1 Introduction

This section first introduces the background, followed by the objectives and motivation of the project. The scope of the project will then be discussed. This section will end by covering the outline of the report.

1.1 Background

Information technology is widely used around the globe in the 21st Century, the development and application of the World Wide Web grows rapidly. There is a large amount of internet users worldwide. According to the statistic provided by Statista, the number of internet users was estimated 4.9 billion and has an increasing trend since 2005 [1]. Meanwhile, cybercrime grows with the expand of internet use. Online frauds are becoming more and more frequent under the development of internet. Online frauds cause a large amount of monetary loss and personal data breaches. In Hong Kong, more than 10500 internet deception cases were reported to the police forces, which caused near 3000 million HKD loss in 2020 [2]. The number of reported deception cases in 2019 was about 5000, the increase from 2019 to 2020 is more than 100% [2]. This showed an increasing trend in the number of online frauds. There is an urgent need to provide a secure and reliable network environment.

1.2 Objectives

Two objectives are included in this project. The first objective is to provide the community with a safer cyber environment. This objective is achieved by filtering the fraud websites and detecting phishing websites. The access towards suspicious or phishing websites will be blocked, so that the public will not be harmed by these websites. The second objective is to raise the awareness of online deception and the sense of cybersecurity of the public, so that the public can learn how to protect themselves from cybercrimes. This is achieved by providing knowledge and the latest information regarding cyber pitfalls.
1.3 Motivations

The motivation of this project is to provide users with a secure cyber environment and raise the awareness of information security of the public. In the mobile application environment, there are applications like ‘BlockSite’ on Google Play Store which aims to provide users with a safer internet environment [3]. However, similar to many other similar applications, not all the functions are available freely, and users are required to pay subscription fees in order to unlock all the protection features [4]. Figure 1.3.1 shows the subscription plan of BlockSite. Many features, including unlimited blocking list, custom block pages and password protections, are only available in the ‘Unlimited Plan’, which requires subscription fee. Hence, our project aims to build a mobile application that provides similar functions to prevent online fraud while without charging the users. The application treats as a free solution against cybercrimes, so that the coverage of users is wide, while a full protection is provided at the same time.

![Figure 1.3.1: Pricing of different Solutions of BlockSite [4]](image-url)
1.4 Scope

Our team introduces Online Guard, an android application that aims to provide users cyber protections and increase their awareness. Android Studio is used as the application development IDE. The application is free to download with all the functions provided freely. The objective of the project is achieved by three main functions in Online Guard. First, the application blocks the access towards reported phishing websites whenever users try to access them. Second, the application detects unknown phishing websites, and users will be warned if they try to access them. Different technologies, including APIs and machine learning models, are used to detect suspicious websites. Lastly, the application provides users latest news and information regarding cyber security in order to help users avoid online deception.

1.5 Report Outline

The remaining of the report will start with discussing the methodology of the project in Section 2. The methodology section will first discuss the overall system design and the choice of the database in section 2.1. The three main functions will then be covered in section 2.2 to section 2.4. The experiment and implementation results will be introduced in section 3. The setup of the HTTP server and the database server will first be introduced in section 3.1 and section 3.2. Followed by the implementation details of the three functions in section 3.3 to 3.5. Section 3 will end with the discussion of the experiments performed and the results obtained on the machine learning model in section 3.6. Section 4 will discuss the application design, which starts with the main design of the application in section 4.1. The layout of the function pages will then be covered, including the browser page in section 4.2 and the setting page in section 4.3. Section 4 will end with the discussion on the UX design. Lastly, this paper will introduce the limitation of the project in section 5 and a conclusion will be given in section 6.
2 Methodology

This section discusses the methodologies and justifications of the project. This section first introduces the overall system design, followed by introducing the three main functions in the project. The working mechanism, methodologies, and design choice will be covered for each function.

2.1 System design

This project is constructed by three main components, including the mobile application, the development server, and the database.

![Overall system design](image)

**Figure 2.1.1 Overall system design**

Figure 2.1.1 shows the overall system design of the project. The mobile application contains different pages with different functions, including (1) Browser page which is a browser providing a safe internet browsing environment, (2) Info page which provides users useful tips to avoid online deception and cybersecurity information, (3) News page which provides the latest news regarding cybersecurity, (4) Quiz page which tests users’ understanding on knowledge of Info page, (5) Blocklist page which contains a list of URLs and domain names defined by users locally.

The development server contains (1) an HTTP server that handles the HTTP requests from the mobile app, (2) a machine learning model which is used to detect unknown phishing
websites, (3) a web crawling program that is used to crawl the latest cybersecurity news. A database server is set up to store the news crawled and users’ feedback on the machine learning model.

2.1.1 Database

MongoDB is chosen to build the database. In this project, the database is used to store the feedback on machine learning model prediction and the news crawled. There are no complex relationships between data, and each datum is served as an individual document. Therefore, a non-relational database is chosen as the database of the project. A non-relational database is suitable for simple data structure, using a key-value database allows fast reading and writing [5]. Compared with a relational database, NoSQL database generally has a better performance on the processing and querying speed when it deals with simple queries and structures [6]. It is more capable with a large amount of data [6]. It also allows the database to scale the database horizontally when needed.

