Journal of Multidisciplinary Engineering Science Studies (JMESS)
ISSN: 2458-925X
Vol. 10 Issue 12, December - 2024

Catch-Phish: An Automated Phishing Page Detection and Reconnaissance System

[1]Kazeem B. Adedeji, [2]Ayobami O. Adedokun, [3]Sammy O. Oladiran, and [1]Titilayo A. Ogunjobi

[1]Department of Electrical and Electronics Engineering, Federal University of Technology Akure, Ondo State, Nigeria
[2]Department of Computer Engineering, Federal University of Technology Akure, Ondo State, Nigeria
[3]Department of Information and Communication Technology, Federal University of Technology Akure, Ondo State, Nigeria
Email: kbadedeji@futa.edu.ng

***Abstract*— The internet has firmly established itself in daily life through its ability to connect people and businesses worldwide. With increasing reliance on the internet for communication, business, and social networking, the security of online information has become a serious problem. Attackers can now target people and organizations on a global scale, and phishing is one of the most popular and effective methods. Phishing attacks have become a major concern for individuals and organizations, with over 200,000 unique phishing attacks being reported in the first quarter of 2021. In response to this growing threat, this study developed an automated phishing page detection and reconnaissance system. The system is built as a desktop application using Electron.js, integrating both the frontend and backend, with the Google Maps API utilized for domain location visualization. The frontend of the web browser application was developed using Bootstrap, while Node.js served as the backend. Both applications interact with IPQualityScore's Malicious URL Scanner API, leveraging machine learning algorithms and querying up-to-date databases of phishing URLs to detect webpage risks and zero-day phishing threats. The system also incorporates a feedback mechanism that allows it to adapt to new phishing techniques used by attackers. The performance of the system was evaluated using a dataset of phishing and legitimate websites and it achieved an accuracy rate of 95.6%. This system offers a promising solution to the problem of phishing attacks and provides individuals and organizations with a powerful tool to protect themselves against these threats.**

Keywords—internet; cyber security; phishing attack detection; machine learning

I. INTRODUCTION

The internet has become an integral part of our lives, connecting people and businesses worldwide. With increasing reliance on the internet for communication, commerce, and social networking, the security of online information has become a major concern. The internet has enabled attackers to target individuals and organizations globally, with phishing being one of the most common and successful attacks as illustrated in Fig. 1. The figure shows the distribution of security attacks on the internet and their victims. 324, 000 were reported to be phishing victims, while victims of other attacks like spoofing, extortion were not up to 50% of phishing victims all through that year. This suggests that phishing is a very viable attack and current guards against it have not been very effective.

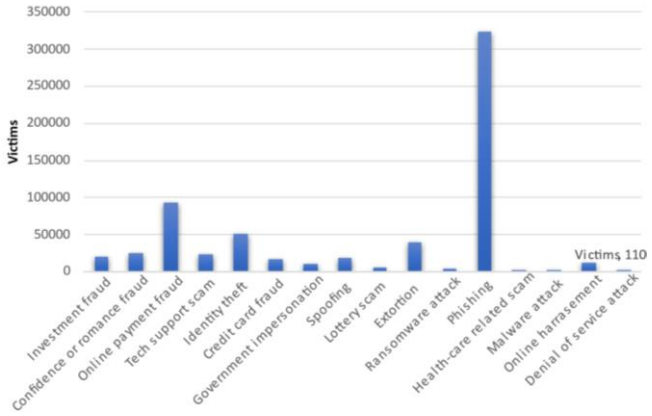

Fig. 1. *Security threat on the internet [1].*

Phishing, among other cybercrimes, has increased in frequency over time, harming people, and businesses severely. The goal of a phishing attack is to deceive a target into disclosing personal information or downloading malware onto their computer. As shown in Fig. 2, phishing attacks are frequently conducted by building a phoney login page that looks real but is under the attacker's control. These pages are frequently used to steal credit card numbers, login credentials, and other private sources of information. A successful phishing attack can have many consequences. Among others, the financial loss due to phishing attacks is significant. An internet crime report presented by the FBI in 2018 revealed that business email compromise attacks cost US businesses over $1.2 billion [2]. The creation of an

automated phishing page detection and reconnaissance system is vital for limiting the hazards posed by phishing attempts.

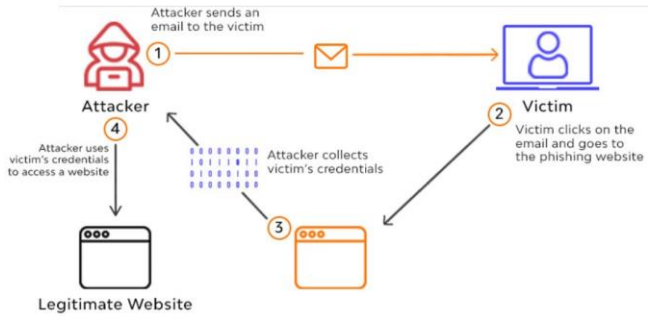

Fig. 2. *An illustartionof phishing attack [2].*

In recent years, online attackers have become more skilled at deceiving their targets, leading to an increase in phishing attempts. Making phony login pages that mimic authentic websites is one of the most widely utilized phishing attack strategies. These pages are frequently stored on servers owned by the attackers or those that have been compromised online. An attacker can exploit a victim's login credentials to access the victim's accounts or steal their data once the victim entered them. Developing a scheme that can automatically detect phishing pages is vital. Several studies have been conducted in the past with varying degree of success [3-7].

Comparing the proposed approach to current phishing detection systems will yield several advantages. First, because of its attachment to existing machine learning algorithms, which can rapidly and accurately identify phishing pages based on their features, it is more accurate and dependable. The solution will also be quicker than current tools, enabling real-time identification and phishing attack prevention. The proposed system is also easier for consumers to use, with a straightforward interface that enables them to quickly and easily recognize and generate reports on phishing URLs. Desktop web browser users can utilize a variety of tools to identify phishing pages; however, the efficacy of these tools is frequently constrained. Many rely on user feedback, which can be slow and unreliable, to find the phishing URLs. Some technologies utilize heuristics to identify phishing pages; however, these heuristics can be easily bypassed by attackers. By utilizing endpoints based on machine learning techniques to automatically identify and categorize phishing pages, the suggested solution seeks to overcome these restrictions. By connecting to other systems that have been trained on a sizable dataset of well-known phishing pages, the system can recognize traits that are typical of phishing pages. Additionally, the system can instantly update its algorithms, enabling it to adjust quickly to new phishing techniques as they appear.

II. Literature Review

Phishing is a fraudulent activity where an attacker attempts to obtain sensitive information such as login credentials, credit card details, or other personal information by posing as a trustworthy entity. Phishing attacks can occur via email, text message, or phone call, and they are becoming increasingly sophisticated, making them difficult to detect. Phishing scams are one of the most common types of web scams, where scammers send fake emails or messages that appear to be from legitimate sources to trick individuals into divulging sensitive information such as usernames, passwords, or credit card details. According to a study by the Anti-Phishing Working Group (APWG) [8], there were over 222,000 unique phishing attacks reported in the first quarter of 2021, with the financial sector being the most targeted.