2.2 Blocking reported phishing websites

Blocking reported phishing websites is one of the main functions in the mobile application. The application should be able to protect users from known phishing websites and the access towards reported phishing websites will be blocked. The application provides protection to users through the web browsing function in the browser page. The URL of the website user tries to access in the browser page will be checked with multiple technologies. As shown in figure 2.1, several malicious URL detectors are used to detect reported phishing websites. Users are also allowed to define a local blocklist that contains domain names and URLs. The URL user tries to access will first be checked by the APIs and the blocklist. If the website is detected as phishing or it is contained in the blocklist, the access towards to the website will be blocked.

2.2.1 External Services

The number of reported websites is very large. There exist databases maintained by trusted organisations that store the reported phishing websites. Services are also provided from these
organisations to help people check URLs against the databases. One can use the service through different ways, including using APIs. This project uses one service and two APIs, which are Google safe browsing service, URLVoid and IP Quality Score to check URLs against the databases. Google safe browsing service is used to detect reported phishing websites. Google safe browsing service provides services to check against a constantly updated list maintained by Google, which contains reported unsafe web resources like phishing and deceptive sites [7]. Google safe Service is available free and publicly in applications and browsers. The access to the website will be blocked if the website has been reported as phishing to Google before.

Two APIs from two platforms, including URLVoid and IP Quality Score, are also used to detect reported phishing websites. The first API, ‘URL Reputation API’ from the platform ‘URLVoid’, checks the URL against more than 25 engines such as OpenPhish, PhishTank and Phishstats [8]. The API returns the risk score, the number of engines that gives a phishing detection result. The second API, ‘Malicious URL Scanner API’ from the platform ‘IP Quality Score’, is able to detect and identify phishing, malware URLs and parked domains over more than 20 datapoints [9]. The API returns the risk score and the website category. If the risk scores returned by the two APIs show the website is a phishing URL, the access to the website will be blocked.

2.2.2 Self-defined Blocklist
Apart from the reported phishing websites stored in databases maintained by different parties, there exist zero-day attacks where the phishing website was not reported before. The blocklist serves as a security patch. If users find a phishing website that has not been reported and Online Guard cannot detect it as phishing, users can add the domain or URL to the blocklist, so that the website will be blocked. Users will no longer be able to visit the websites in the blocklist and users are protected from fraud websites. The blocklist is set to be a local blocklist, where the blocklist is stored in the local device instead of the development server. This is to prevent users maliciously reporting or adding safe websites into the blocklist. If all users share the same blocklist, they will be easily affected by the malicious reports.
2.3 Scanning unknown phishing websites

The second main function of the mobile application is to detect unknown phishing websites. The application should be able to protect users from zero-day vulnerability by detecting unknown phishing websites. The protection against unknown phishing websites is provided to users through the browsing page. Similar to the first function, the URL of the website users try to access in the browsing page will be first checked with multiple technologies. The URL will be verified by APIs and machine learning models. The access towards the website will be blocked if the website is detected as a potential phishing website.

2.3.1 APIs

There exist services where an URL can be checked whether is it an unknown attack. The APIs mentioned in section 2.2 will also be used to scan suspicious websites. The first API, ‘URL Reputation API’ from the platform ‘URLVoid’, detect potential phishing websites by using thousands of smart internal rules [8]. The URL passed to the API will undergo deep analysis, including URL content, domain name, HTTP headers and pattern.

```
"risk_score":{
  "result":100
},
"security_checks":{
  "is_host_an_ipv4":false,
  "is_suspicious_url_pattern":false,
  "is_suspicious_file_extension":false,
  "is_suspended_page":true,
  "is_most_abused_tld":false,
  "is_uncommon_clickable_url":true,
  "is_phishing_heuristic":true,
  "is_suspicious_content":false,
  "is_empty_page_title":false,
  "is_domain_blacklisted":true,
  "is_suspicious_domain":false,
  "is_sinkholed_domain":false,
  "is_defaced_heuristic":false,
  "is_risky_geo_location":false,
}
```

Figure 2.3.1 Security checking performed by URLVoid [8]
Figure 2.3.1 shows part of the security checks performed by URL Void API. The URL pattern and the contents of the websites are checked by the API, and a risk score which ranges from 0 to 100 is returned. A risk score of 100 means the website is very likely to be phishing, while 0 means the URL is expected to be safe. The second API, ‘Malicious URL Scanner API’ from the platform ‘IP Quality Score’, detects zero-day phishing link in real time by using live threat intelligence [9]. Different approaches are used by the API, including Fraud Fusion Network and machine learning model. The API will return a risk score.

The URL will be passed to the two APIs. A warning will be given if the risk score returned shows the URL is detected as potentially phishing, the access towards the website will be blocked.

### 2.3.2 Machine Learning model

Machine learning model is also used to detect phishing websites. The model is developed by our team through experiments. The URL will be passed to the server when users access an URL. Features are extracted from the URL and the website, then are inputted into the model to perform prediction. The model will return the classification result and return the result to the application through the server. A warning will be given to users if the website is detected as phishing and the access towards the website will be blocked. Users are able to provide feedback on the prediction of the model.

Experiments were conducted to choose the model with the best performance. Multiple approaches and methods are used in different trials. In order to further enhance the performance of the model in the future, user feedback is collected and stored in the Database. The model undergoes incremental training by continuously fitting on the feedback. Therefore, only architectures that are able to perform continual learning are considered. The performance of the model was evaluated on data from the real-world environment to test how it can detect unknown phishing websites in practice.
2.4 Information regarding cybersecurity

The last main function of the project is to provide the latest information regarding cybersecurity. This project aims to enhance users’ sense of cybersecurity, so that they are able to protect themselves from online frauds. This function consists of three sub-functions, including news sharing, information sharing and quizzes on cybersecurity knowledge.