Another security threat on the internet is malware. Malware is malicious software designed to damage or disrupt computer systems, steal information, or gain unauthorized access to a computer network. Malware can be introduced to a computer system through various means, such as downloading a malicious attachment, clicking on a malicious link, or visiting a compromised website. Other internet security threats include hacking, denial of service attacks (DoS), distributed denial of service (DDoS), ransomware, and social engineering. Hacking involves unauthorized access to computer systems, while denial of service attacks aims to disrupt the availability of a particular service or website by flooding it with numerous spoof request. Ransomware is a type of malware that encrypts data on a computer system and demands a ransom payment to restore access, while social engineering involves manipulating individuals into divulging sensitive information. Some components of social engineering include Cyber-stalking, spear-phishing, vishing, smishing etc.

The consequences of these security threats can be severe, ranging from financial losses to reputational damage for organizations. Fig. 3 shows that during the COVID-19 pandemic, phishing was the mostly used social engineering technique used for cyber-attacks (35.3%) while cyber-stalking was the least used (1.3%).

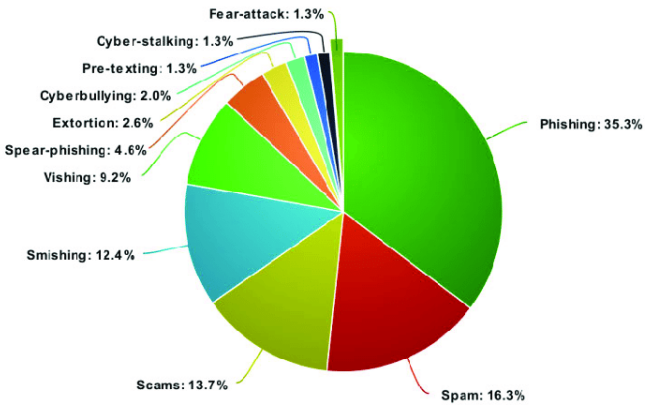

Fig. 3. *Different social engineering techniques used for cyber-attacks druing the cOVID-19 pandemic [9].*

A. Phishing Background and its Impact

Phishing detection systems are critical in mitigating the negative impact of phishing attacks. These systems aim to identify and block phishing messages before they reach the intended victims, or to detect and block phishing websites that try to lure users into divulging sensitive information [10]. Effective phishing detection systems can help prevent financial losses, identity theft, and reputational damage for individuals and organizations [11]. Phishing detection systems can also help organizations comply with legal and regulatory requirements related to data protection and privacy. For example, the General Data Protection Regulation (GDPR) requires organizations to implement appropriate technical and organizational measures to protect personal data from unauthorized access, disclosure, or theft [12]. Moreover, phishing detection systems are essential in maintaining user trust and confidence in online services and e-commerce platforms. By ensuring that users can transact safely and securely, organizations can foster customer loyalty and build their reputation [13]. An effective phishing detection and mitigation mechanisms can help prevent financial losses, identity theft, and reputational damage, and maintain user trust and confidence in online services.

Website anti-phishing techniques can be categorized into 4 according to the studies presented by [14]. This category is illustrated in Fig. 4.

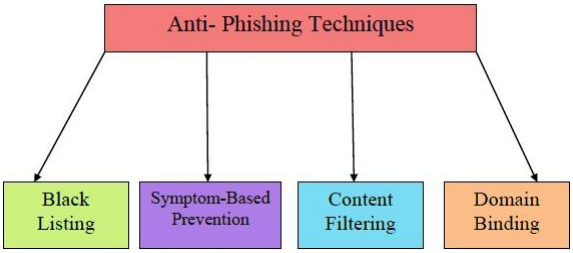

Fig. 4. *Types of anti-phishing techniques.*

(a) **Blacklisting:** This is a technique that identifies phishing websites by comparing URLs against a database of known phishing sites, such as those maintained by the Anti-Phishing Working Group (APWG) and Phish Tank. Tools like Netcraft, which operate on this method, rely on continually updated blacklists sourced from spam emails and other reports. However, the exponential rise in phishing websites has rendered this approach less efficient, as the sheer size of the blacklist may lead to operational difficulties [15].

(b) **Symptom-based anti-phishing techniques:** These focus on analyzing the content of webpages, generating alerts based on the detection of specific phishing indicators. Many of these approaches leverage visual similarity methods, such as document object model comparisons, visual features, and CSS characteristics, to identify phishing attempts. While highly effective, this method requires significant computational resources, making it impractical for manual use and more suitable for automated engines that deliver these services to users [15].

(c) **Content filtering:** This is another anti-phishing strategy, utilizes the content of emails or webpages to detect phishing, often employing advanced techniques like Bayesian statistics and support vector machines (SVM). This approach, though sophisticated, similarly demands considerable computational power, particularly as it integrates artificial intelligence methods [15].

(d) **Domain binding:** This is a browser-based defense mechanism that associates sensitive information with specific domains, alerting users to the appropriate domain for that information. Browser extensions, such as PhishDetect and PhishDetector, have incorporated this technique.

B. Exixting Phishing Detction System

With the increase in phishing attacks, numerous phishing detection systems have been developed to counter these threats. Google Safe Browsing, Symantec Norton Safe Web, and McAfee SiteAdvisor are some of the existing phishing detection systems. These systems utilize various detection techniques and features to identify and protect users from phishing attacks. For example, Google Safe Browsing analyses web pages and generates lists of suspected phishing and malware pages, while Symantec Norton Safe Web provides website ratings based on security and safety ratings. McAfee SiteAdvisor uses Global Threat Intelligence (GTI) to catalog the reputations of IP addresses around the globe. IP addresses associated with phishing websites, sites infected with malware, or otherwise malicious sites, have a ‘bad’ reputation in the GTI database. So, they’re blocked from connecting to your PC and show up as risky connections. It also blocks unsafe websites and lets you know if a site is known for phishing or other malicious activity. Table I shows a comparison between the three existing phishing detection systems discussed. It can be observed that the three detection/prevention tools work on the same platforms but differ in terms of the kind of security they provide and the speed at which the detect/prevent phishing pages. They all provide malware and phishing protection.

C. Comparative Analysis of Existing Phishing Detection System

Several techniques have been used for phishing detection. These are rule-based, signature-based, and machine/deep learning-based techniques. Rule-based and signature-based techniques are based on predefined rules and signatures to detect phishing attacks, which can limit their effectiveness in detecting new and previously unknown attacks. In contrast, machine learning-based techniques, such as decision

TABLE I. COMPARISON BETWEEN GOOGLE SAFE BROWSING, MCAFFEE SITEADVISOR, AND NOETON SAFE WEB OF THE OVERALL PERFORMANCE OF THE CLASSIFIERS.

Metrics	Google Safe Browsing	McAfee SiteAdvisor	Norton Safe web
Security	Real-time malware detection	Real-time malware detection and removal	Real-time dark web monitoring and malware detection
Phishing Protection	Yes	Yes	Yes
Malware Protection	Yes	Yes	Yes
Scan Speed	Fast	Slow	Very fast
Browser Compatibility	Firefox, Chrome, Microsoft Edge and Safari	Internet Explorer, Firefox and Chrome	Chrome
Customer Support	FAQs	Phone, forums and frequently asked questions (FAQs)	24/7 live chat, phone and FAQs

TABLE II. COMPARISON PHISHING DETECTION TECHNIQUES.