The news sharing function provides users with updated cybersecurity news from different sources. There are two sources of news, including ‘The Hacker News’ and ‘Cyware Social’. The Hacker News is a cybersecurity publication and provides the latest cybersecurity news and in-depth reports regarding information security [10]. Cyware Social provide the latest security articles which aim at keeping the public up to date on the security threat landscape. Users are able to get updated with the latest attacks and cybersecurity threats. The information sharing function provides users common tips and practices to avoid online deception, and the knowledge learnt can be tested with the quizzes function.
3 Experiments and Results

This section discusses the experiments performed and the implementation details of the project. This section first introduces the implementation of the HTTP server and the database, followed by the design and implementation of the main functions. For each function, the workflow and justification will be introduced. This section will then discuss the machine learning model implemented, where the experiments performed, and the evaluation of the results will be covered.

3.1 HTTP Server

An HTTP server was set up to handle the request from the mobile applications and query the database. The mobile application will not query the database directly. Instead, the queries are performed by the HTTP server. The server was set up using python library Flask. Flask is a light weight WSGI web application framework [11]. It allows the development of a simple server quickly while keeping the ability to scale up in the future.

---

Figure 3.1.1 Workflow of the HTTP server

Figure 3.1.1 shows the workflow of the HTTP server. The HTTP server handles three main GET requests from the mobile application. First, it handles requests for scanning a URL using the machine learning model and the server returns the prediction result in a response. Second, it handles requests for user feedback on the machine learning model, it stores the feedback in the database. The last request it handles is to provide news regarding cybersecurity by querying the news crawled from the database.
3.2 Database

A database is built using MongoDB. The database is used to store the user feedback on the machine learning model and the news crawled by the crawler program. There are totally two collections in the database, which are ‘user_feedback’ and ‘news’.

Table 1 The structure of the ‘user_feedback’ collection

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
<th>Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID of the document</td>
<td>ID object</td>
</tr>
<tr>
<td></td>
<td>Assigned by the database</td>
<td></td>
</tr>
<tr>
<td>url</td>
<td>The URL of the website</td>
<td>String</td>
</tr>
<tr>
<td>label</td>
<td>True label of the URL</td>
<td>String</td>
</tr>
<tr>
<td>features</td>
<td>Features extracted from the URL</td>
<td>List of floats and integers</td>
</tr>
</tbody>
</table>

Table 3.2.1 shows the data contained in the ‘feedback’ collection. This collection stores the user feedback on the machine learning model, which will be used to train the model. The collection contains a unique ID object, the URL, the true label, and the feature extracted for each document.

Table 2 The structure of the ‘news’ collection

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
<th>Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID of the document</td>
<td>ID object</td>
</tr>
<tr>
<td></td>
<td>Assigned by the database</td>
<td></td>
</tr>
<tr>
<td>url</td>
<td>The URL of the news</td>
<td>String</td>
</tr>
<tr>
<td>title</td>
<td>The title of the news</td>
<td>String</td>
</tr>
<tr>
<td>source</td>
<td>The source of the news</td>
<td>String</td>
</tr>
</tbody>
</table>

Table 3.2.2 shows the data contained in the ‘news’ collection. This collection stores the news crawled. The collection contains a unique ID object, the URL, the title, and the source of the news for each document.
3.3 Function1: Blocking Reported Phishing Websites

The access towards reported phishing websites is blocked in the browser page of the application. The URL user tries to access is passed to several components to check against.

![Diagram](image)

Figure 3.2.1 Workflow of checking against reported phishing websites

Figure 3.2.1 shows the workflow when checking an URL against reported phishing websites. The URL will be checked with the user-defined blocklist stored in the application. It will also be checked with external malicious URL detecting services, including the Google Safe Browsing service and the two APIs. If any of the returned results is positive, the access towards the website will be blocked.

3.3.1 External Services

External services are used to check an URL against large databases or lists of reported phishing websites maintained by different organisations. Figure 3.2.2 shows the screenshot of the access towards a reported phishing website being blocked by Google safe browsing service. The browser page in the application consists of a WebView component. Google safe browsing is configured as enable for internet surfing. As shown in Figure 3.2.2, users will be warned by a red page with a warning message will be shown if a dangerous website is being accessed or a harmful app is being downloaded. The reason for blocking the website is shown in the detail page. Users can choose to continue to access the website by clicking the ‘visit the unsafe website’ button.
Two APIs are also used to block the access towards reported deception websites. Figure 3.2.3 shows the screenshots of blocking reported phishing websites using the two APIs. If the website is detected as phishing by the ‘URL Reputation API’ from the platform URLVoid, a pop-up window with the warning message will be shown. The warning message includes the reason for blocking, the risk score returned by URLVoid, the number of engines that returns a ‘phishing’ result and the title of the website. Users will be forced to return to the previous page. Users can choose to continue to access the website by clicking the ‘still go’ button in the pop-up window.