Technique	Advantages	Disadvantages
Rule-based	Easy to implement and low false positive rate	Low efficiency against new phishing attacks
Signature-based	High accuracy rate and low false positive rate	Low efficiency against new phishing attacks
Machine learning-based	High accuracy rate and can detect new phishing attacks	Requires large datasets for training

trees (DT), random forests (RF), and support vector machines (SVM), have shown promising results in detecting new and previously unknown phishing attacks. Rule-based techniques use a set of rules to identify phishing emails. These rules are based on the characteristics of phishing emails such as the presence of certain keywords or phrases, suspicious URLs, or attachments. Signature-based techniques use a database of known phishing emails to identify new phishing emails. These databases are created by security experts who analyse phishing emails and extract their signatures. Machine learning-based techniques use algorithms to learn from a large dataset of phishing and legitimate emails to identify new phishing emails. These algorithms can be trained using various machine learning techniques. In Table II, a comparison of these techniques was presented. As can be observed, the machine learning based technique is preferable but requires a large dataset for training.

Among the three phishing techniques, machine learning-based detection technique has become a popular approach for phishing detection due to its ability to learn from previous attacks and identify new and previously unknown attacks. Various machine learning approaches, such as supervised and unsupervised learning, have been employed for phishing detection. For example, K-means clustering has been used for unsupervised learning to cluster phishing websites based on their characteristics in a study by Sahu & Shrivastava [16]. On the other hand, supervised learning algorithms such as logistic regression and artificial neural networks have been used for detecting phishing attacks with high accuracy as used in a study by Shahrivari et al. [17].

D. Machine Learning Techniques for Phishing Detection

Machine learning (ML) algorithms are widely utilized in phishing detection due to their ability to process vast amounts of data and identify patterns that may indicate malicious activity. Several popular ML algorithms, including DT, RF, Naive Bayes (NB), and SVM, have been applied to phishing detection, each offering distinct strengths and limitations [18]. Decision trees are supervised ML algorithms that classify data by recursively splitting the feature space, forming a tree-like structure used for predictions. The RF, an ensemble method, combine multiple decision trees to enhance accuracy and reduce the risk of over fitting. Studies have demonstrated the effectiveness of these approaches in phishing detection. For instance, Ahmadian et al. [19] used DT to classify phishing URLs with an accuracy of 98.7%. The NB relies on Bayes' theorem. It calculates the probability of a sample belonging to a specific class based on its feature values. SVMs, on the other hand, aim to find an optimal hyperplane that separates the data into distinct classes. Both algorithms have shown efficacy in phishing detection. For example, Alhaisoni and Khan [20] classified phishing emails using NB with 97.3% accuracy, while Singh et al. [21] applied SVMs to phishing website detection, reaching 98.8% accuracy. Hybrid models have also been proposed for phishing attack detection [22]. Zhang and Li [23] focuses on using machine learning algorithms to

detect phishing websites in real-time. In this study, a support vector machine was utilized. The accuracy of the developed system was analysed and the results presented show that phishing websites are correctly identified with a remarkable 97.3% accuracy. The dataset used for training and testing the support vector machine model consisted of 1,500 phishing websites and 1,500 legitimate websites. It is unclear where or how the authors obtained this dataset. In Kaur and Kaur [24], the authors explore the use of machine learning algorithms for phishing website detection. Two machine learning algorithms namely; DT and RF algorithms were investigated. Experimental result found that the RF algorithm was able to achieve a 96.3% accuracy rate. Gupta and Jain [25] focus on the use of SVM algorithm for phishing website detection. Experimental results found that their system was able to achieve a 97.8% accuracy rate. In Bhadoria and Singh [26], the accuracy of two machine learning algorithms; RF and SVM was investigated for phishing website detection. The results obtained found that the RF algorithm was able to achieve a high accuracy rate of 96.7%. A comparative analysis of these machine learning techniques highlights their respective strengths and limitations in phishing detection. Singh et al. [27] compared SVMs, DT, and RF, concluding that SVMs were the most effective for phishing detection. Ultimately, the effectiveness of any ML algorithm depends on factors such as the dataset, selected features, and the configuration of the algorithm itself. Therefore, careful consideration is required when choosing the most appropriate ML algorithm for phishing detection tasks.

E. Deep Learning Techniques for Phishing Detection

Deep learning, a subset of machine learning, has demonstrated significant promise in areas such as image and speech recognition, natural language processing, and anomaly detection. In the context of phishing detection, deep learning techniques are used by training models on large datasets of phishing and legitimate websites to accurately differentiate between them. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are among the most popular deep learning methods applied to phishing detection [28-30]. In CNNs, phishing detection was achieved by extracting features from webpage images to identify visual similarities indicative of phishing attempts. Studies have shown encouraging results in this regard. Korus et al. [31] developed a phishing detection model combining CNNs with decision trees, achieving high accuracy. Nguyen et al. [32] designed a novel CNN architecture to address imbalanced datasets in phishing detection, employing convolutional layers with varying filter sizes to enhance feature extraction. Zhu et al. [33] introduced an improved CNN-based method that incorporated both textual and visual features of webpages, combining convolutional layers with pooling layers to classify sites as phishing or legitimate. In Kim et al. [34], the use of deep learning model to detect phishing websites was presented. In this study, a Convolutional Neural Network (CNN) model was trained to classify websites as either phishing or legitimate. Experimental results found that the model had an accuracy of 98.2%. Similarly, Ahmadi and Ghorbani [35] introduced a new approach for detecting phishing websites by using deep learning algorithms. The authors specifically utilized a Deep Belief Network (DBN) model, which was able to classify websites with an accuracy of 95.6%. RNNs, on the other hand, excel in processing sequential data, making them suitable for phishing detection by modelling the flow of user interactions with websites. By analyzing sequences like user keystrokes and mouse movements, RNNs can detect patterns typical of phishing attacks. For instance, Lee et al. [36] proposed a phishing detection model using a combination of RNNs and graph-based features, achieving high accuracy. Mehmood et al. [37] similarly applied RNNs with LSTM layers to email phishing detection, using both textual and non-textual features for classification. Pham et al. [38] further extended this approach by integrating email content and metadata features into their RNN-based model for phishing detection. These studies underscore the potential of RNNs in phishing detection, particularly when applied to sequential and behavioral data.

The review of related studies has shown the importance of continuing research in this domain. Moreover, phishing techniques are continuously evolving, with attackers utilizing new methods such as social media, mobile devices, and machine learning-based tactics. These emerging trends demand advanced detection strategies capable of analyzing diverse data sources and identifying subtle phishing patterns. Addressing these challenges is crucial for enhancing the effectiveness of phishing detection systems in the face of evolving cyber threats.

III. RESEARCH METHODS

The Catch-Phish system architecture is shown in Fig. 5. It consists of five key components, which work together to detect phishing attempts across web applications. The user can interact with the system via either a web browser or a desktop application, both of which are integrated with the API and application logic. As shown in Fig. 5, a browser extension enables seamless interaction between the web and desktop platforms. The main components of the catch-phish system include:

- PC: A personal computer is required to run Catch-Phish, given its desktop application nature. It must be equipped with a stable internet connection to support the tool's functionality.
- Web Browser: As an essential part of the system, the web browser is responsible for retrieving web page files from servers and displaying them to users. Catch-Phish monitors these webpages for phishing and other security risks during browsing.
- UI/Frontend: The user interface is designed to be intuitive and interactive, with a focus on creating

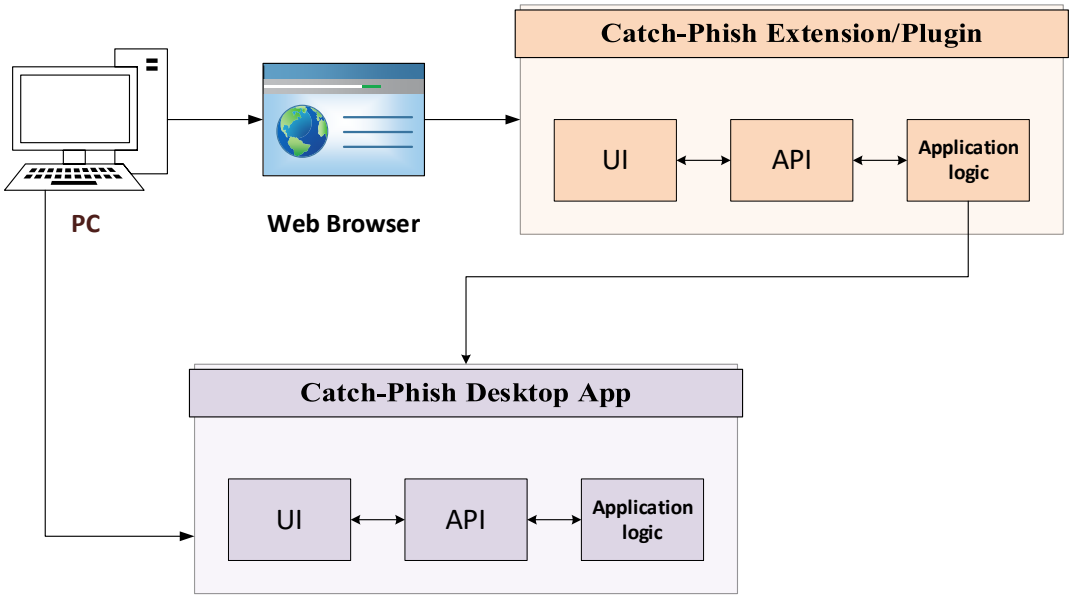

Fig. 5. *The catch-phish system architecture.*

an easy-to-use and visually appealing experience for users. It also manages interactions and state updates.

- API: The API facilitates communication between the desktop application and anti-phishing servers. When a user encounters a potential phishing webpage, the page contents are sent via API requests to be analyzed for threats.
- Application Logic: This component ensures the system's proper functioning, containing the logic and instructions for system operation. It includes API connection keys and processes user interactions based on the analysis of API results.

A. System Development Tools

Catch-Phish was developed on an HP Folio 9480m personal computer using different software tools. For the Integrated Development Environment (IDE), a simple but extensible text editor; the Visual Studio Code (Microsoft, 2022b) was used because of its easy-to-use environment and very good intelligence. It is a code completion tool by Microsoft Inc. For the Web Browser Extension/Plugin; The UI/Frontend was developed using Bootstrap.js, created at Twitter. The backend/application logic and API requests were handled using the latest version of Vanilla JS also referred to as pure JavaScript. This is a lightweight and performant language developed for Netscape 2 to build reusable and responsive web components. This was used in combination with and JQuery; an open-sourced JavaScript library that simplifies creation and navigation of web applications. For the development of the Desktop Application; the UI/Frontend was developed using Electron.js. It is an open-source runtime framework that allows the user to create desktop-suite applications with HTML5, CSS, and JavaScript. The backend/application logic was developed using Node.js; an open-source JavaScript runtime engine that makes it possible to make asynchronous API requests due to its speed, easy scalability and efficiency. IPQualityScore's malicious URL scanner API was used as the webpage risk score assessment gateway in both applications of the system to assess webpage contents and features for internet security risks. GeoPlugin API was used to retrieve the latest and relevant geographical location information on domains and URLs fed into the application.

Fig. 6 is a block diagram of the Catch-phish system showing the relationship between the components of the system. These are a web browser plugin, a desktop application and IPQuality score's malicious URL scanner API.

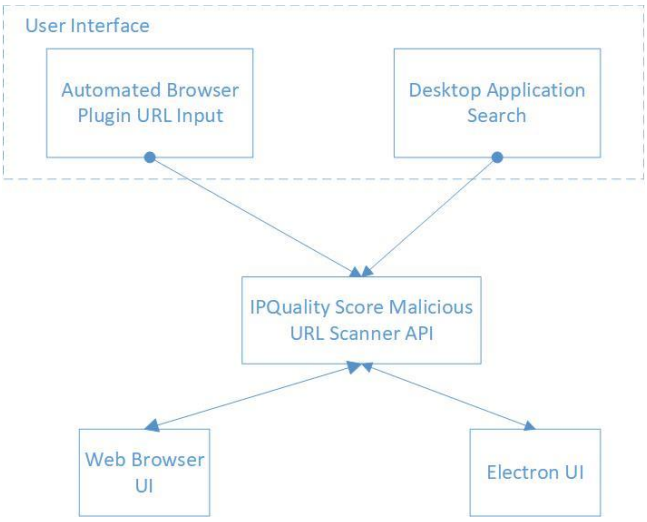

Fig. 6. *Relationship between blocsk in the catch-phish sytem.*

Requests to the API can be made from both the web browser plugin and the desktop application while the

response from these requests made interflows between the web browser interface and the desktop application interface. The UML activity diagram of the system is represented in Fig. 7. Based on the risk assessment of the webpage/URL, a page is either flagged as legitimate/safe or suspicious/phishing. No cache is stored to ensure freshness on every activity. This research utilized Bootstrap as the technology to implement a user interface for the Web Browser Extension. For security reasons, other frameworks like React.js or Node.js cannot be used. A workflow for how the plugin would function was initially created and later improved based on the primary purpose of the Catch-plugin plugin, which is to automatically check for phishing and provide advisory on the current webpage the user is on. Fig. 8 is a screenshot of the plugin interface taken as it was being used on a webpage linked to the Surfshark domain.