The second API used is the ‘Malicious URL Scanner API’ from the platform ‘IP Quality Score’. If the website is identified as a phishing by this API, a pop-up window containing the warning message will be shown. The message will contain the reason for blocking, the risk score returned by the IP Quality Score, the reason for blocking the website given by the API and the website category. Users will be forced to return to the previous page. Users can choose to continue to access the website by clicking the ‘still go’ button in the pop-up window.
3.3.2 User-Defined Blocklist

Users may find the website they are visiting is actually a zero-day phishing website that is not reported. The user-defined blocklist serves as a security patch. Users are able to define a local blocklist to block the access towards specific websites. The blocklist is stored locally on each device and it will not be shared with other users. The blocklist consists of complete URLs and domain names. Users can check the content of the blocklist by clicking the circle icon located at the top right corner in the top action bar. Figure 3.2.4 shows the screenshot of the blocklist page. Users can add items to the blocklist by typing the URL or the domain name in the text field, and then click the add button. In the figure, the blocklist contains a URL ‘https://www.cuhk.edu.hk/English/index.html’, and a domain ‘youtube’. Users can delete the item by tapping the item. There will be a pop-up window, the item can be deleted by clicking the ‘confirm’ button. A toast message will be shown upon successful editing, and the blocklist will be updated immediately.
Users are also able to add a website into the blocklist when they find the website they are accessing is actually a phishing website. Figure 3.2.5 shows a screenshot when adding the current website in the browser page to the blocklist. Users can click the red cross button located at the top right corner of the browser page. A pop-up window will be shown, which asks for confirmation from users. By clicking the ‘ADD’ button, the current URL will be added into the blocklist. The application will be updated immediately, the website will become inaccessible. The browser page will return to the previous page. A toast message will be shown upon successful adding. Users are able to provide feedback to the machine learning model after adding, which will be discussed in section 3.4.2.
Figure 3.2.5 Screenshot of blocking a website in Browser Page

The access towards a website will be blocked if there is an exact match with any URL in the blocklist, or the domain name is contained in the blocklist. Figure 3.2.6 shows a screenshot of blocking an access towards a website where its URL or domain is contained inside the blocklist. A pop-up window containing the warning message will be shown. The reason of why the website is being blocked is shown. Users can choose to access the blocked website by clicking the ‘still go’ button in the pop-up window or return to the previous website by clicking the ‘cancel’ button.
3.4 Function 2: Blocking Suspicious Websites

The access towards suspicious or zero-day phishing websites will be blocked in the browser page. Several methods are used to detect potential phishing websites.

![Workflow of detecting potential fraud websites](image)

Figure 3.3.1 Workflow of detecting potential fraud websites

Figure 3.3.1 shows the workflow to detect suspicious phishing websites. The URL of the website will be checked with the external malicious URL detector, including the two APIs. The URL will also be checked with the machine learning model in the development server. If any of the returned results is positive, the access will be blocked.
3.4.1 External Services

The two APIs mentioned in section 3.2.1 are also used for detecting suspicious websites. Similar to the previous function, a pop-up window with a warning message will block the access to the websites which are detected as phishing.

Figure 3.3.2 Screenshot of blocking suspicious websites by URL Void (right), IP Quality Score (left)

Figure 3.3.2 shows the screenshots when a website is blocked by the two APIs. If the website is blocked by the URLVoid API, the reason, risk score, the number of engines returned a phishing result, and the website title will be shown in the pop-up window. If the website is blocked by IP Quality Score, the reason, the risk score, the reasons given by the API, and the website category will be contained in the pop-up window. The browser page will return to the previous page if users click the ‘cancel’ button in the pop-up window. Users can also choose to access the suspicious website by clicking the ‘still go’ button.
3.4.2 Machine Learning Model

Our team trained a machine learning model to detect zero-day phishing websites. The model is stored in the development server and the URL will be passed to the machine learning model to perform prediction. The URL will be sent to the HTTP server by GET request. Upon receiving the URL, the HTTP server will process the URL and extract the attribute from the URL. During the process of attribute extraction, if the URL is found to be a shortened URL, or it will redirect users to another URL, the final URL address will be chosen as the URL to be detected. The extracted attributes will be fed into the machine learning model. The model will perform a binary classification and return the classification result. The result will be contained in the body of the HTTP response from the server. If the returned result is phishing, the access towards the website will be blocked.

Figure 3.3.3 Screenshot of blocking a suspicious website by machine learning model

Figure 3.3.3 shows the screenshot of a suspicious website being blocked by the machine learning model. A pop-up window with the warning message will block the user’s access. The warning message contains the reason for blocking the website. Users can choose to continue to access the website by clicking the ‘still go’ button or return to the previous page by clicking the ‘cancel’ button.
As stated in section 3.6, XGBoost is chosen as the architecture of the model and XGBoost supports incremental learning. A trained model can be loaded and perform continual training on the model [12]. The model is able to perform online learning and can be trained on stream of data. The model improves its performance and learns knowledge by fitting on new data. Feedback on the model is collected from users to help the model provide more accurate predictions. Users are able to provide feedback to the model if the prediction of the model is incorrect. When users click the red cross button at the top right corner to block the current website, or press the ‘still go’ button in the pop-up window of machine learning prediction, a snack bar will appear at the bottom, which asks for user feedback.

Figure 3.3.4 Screenshot of the model improvement request (right) and the feedback window (left)

Figure 3.3.4 shows the message shown at the bottom of the app, asking the user to help improving the model. If the user clicks the ‘REPORT’ button, a pop-up window will be shown as shown in Figure 3.3.4. Users can choose the label of the website, which is ‘safe’ or ‘phishing’.
Figure 3.3.5 shows the workflow of receiving user feedback. The feedback, which is the URL and the label, will first be sent to the HTTP server. Then the attributes of the URL will be extracted. Finally, the URL, extracted features and the label of the URL will be stored into the database. The fitting of the model is performed regularly. A scheduled program will run automatically every a fixed period, which first queries the feedback from the database, then it will count the number of feedbacks. If it exceeds the threshold, the features and the labels will be fitted into the model. The updated model will replace the old model, and the feedback used will be deleted from the database.