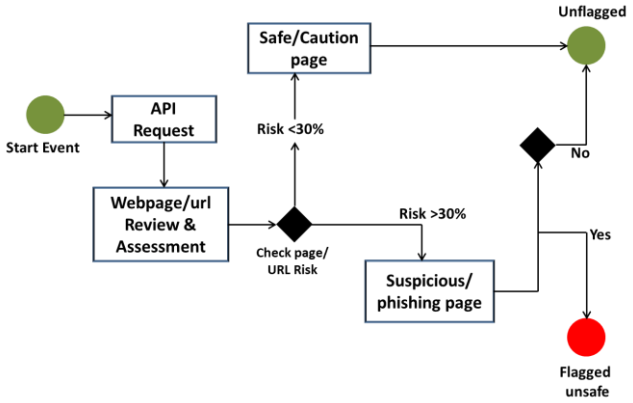

Fig. 7. *UML activity diagram of the catch-phish system.*

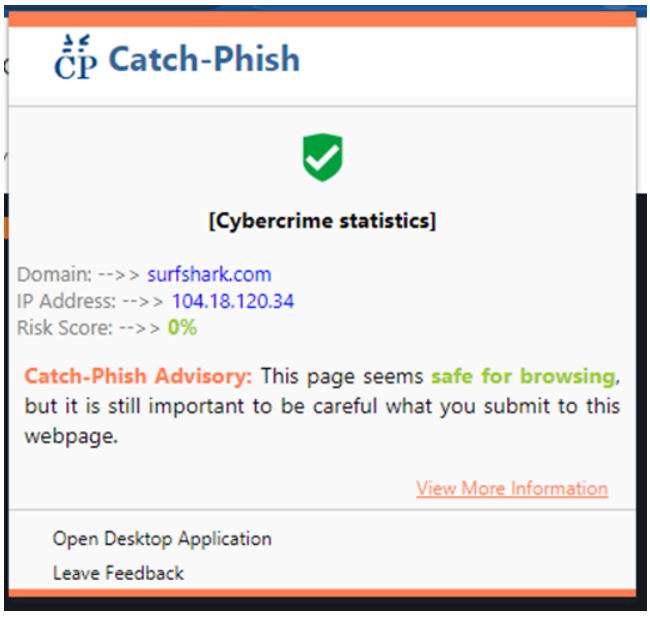

Fig. 8. *The catch-phish plugin user interface.*

The inclusion of a browser plugin in the Catch-Phish system is driven by several factors that make it a vital component for phishing detection. One of the primary advantages is that browser plugins offer integrated security policies and guides, providing an additional layer of protection for users. It also allows for secure background processes to be executed, making it ideal for real-time monitoring of potential phishing threats while users browse the web. Furthermore, browser plugin enables easy addition of new features, thereby expanding the system's capabilities without requiring significant modifications to the core architecture.

B. Creating a Browser Plugin

Before proceeding to create a browser plugin/add-on/extension, a few things are required: Chrome Web browser must be installed on the system with developer mode enabled on it. Also, Node.js and NPM must be installed in the system. The process to set up involves using the workspace terminal in a text editor like visual studio code to create a directory to house the plugin development files. A JSON package file (package.json), A JSON manifest file (manifest.json) and an index HTML file (index.html) are the core plugin development files that must be created for the plugin to work as shown in Fig. 9. A brief description of these files is presented.

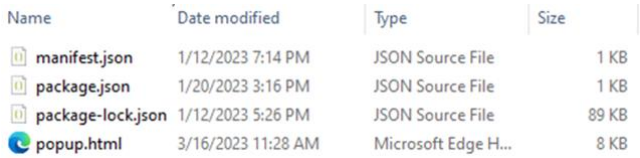

Fig. 9. *Folder structure of the plugin after setup.*

The development of the Catch-Phish browser plugin relies on several key files that define its structure and functionality. The first essential file is the package.json, which holds important metadata about the project, such as the name, version, and dependencies required for the plugin's development. Notably, dependencies like Axios, webpack, and webpack-cli are included to streamline and speed up the development process. Another critical component is the manifest.json file, which informs the browser about the plugin's behavior. It defines various attributes such as the plugin's icon files, the HTML and JavaScript files to be launched when the plugin is activated, and the necessary permissions the plugin will require from the user's browser to function correctly. This file essentially serves as the blueprint that enables the browser to understand and manage the plugin.

Lastly, the Index HTML file plays a role in defining the user interface or any front-end functionality. Unlike the package.json and manifest.json files, the name of the index file can be modified, but it must be referenced correctly in the manifest file. This index file also allows for linking to other HTML, CSS, and JavaScript pages, enabling a flexible and dynamic user experience within the plugin's design. Together, these files serve as the foundation for building, deploying, and maintaining the Catch-Phish browser plugin.

C. *Interface design*

Fig. 10 shows the flow chart of the Catch-Phish plugin. It shows each step that would be taken from when users access the plugin. After the user installs the catch-phish extension from the Chrome Web Store, the user navigates to a website or clicks on a link that may be a phishing attempt. The catch-phish extension then analyses the URL and compares it to a database of known phishing site. If the URL is identified as a phishing attempt, the extension will display a warning message to the user advising them not to proceed to the website. If the user ignores the warning and proceeds to the website, the extension will continue to monitor the website and provide real-time protection against phishing attempts. The extension also allows users to report phishing attempts and leave feedback to help improve the system.

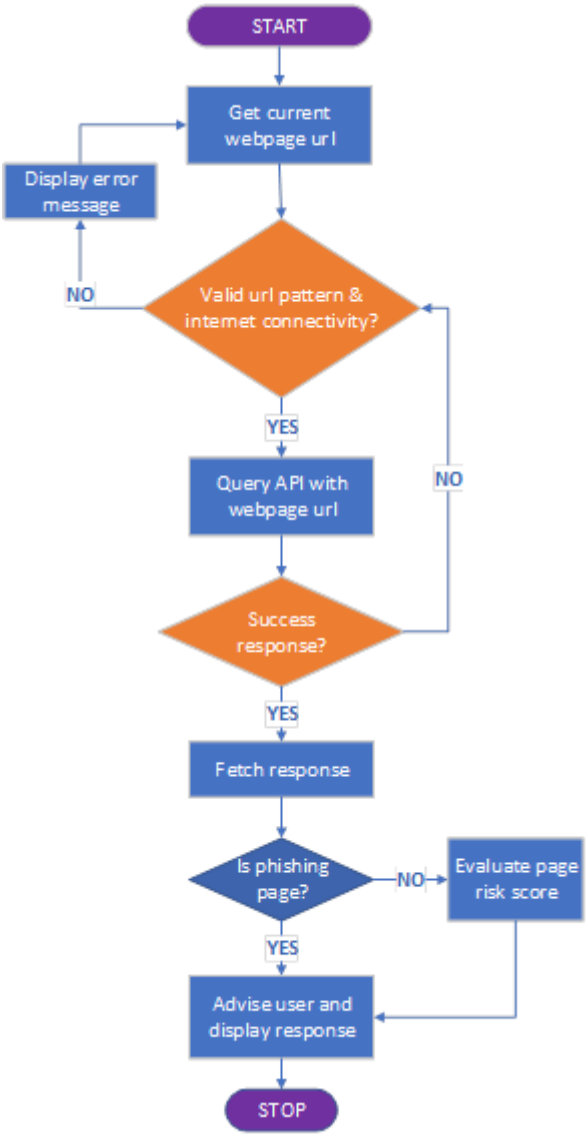

Fig. 10. *Flow chart of the catch-phish browser plugin.*

The IPQuality Score's malicious URL scanner API plays a crucial role in the backend logic of the Catch-Phish browser extension. This API is designed to detect malicious URLs associated with phishing campaigns, misleading advertisements, and malware. It utilizes live threat intelligence feeds to identify zero-day phishing links and suspicious behavior, scanning URLs to uncover poor reputation domains and phishing threats. Additionally, it can recognize parked or hijacked domains, providing real-time intelligence supported by advanced machine learning models, and is straightforward to implement.

In the Catch-Phish plugin, the API is employed for real-time phishing detection. If the API response indicates that a URL is classified as phishing, users are immediately warned against submitting any information on that site. Conversely, if the URL is deemed safe, the plugin generates a risk score based on the API response. A score below 35% suggests that the page likely lacks forms, advising users that it is safe to input details but may still pose other risks, such as SSL certificate expiration. A risk score above 35% without a phishing flag indicates that the domain has exhibited malicious activity in the past 48 hours. This prompts the users to be cautious and refrain from entering sensitive information.