3.5 Function3: Information on Cybersecurity

The application provides the latest information regarding cybersecurity and useful practice for users to protect themselves from cyber frauds. This function consists of three parts, which are news sharing, providing tips for avoiding online deception and quizzes to access users’ knowledge on cybersecurity.

3.5.1 News on Cybersecurity

The application provides the latest news regarding cybersecurity in the news page. Users are able to view updated information about computer security from different sources.
Figure 3.5.1 Workflow of the news sharing function

Figure 3.5.1 shows the workflow of the news sharing function. The news is stored inside the database. A web-crawling program regularly runs to crawl the latest news from different sources. The program crawls the title, URL and the source of the news from the news. The crawled items are wrapped into one document and are stored into the database. The old documents are deleted from the database when updated news is stored into the database to ensure that the latest information is provided. A GET request is sent to the HTTP server when the users access the news page in the application. The server will query the database and return the title, URL, and the source of the news by response.

Figure 3.5.2 Screenshots of the news page (left) and an example news (right)
Figure 3.5.2 shows the screenshots of the news page. The news page contains the list of news returned from the server. For each news, the title and source are shown. Users can click on the news title to check the information they are interested in. Upon clicking, a web view accessing the news URL will be shown, which loads the content of the website.

### 3.5.2 Practice on prevent Online Fraud

Useful practices on cybersecurity are shared in the info page of the mobile application. Figure 3.5.3 shows the screenshot of the info page. The practices are listed in a scroll view. There are totally ten tips, which covers different topics, including security concepts on password, anti-virus software and authentication.

![Online Guard](image)

Figure 3.5.3 Screenshot of the info page

### 3.5.3 Quiz on Cybersecurity Knowledge

Users are able to address their knowledge of cybersecurity and the practices listed in the info page by attending the quizzes in the quiz page.
Figure 3.5.4 Two Screenshots on the Quiz page

Figure 3.5.4 shows two screenshots of the quiz pages. The quiz page contains ten multiple choice questions, which test users’ knowledge of common cybersecurity concepts and the tips listed in the info page. Users can check the correctness of their answers by clicking the ‘submit’ button at the bottom. Upon submitting, the score of the attempt will be shown at the bottom. If the choice of the user is correct, the answer will be highlighted in green. While for an incorrect answer, the correct choice will be highlighted in red.

3.6 Machine Learning Model

Machine learning model is used to detect suspicious phishing websites. Our team had performed experiments to obtain the model which gives the best performance. In this section, the training progress and the decision made will be discussed. The following sections will first cover the training and testing dataset used, followed by introducing the training progress of the model. This section will then cover the feature selection process and the model tuning process. This section will end with the discussion of the results.
3.6.1 Training Dataset

The dataset our team used to train the model is from the paper ‘Datasets for phishing websites detection’ [13]. The dataset contains features extracted from 30,647 phishing websites. These websites are confirmed phishing URLs from PhishTank. The dataset also contains features extracted from 58,000 legitimate websites that are collected at Alexa. The unbalanced variant in the number of phishing and legitimate websites is to mimic the real-world situation, where the number of safe websites is much larger than the number of phishing websites [13]. For each website, a total of 111 attributes are extracted as features. The features are divided into six main groups.

Figure 3.6.1 Dividing a URL into different substrings [10]

Figure 3.6.1 shows how a URL is divided into different groups of substrings when extracting the features in the dataset. An URL is divided into four parts according to their roles and functions, including the domain, directory, file and parameters parts. Features of each substring are extracted. Attributes based on the whole URL string are also included in the dataset. These substring group attributes consist of string properties, including the number of specific symbols presented, such as ‘@’, ‘&’, and the number of parameters presented in the substrings. Also, features based on external services, including domain lookup time, and domain activation time, are extracted.

3.6.2 Testing Dataset

The performance of the model was evaluated on a separated testing dataset. Our team build a small dataset to test the model performance in the real-world environment. Phishtank provides a list of phishing websites collected from the community and is operated by Cisco Talos Intelligence Group (Talos) [14]. Our team collected 200 confirmed phishing websites from Phishtank. 290 legitimate websites are also collected from Alexa and our team. The unbalanced variant is to simulate the real-world situation where the number of phishing websites is smaller than the number of legitimate websites. The URLs collected underwent the same attribute extracting process. A dataset having the same format as the training dataset
is built using the resulting attributes. For a shortened URL or redirecting URL, the features are extracted from the final URL address after redirecting.

### 3.6.3 Model Training

In the model training stage, three models were constructed and trained to fit the dataset. Experiments are conducted to pick the model with the best performance. Support vector machine (SVM), extreme gradient boosting (XGBoost) and logistic regression model are constructed. The logistic regression model was treated as the baseline model, and other models were constructed to fight its performance. The model having the best performance on the training dataset will be chosen as the final model.

Data pre-processing is conducted before the data are fed into the model. First, there were columns which only contain one value. This means that those columns are independent of the labels. Those columns are removed as they cannot provide information during the classification process. Second, columns containing continuous values underwent feature scaling before they are fed into SVM and logistic regression model. The values are normalised into the same range. This can result in a faster iteration and better convergence [15]. The dataset was split into training dataset and validation dataset in a ratio of 80% to 20%. The performance of the models was evaluated on the validation dataset.