D. *Cath-Phish Desktop Application*

The Catch-Phish desktop application uses Electron.js for both its backend and frontend implementation. Electron.js is an open-source runtime framework built on Node.js that facilitates building desktop applications with HTML5 and CSS DOM elements by Cheng Zhao, an engineer at Github Inc. Fig. 11 illustrates a preview of what the Catch-Phish desktop application looks like on launch. Electron.js is a preferred framework for developing the Catch-Phish desktop application due to its flexibility in designing user interfaces, allowing for limitless aesthetics. It enables developers to maintain a single JavaScript codebase, facilitating the creation of cross-platform applications compatible with Windows, macOS, and Linux without requiring native development.

Electron.js comprises three primary components: Chromium, Node.js, and custom APIs as shown in Fig. 12. Chromium is responsible for rendering and displaying web content, allowing access to all browser APIs and development tools akin to those found in Google Chrome. Node.js provides access to system capabilities, enabling interactions with the filesystem and operating system functionalities. Additionally, Electron.js includes custom APIs that facilitate the creation of common desktop experiences, making it easier for developers to implement features such as context menus, desktop notifications, and keyboard shortcuts.

E. *The Main and Renderer Processes*

A running Electron.js app maintains two types of processes, the Main process, and one or more Renderer processes. Fig. 13 visualizes how these two process types relate to each other. The entry point of an Electron.js application is the Main process, which is simply a Node.js environment. This is where all the

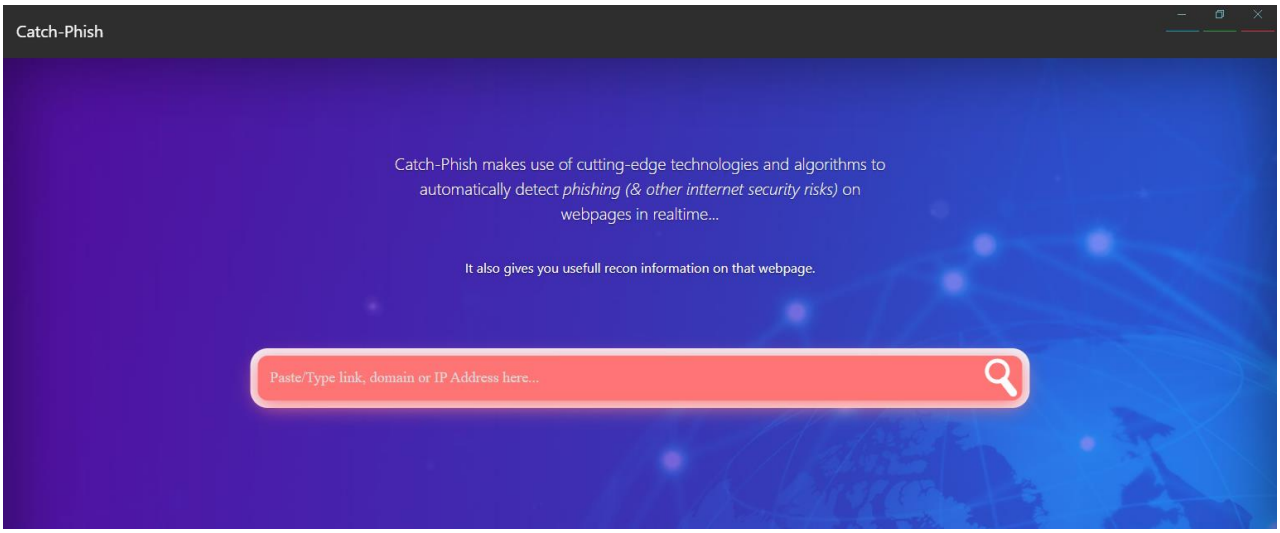

Fig. 11. *The catch-phish plugin user interface.*

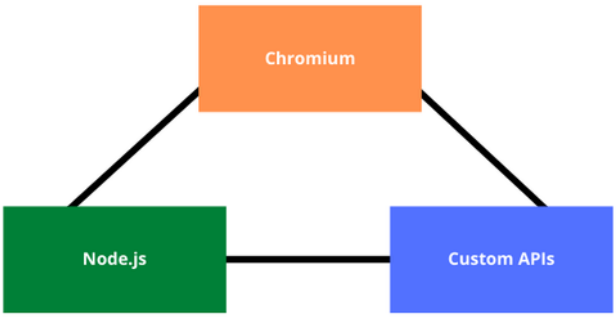

Fig. 12. *Structure of an electron.js application.*

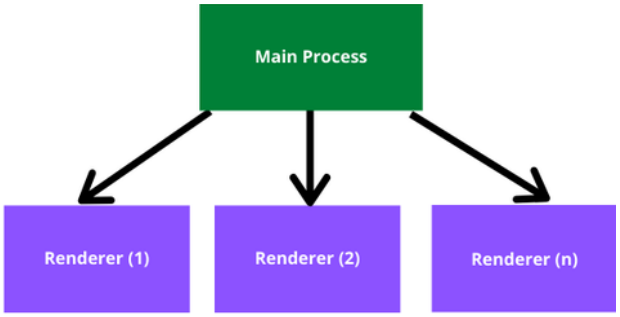

Fig. 13. *Relationship between electron.js main process and renderer processes.*

interaction with native functionality occurs. The Main process is responsible for creating web pages. It does this by creating a new instance of the Electron.js Browser Window object. This creates a new web page that runs in its own Renderer process.

The Main process can create more than one web page each running in its own Renderer process. Typically, Electron.js applications boot up with a default web page which is the app's start-up screen. More screens can be created if the application requires them. Each Renderer process manages its web page and is completely isolated from other Renderer processes and the Main process itself. Thus, if one Renderer process terminates, it does not affect another Renderer process. A Renderer process can also be terminated from the Main process by destroying its Browser Window instance. Out of the box, the Renderer process only has access to browser APIs like the window and document objects. This is because the Renderer process is simply a running Chromium browser instance. It can, however, be configured to have access to Node.js APIs such as process and require. Oftentimes, one may want to use native functionality in an Electron.js application in response to events, like a user clicking a button. However, because the Renderer process and the Main process are completely isolated from each other, native functionality cannot be accessed directly from the web page. To make this possible, Electron.js provides an Inter-process communication (IPC) channel that allows the Renderer process to communicate with the Main process and vice-versa as shown in Figure 13. Using the ipcMain and ipcRenderer modules for the Main process and Renderer process respectively, it is possible to emit events from one process and listen for events in the other process. It is also possible to pass data from one process to another.

F. The Backend/Application Logic

The backend logic of the Catch-Phish desktop application focuses on detecting malicious sites using IPQS live URL scanning through on-demand API requests. The URL scanning API evaluates a valid URL and returns over 20 data points that summarize its associated risk level. Additionally, geolocation data can be retrieved using the IP address obtained from the first API response. Both API requests operate asynchronously to enhance efficiency, with an 'await' clause ensuring the second request waits for the first response. It's important to note that the desktop application does not generate user advisories since users are not actively submitting details while using the application. Instead, it provides comprehensive insights, allowing users to make informed decisions. For instance, a domain's age and risk score are presented, highlighting potential concerns if the domain is less than a week old with a risk score

TABLE III. IPQS MUS API ADDITIONAL REQUEST OPTIONS.