Table 3. Performance of the three models on the validation set

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Regression</td>
<td>0.9187</td>
<td>0.923</td>
</tr>
<tr>
<td>XGBoost</td>
<td>0.9685</td>
<td>0.956</td>
</tr>
<tr>
<td>SVM</td>
<td>0.9281</td>
<td>0.923</td>
</tr>
</tbody>
</table>

Table 3.6.1 shows the performance of the three models on the validation set after training on the training dataset. The performances of the models were accessed by the overall accuracy and the recall rate of the phishing class. The recall value shows the percentage of how the
model can correctly classify a phishing website as phishing. The recall rate was being more focused on than the precision value because the consequence of the model missing a phishing website is much more serious than incorrectly classifying a legitimate website as phishing. The baseline model, logistic regression, had a performance of accuracy at 91.87% and a recall rate at 92.3%. The performance of SVM was better than the baseline performance, having an overall accuracy of 92.81% and a recall rate of 92.3%. XGBoost had the best performance among the three models, its accuracy was 96.85% and the recall rate was 95.6%.

The XGBoost model was then tested with the testing dataset. The performance was evaluated.

![Confusion matrix, Precision matrix, Recall matrix](image)

**Figure 3.6.2 Performance of XGBoost on testing dataset**

Figure 3.6.2 shows the three matrixes used for performance evaluation. Confusion matrix, precision matrix and recall matrix are used. The performance of the model on the testing dataset had a great decrease compared with the performance on the validation dataset. The overall accuracy is 67.29% and the recall dropped to 56.0%. The result showed the model did not have a good performance when predicting real-world data.

Cross validation (CV) was performed to have a better understanding of the performance of the model and the cause. K-fold cross validation divides the training dataset into k groups of samples, k-1 groups of samples are used for training, and the remaining group is used for validating the resulting model [16]. This process is repeated for k times. By taking an average of the k validation scores, the model can be evaluated on different sets of unseen data. CV is able to check whether the model is overfitting. The trained XGBoost model underwent a 10-fold CV on the training dataset. It obtained a performance of mean CV score of 96.96, and standard deviation of 0.0028. It means the model has similar high performance on all the
folds. This result showed the model did not overfit to the validation set and the training set. Experiments were performed to evaluate the cause of the performance difference in the training dataset and the testing dataset.

### 3.6.4 Feature Selection

The importance of each feature was evaluated to have a deeper understanding of the training datasets. The importance of features was measured by the MDI. MDI measures how a feature decreases the impurity in all the data. A random forest is trained to fit on the training dataset. The decrease in the impurity of each feature in each tree in the forest is measured, and the average was taken. Figure 3.6.3 shows the 15 most important features sorted by feature importance measure in MDI. The blue bar is the MDI of each feature, while the black line is the deviation of the MDI of the feature among all the trees. The length of directory has the highest MDI, followed by the activation time of domain, number of spaces in file substring and the number of slashes in the whole URL. The MDI values of the features are low, all of the features had MDI lower than 0.1. For the substring attributes, the deviation is very high. Almost all the substring attributes had a deviation range covering the negative values, which means these attributes did not promise a drop in impurity. These features may lead to an overfitting of the model on the training dataset, and the knowledge that the model learnt is not generalised.
To improve the performance of the model on the testing dataset, the model needs to learn more generalised knowledge. Most of the substring attributes, which have large deviation ranges, were removed. While all the URL resolving features, and external service attributes were kept.

Figure 3.6.3 Feature importance measured in MDI of training dataset
Figure 3.6.4 Feature importance measured in MDI after feature selection

Figure 3.6.4 shows the new MDI of the remaining features. The MDI of features was increased. The feature with the highest MDI had an MDI of 0.2, there were also a few attributes that had an MDI of about 0.1. The deviation of the features also decreased. Most of the deviation range is close to the mean value. This result implies the dataset had less noise features and the model should be able to learn more generalised knowledge. The model had an overall accuracy of 88.00% and recall rate of 88.6% on the phishing class in the validation set.

Figure 3.6.5 Performance on selected features in testing dataset

Figure 3.6.5 shows the performance of the model on the selected features in the testing dataset. The model had an overall accuracy of 74.5% and recall rate of 58%. The performance of the model is slightly improved. However, the performance gap between the
The difference was about 13.5% gain in accuracy and 30% in phishing class recall rate.

3.6.5 Model Tuning

Our team also performed the same feature importance analysis using MDI on the testing dataset. A random forest was built to fit on the testing dataset. Figure 3.6.6 shows the feature importance measured in MDI in the testing dataset. Compared with the training dataset, the magnitudes of MDI are similar. There were large differences in some particular features. First, the time-to-live of hostname in the training dataset had a high value of MDI. Its MDI was about 0.15 and was ranked the second among all the features in the training dataset. While in the testing dataset, the importance of this feature was extremely low. The MDI of this feature in the testing dataset was zero and had a very low ranking, which meant this feature did not contribute much to the prediction. Second, the range of deviation of the features in the testing dataset was much larger than the range in training dataset. Many of the features in the testing dataset had large deviation range. In Figure 3.6.6, most of the black line covers the plus minus 50% of the MDI values, which implies the value of MDI varied in a large range among the trees.
Our team claimed that there was a mismatch between the training and testing datasets, where the training dataset did not totally reflect the real-world situation. The testing dataset was spitted into two groups in ratio of 80% to 20%. The XGBoost model trained on the training dataset was further fit to the 80% group to learn the knowledge of the real-world data. The final performance of the model was evaluated by the 20% group.