Method	Value	Example
GET	key	?key=YOUR_API_KEY_HERE&url=https%3A%2F%2Fgoogle.com
POST	key	key=YOUR_API_KEY_HERE&url=https%3A%2F%2Fgoogle.com
Header	IPQS-KEY (Additional parameters passed as either GET or POST)	IPQS-KEY: YOUR_API_KEY_HERE

TABLE IV. IPQS MUS API ADDITIONAL REQUEST PARAMETERS.

Field	Description	Possible Values
strictness	How strictly IPQS scans the URL? Stricter checks may provide a higher false-positive rate. IPQS recommends defaulting to level "0", the lowest strictness setting, and increasing to "1" or "2" depending on your levels of abuse.	integer (0-2)
Fast	When enabled, the API will provide quicker response times using lighter checks and analysis. This setting defaults to false.	boolean, string (true or false)
timeout	The maximum number of seconds to perform live page scanning and follow redirects. If your implementation requirements do not need an immediate response, we recommend bumping this value to the default value of 2 seconds.	integer (1-10)

greater than 0%. Ultimately, the application displays the risk score and domain age of the entered URL or domain without issuing direct warnings. The backend process for handling user input and fetching results in the Catch-Phish desktop application is outlined as follows:

The requested URL for fetching results using the Axios utility in Electron is structured as: *https://ipqualityscore.com/api/json/url/YOUR_API_KEY_HERE/URL_HERE.*

When a user inputs a domain or full URL into the application, it must be URL encoded for proper processing. This is achieved using the code: *Await_fetch("https://ipqualityscore.com/api/json/url/API_KEY_HERE/"+encodeURIComponent(USER_URL_INPUT));.*

There are also alternative options for submitting requests to the IPQS API, particularly when platform requirements or frameworks necessitate not including the API key in the URL. Instead, the API key can be transmitted through GET, POST, or Headers, utilizing the specified endpoints as illustrated in Table III.

Custom tracking variables (such as "userID", and "transactionID") established in the developer account settings can be passed with each API request. This allows IPQS reporting tools to filter by specific users, products, campaigns, transactions, etc. so that it can easily match up records with the system to identify fraudulent activity. Additional request parameters were not used in the Catch-Phish Desktop application because they were not deemed needed. Table IV gives a detailed desiption and possible values of additional request parameters the IPQS MUS API allows.

G. Response Field Definition

The Malicious URL Scanner API returns many data points so that business logics can make the best decisions for their audiences. Analysing the overall Risk Score is usually the best way to determine domain reputation and the overall scoring confidence level. When this value is 100, there is 100% confirmed activity of phishing, malware, or similar abuse. Suspicious URLs can be identified with the "suspicious" data point or by analysing Risk Scores (30 – 80). URLs or domains with Risk Scores ≥ 85 are suspicious and likely to be a poor reputation domain

or malicious URL. Risk Scores ≥ 85 have been classified by IPQS deep machine learning as suspected of phishing, malware activity, or similar type of abuse. Risk Scores of 100 will provide confirmation the URL is accurately classified as a malicious link. It is recommended to block or flag a URL as malicious using a combination of the "risk score", "phishing", "malware", "suspicious", "parking", and "spamming" variables.

IV. Validation Results And Discussions

The browser plugin and desktop application in the Catch-Phish system was tested against various webpages and domains. Storage use and network performance of the browser plugin is also monitored using Chrome DevTools which is built directly into the Google Chrome browser, while for the desktop application, the Windows Task Manager is used. The first section defined the dataset used for testing and which metrics were of importance in the system's evaluation. Thereafter, the system is evaluated based on the selected metrics. Then, the results were analysed.

A. Dataset and Setup

The dataset used to test the Catch Phish system are based on the statistics gathered by PhishTank which contains URLs suspected or verified to contain phishing webpages submitted by different users. Testing CatchPhish applications focused on the dataset from recent submissions on PhishTank. At the time of this test, the phishing validity of some of the URL submissions used for testing were unknown while others were verified as phishing pages on Phish tank. The URL of the Google homepage was also used as input in the test to see if both applications have the problem of generating false positives. The datasetused for all tests were collected on the 27th of March, 2023 at 11:18 pm and on 7th of April, 2023 at 11:42 am from PhishTank using random selection. An overview of the input data and test results is given in Table V.

By defining the dataset, it becomes clear that the metrics required to verify the results must be determined. The system's goals are to collect and analyse the contents, features and patterns of webpages to detect security risks in them, especially phishing. The accuracy of the result is determined by checking if the results of Catch-Phish applications (Phishing status and risk score) match the phishing validity on PhishTank.

In the summary of phishing detection test result, 69.5% (16) of the total 23 URLs used as a dataset to test the Cath-Phish system's accuracy and reliability in identifying real phishing webpages were confirmed as authentic phishing pages by Catch-Phish, while 60.8% (14) were confirmed as such by PhishTank. The test results show that the system achieved 95.6% (22) accurate results and that Catch-Phish is more effective at identifying phishing websites, even zero-day websites, as it was able to authenticate certain URLs that had just been submitted (and were as-yet-unverified) to PhishTank. Due to the fact that it did not classify secure websites as phishing websites, the Catch-Phish can also be used to prevent false positives.

Normally, caching would be the best mode of optimization for both applications. In the context of this study, not enabling caching was the best way to optimize it. Caching would prevent fresh rendering of response every time a request is made so it was avoided to be able to catch zero-day phishing pages in real-time. Also, very few resources that had nothing to do with API responses (like icons and images) were used to prevent extended load time.

B. Testing the System

The system was tested using the data sets previously defined, and subsequent visualizations on the browser plugin and desktop application are shown in Fig. 14 and Fig. 15 respectively. In Fig. 14, when the plugin was used a webpage on the 'kalashpayments.com' domain which is a payment platform, the risk score was 61% which raises suspicion. When used on the 'mettechmetal.com.tr' domain that webpage was found to be a phishing page by all metrics as the risk score was 100. The webpage on 'google.com' domain revealed safe browsing activity. More insight on a particular webpage is displayed to the user as depicted in Fig. 15. A webpage might not be a phishing page but can have other risks like invalid or outdated domain certificates making such a page prone to man in the middle attacks. The user can deduce more about a page from the domain age, page size, content type etc. even though the system might not out rightly flag that page as suspicious or a phishing page.

C. Detection Speed

Google DevTools which is built into the Google Chrome web browser was used to get detailed insight into the performance of the Catch-Phish plugin in terms of time to load (speed). Fig. 16 reveals that when the time to load of the browser plugin was tested, the API request made to IPQS MUS took the most time (3.68 seconds) to respond. The total time it took for the system to respond and display the results was 3.69 seconds implying that the API request and response is responsible for 99.7% of the process time. It also implies that the plugin is very fast especially for one that does not implement caching.

TABLE V. URLS/WEBPAGES CONSIDERED FOR VALIDATION.