XGBoost is a decision tree ensembles model. The model consists of a number of classification and regression trees [17]. Each of the leaves in the tree is associated with a score, and the classification results are the sum of the scores. The addictive training strategy is used during the training process of XGBoost [17]. What the model has learnt will be fixed, and a new tree is added every timestep. Therefore, the model can keep the knowledge learnt on the training dataset while learning the new knowledge on the testing dataset.

Hyperparameters tuning was performed when fitting the model on real-world dataset. Hyperopt was used to find the optimised hyperparameters. Hyperopt finds the best hyperparameters by defining a search space, and an objective function to minimise [18]. The hyperparameters search space included the maximum depth, learning rate and other arguments. The objective function tries to maximise the overall accuracy of the model. Tree of Parzen Estimators is used as the searching algorithm. The model is tested with different hyperparameters for 200 trails.

3.6.6 Results

The performance of the tuned final model was tested with the 20% group of the real-world dataset.
Figure 3.6.7 Performance of the final model

Figure 3.6.7 shows the performance of the model on the 20% group testing dataset. The overall accuracy of the model is 80.20%. The recall rate on the phishing class was 74.4%, which means the model is able to detect 74.4% of the phishing websites. The precision rate on the safe class was 82.8%. This means among all the website predicted as safe, 82.8% of them is really safe, which gives a high confidence. The performance of the model on the training dataset did not change, the accuracy kept on 88.00%, which showed the model kept the knowledge learnt in the training dataset. The accuracy difference now is less than 8%. It was a great improvement compared with the original performance, which was nearly 30%. The recall rate difference in the phishing class was 14%. Improvement was made, but the difference is still large when compared with the accuracy difference.
4 App design

This section will introduce the design of the mobile application, including the user interface, user settings and the user experience.

4.1 Main Design

The application contains four main pages for the main functions and two sub-pages for user settings and support. All the pages follow the same general design.

Figure 4.1.1 General user interface design

Figure 4.1.1 shows the general user interface design of the mobile application. The user interface contains the green top action bar, which shows the name of the application, blocklist button and menu buttons. The menu button will show the button to the setting page and the help page upon clicking. The interface also contains the bottom navigation bar that helps users switch between the four main pages. Users can switch to different pages by clicking the icon button, which is from left to right, the browser page, news page, info page, and the quiz page. The centre part of the interface shows the main content of different main pages.
Fragments are used to show the contents of different pages and the fragment is switched upon page switching.

4.2 Browser Page

The browser page is designed to provide a convenient and safe internet browsing service to users.

![Browser Page Screenshot](image.png)

Figure 4.2.1 Screenshot of the browser page

Figure 4.2.1 shows the screenshot of the browser page. The fragment of the main content contains four components. An address bar is located at the top of the fragment. Users can enter a URL they want to access, or search keywords using the Google Search engine. The address bar is updated immediately upon website changing. Two buttons are placed at the right of the address bar. The green button is the ‘Go’ button, where user can load the URL or search the keyword inputted in the address bar by clicking it. Alternatively, the same function can be performed by clicking the search button on the keyboard. The red button is the block button, where users can add the current website into the blocklist. The browser is a WebView component implemented with a customised WebViewClient. The default home page of the
browser is ‘https://www.google.com’, users are not able to block the home page URL nor the google domain.

The browser page can also be used to load an URL outside the application.

Figure 4.2.2 Screenshot of loading URL outside

Figure 4.2.2 shows the screenshot of loading an URL outside the application. Users are able to choose Online Guard as the browser to open an URL.

4.3 User Settings
In the setting page, users are able to configure the application. Users can open the setting page by clicking the menu button on the top action bar, then clicking the setting button. Figure 4.3.1 shows the screenshot of the setting page. Users are able to change the setting of the behaviour of the browser page. First, users can enable or disable the vibration function when a website is blocked. Second, users are able to choose what technologies to be used to block phishing websites, including the blocklist, APIs and the machine learning model. Third, users can edit the behaviour of the notification when a website is being blocked. Figure 4.3.2 shows the two approaches when blocking a website. Users can choose to have a pop-up window showing the warning message, and users can decide whether to continue to access
the website or not. Users can also choose to turn off the pop-up window function, and the website will be blocked directly without asking the user.

Figure 4.3.1 Screenshot of the setting page

Figure 4.3.2 Block a website with pop-up (left) and toast (right)
4.4 Help Page

Users can learn more about the application through the help page. Users can access the help page of the application by clicking the menu button on the top action bar, then clicking the help button.

![Help Page Screenshot](image)

**Figure 4.4.1 Screenshot of the help page**

Figure 4.4.1 shows the screenshot of the help page. The help page contains some frequently asked questions about the application and the corresponding answer, including the introduction of the application and instructions on how to use the application.

4.5 UX design

Several configurations are made to provide a good UX to users. Firstly, the application is designed to be self-descriptive. The components are able to explain themselves such that the users can understand what they can do with the components and what the page is doing.
Figure 4.5.1 Examples on self-descriptive components

Figure 4.5.1 shows the screenshots of some self-descriptive components. Users can long press on the buttons, and the description of the button will be shown. Also, there are hints in the EditText view. For example, in Figure 3.2.4, instructions are given to users by the hint inside the TextView in the blocklist page.
Secondly, the application is responsive. The application interacts with the actions of users and changes the interface. In Figure 4.5.1, the page in active will have a green icon, while other pages will have an inactive icon. Moreover, the application provides informative feedback on users’ actions. Users are able to check the result of the actions and know what changes have been made. Figure 4.5.2 shows the toast message that informs the user a website has been added to the blocklist successfully.
5 Limitations

This section introduces the limitation of this project. For each limitation, the cause and the effect will be discussed.

5.1 Verify User Feedbacks

In this project, users are able to provide feedback on the prediction of the model or report phishing websites. However, due to the lack of resources, our team will not verify the report provided by the users.

There exist phishing websites database services like the PhishTank, which maintain a list of phishing websites and receive phishing reports from users. These services verify phishing reports to ensure the reports do not mislead other users. For example, PhishTank verifies a phishing report by voting [14]. Users need to be registered and signed in in order to have the right to vote. This approach requires a large user base and the quality of the users needs to be checked. Account holders need to have the ability to distinguish between phishing and safe websites. While in this project, time resources are not enough to build a user base having an enough size and quality. Other approaches to verify phishing reports, including using machine learning model or using engines to verify, are not applicable as these approaches may result in a chicken-egg problem.

Therefore, our team did not implement approaches to verify phishing reports. This leads to the phishing websites explored by users cannot be shared with other users efficiently. The users can define a blocklist and block the access locally if they discover a phishing website. However, they can only share this information indirectly, which is giving feedback to the machine learning model and the model should be able to block the website next time after the model is updated. While if the phishing reports can be verified, users can share a common blocklist together, where the access of all users toward a phishing website will be blocked immediately after they are reported. The second problem of this limitation is that our team cannot ensure that the feedback to the machine learning model contains the true label of the website. This may lead to a mismatch between the data label and the true label. The model
may learn incorrect knowledge and its performance may be even worse after fitting to feedback.

5.2 Lack of Datasets
The most significant limitation faced when training the machine learning model was the lack of training dataset. There are not many available choices of dataset. As mentioned in the previous section, the dataset used to train the model did not match the real-world situation very well. There exist features that may lead to overfitting, and there exists mismatch in feature importance between the training and testing dataset. Apart from the dataset used in this paper, another dataset from another paper was also used to perform experiments and train the model. The dataset is ‘Phishing Websites Data Set’ from the UCI repository [19]. Different features are extracted from websites in this dataset.

Figure 5.1 Feature importance on the UCI dataset

Figure 5.1 shows the feature importance calculated by MDI in a random forest model fitting on the dataset. The feature having the best performance was the SSL_final_State, which shows whether a website has a valid HTTPS certificate. A valid certificate is defined as a certification issued by a trusted issuer and the time to expire is longer than one year. In the
training dataset, 91.44% of the phishing websites do not have any certificate, and 0.34% of the phishing URLs have certificates that are not valid. However, in the self-collect dataset, only 37.9% of the phishing URLs do not have any certificate and all the remaining URLs, more than 60%, have certificates that are not valid. The mismatch between the two datasets leads to the model trained on this dataset cannot learn to make correct predictions in the real-world situation.

This could be solved by creating a new dataset on phishing classification. The dataset should be large enough for training, for example, a ten thousand level, so that the model can learn generalised knowledge. Moreover, the features chosen should be relevant and appropriate, so that the model can perform well in the real-world situation. However, this is beyond the scope of this project. Our team decided to tune the model that is trained on existing datasets.
6 Conclusion

This section discusses the future works possible and gives a conclusion of this paper.

6.1 Summary

This paper presented the background and the methodology behind the project. The implementation details, the experiment performed, and the results obtained are also discussed. The application design and the limitation faced are also covered.

This project is motivated by the increasing trend of the number of cyber frauds. Our team aimed to cope with the growing cybersecurity threat by providing a free solution to users. The project should be able to provide them with a safe internet environment and increase their sense of cybersecurity. This paper introduced a mobile application project, with all the functions available freely.

The application provides three main functions. The first function provides users a safe internet environment by blocking reported phishing websites. It uses user-defined blocklist, APIs and Google safe browsing services to block access towards known phishing websites. The second function is to detect unknown phishing websites and block the access towards them. Machine learning model and APIs are used to scan the URLs. The third function shares information on cybersecurity. Users are able to get updated with the latest information regarding online frauds and learn common practices to protect themselves. The methodology, implementation details and workflow are discussed in this paper.

A machine learning model is built to detect suspicious websites. The training dataset is from another academic paper, and our team built a testing dataset from real-world URLs. Experiments are carried out to find the model with the best performance and XGBoost is chosen as the final model. Several analyses and modifications are conducted to further improve the performance model, including feature selection and model tuning. The resulting model has an overall accuracy of 80.4% and a recall rate of 74.4% on the phishing class.
Lastly, this paper discussed the limitation of this project. Due to the lack of resources, our team encountered difficulties when handling the phishing reports. This resulted in a local blocklist, where the phishing information between users cannot be shared efficiently. The second limitation faced was the lack of suitable dataset. There were not sufficient dataset resources that can reflect the real-world situation. This leads to a performance gap of the model between the training dataset and the real-world data.

### 6.2 Future Works

At the stage of the model training, our team discovered that the features provided by the URL might not be able to provide sufficient information for a model to give a good prediction result. Natural language processing may be one of the possible solutions. A deep learning model can take the HTML and JavaScript of the website as the input, and output the label of the website. This approach allows the model makes prediction on the content of the websites, which may help the model having a higher accuracy.
References