ID	Phish URL	Submitted	PhishTank Valid?	Catch-Phish Valid?
8093670	https://kalashpayments.com/bancadigitaloccidentaal/index.html added on Mar 27th 2023 10:08 PM	by lhernandez	Unknown	**VALID PHISH**
8093663	https://mettechmetal.com.tr/index2.html added on Mar 27th 2023 10:03 PM	by soclatam	**VALID PHISH**	**VALID PHISH**
NA	https://google.com/	NA	***INVALID***	***INVALID***
8109274	https://blazej.szymonpa.pl/a1legro/email@example.com... added on Apr 7th 2023 10:12 AM	by Amarena98	Unknown	**VALID PHISH**
8109273	https://regvc.vqkxqmw.cn/ added on Apr 7th 2023 9:57 AM	by Micha	**VALID PHISH**	**VALID PHISH**
8109272	https://iyhgh.acabe.cn/ added on Apr 7th 2023 9:57 AM	by Micha	**VALID PHISH**	**VALID PHISH**
8109271	https://trgmh.kegesi.cn/ added on Apr 7th 2023 9:56 AM	by Micha	**VALID PHISH**	**VALID PHISH**
8109270	https://ytutf.fj1144.cn/ added on Apr 7th 2023 9:55 AM	by Micha	**VALID PHISH**	**VALID PHISH**
8109269	https://jycv.qaxuacl.cn/ added on Apr 7th 2023 9:55 AM	by Micha	**VALID PHISH**	**VALID PHISH**
8109268	https://sdbs.life/ added on Apr 7th 2023 9:42 AM	by WilliamSeah	**VALID PHISH**	**VALID PHISH**
8109267	https://galxe.cash/airdrop/ added on Apr 7th 2023 9:31 AM	by phishb8	Unknown	**Unknown**
8109266	https://docs.google.com/presentation/d/e/2PACX-1vS-y9Mpzj-xETvkl6dpp5d... added on Apr 7th 2023 9:03 AM	by verifrom	**VALID PHISH**	**VALID PHISH**
8109265	https://eur02.safelinks.protection.outlook.com/?url=https://docs.googl... added on Apr 7th 2023 9:03 AM	by verifrom	Unknown	Offline
8109264	https://reglement-amendegouvcom.fr/ added on Apr 7th 2023 8:57 AM	by Nameshield	Unknown	***INVALID***
8109263	http://pecaruba070423.servequake.com/ added on Apr 7th 2023 8:50 AM	by D3Lab	**VALID PHISH**	**VALID PHISH**
8109262	https://www.smbc-cacd.ccm.ijwempw.cn/ added on Apr 7th 2023 8:49 AM	by soclatam	**VALID PHISH**	**VALID PHISH**
8109261	https://www.smbe-carb.ccm.aia82.top/ added on Apr 7th 2023 8:49 AM	by soclatam	**VALID PHISH**	**VALID PHSIH**
8109260	http://400517034274668.sepa-00980-force-drop.oa.r.appspot.com/?fbclid=... added on Apr 7th 2023 8:41 AM	by kkalmus	Unknown	Offline
8109259	https://hghsociety.com/wp-content/upgrade/godrts/... added on Apr 7th 2023 8:38 AM	by GovCERTCH	**VALID PHISH**	**VALID PHISH**
8109258	https://fragmentwalls.com/20bc23528edfb3741e0ad75d0cdbe336... added on Apr 7th 2023 8:36 AM	by GovCERTCH	Unknown	Unknown
8109257	http://m.cashwire.com/qban.xx?dXdWdhcc89ytcxLPFcGcR7cycBGprf77Wcbbb4P... added on Apr 7th 2023 8:36 AM	by GovCERTCH	**VALID PHISH**	**VALID PHIHS**
8109256	https://ip-178-118-29-33.main.jp/Paket-id-28193/Seleccione_medio_de_pa... added on Apr 7th 2023 8:36 AM	by GovCERTCH	Unknown	***INVALID***
8109255	https://116.62.202.222/ added on Apr 7th 2023 8:32 AM	by WilliamSeah	**VALID PHISH**	**VALID PHSIH**

Fig. 14. *Catch-Phish plugin visualizations (a) suspected but unverified phishing webpage (on kalashpayments.com) (b) verified phishing webpage (on mettechmetal.com.tr) (c) phish-free webpage (on google.com).*

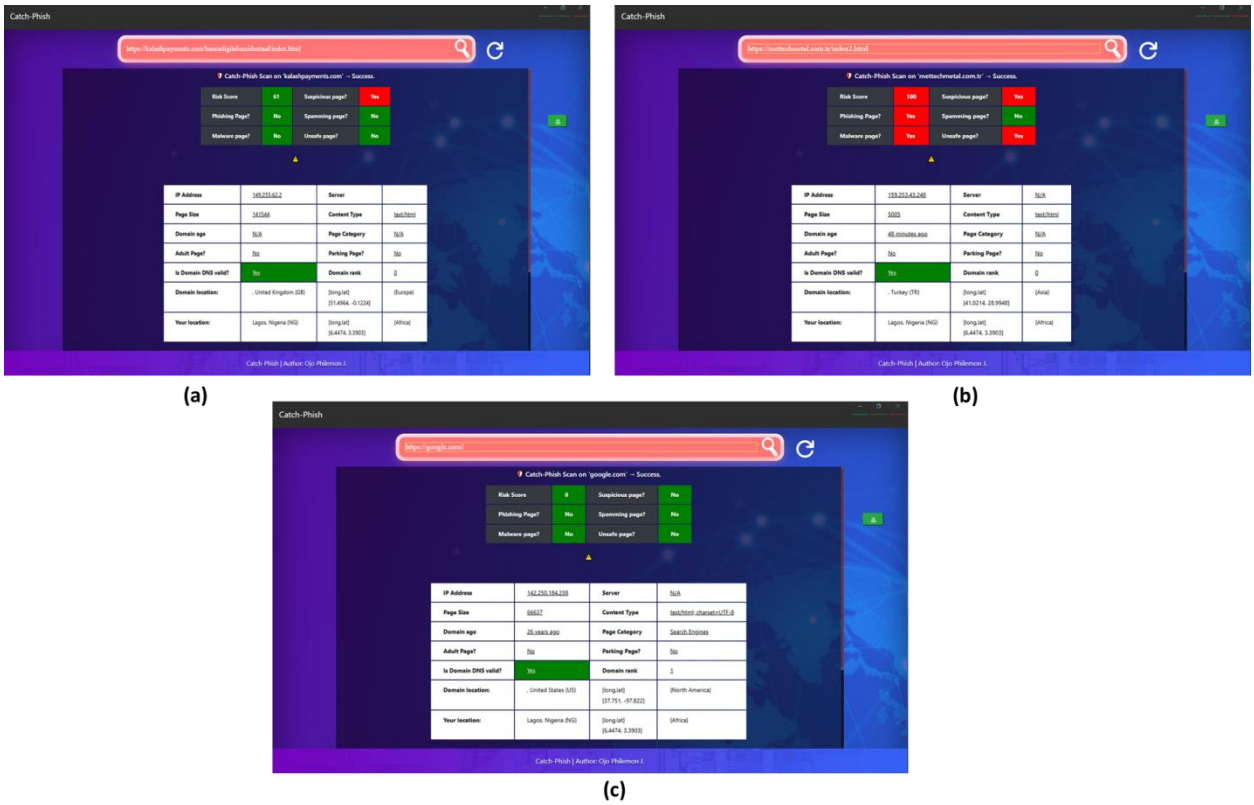

Fig. 15. *Catch-Phish desktop application visualizations (a) suspected but unverified phishing webpage (on kalashpayments.com) (b) verified phishing webpage (on mettechmetal.com.tr) (c) phish-free webpage (on google.com).*

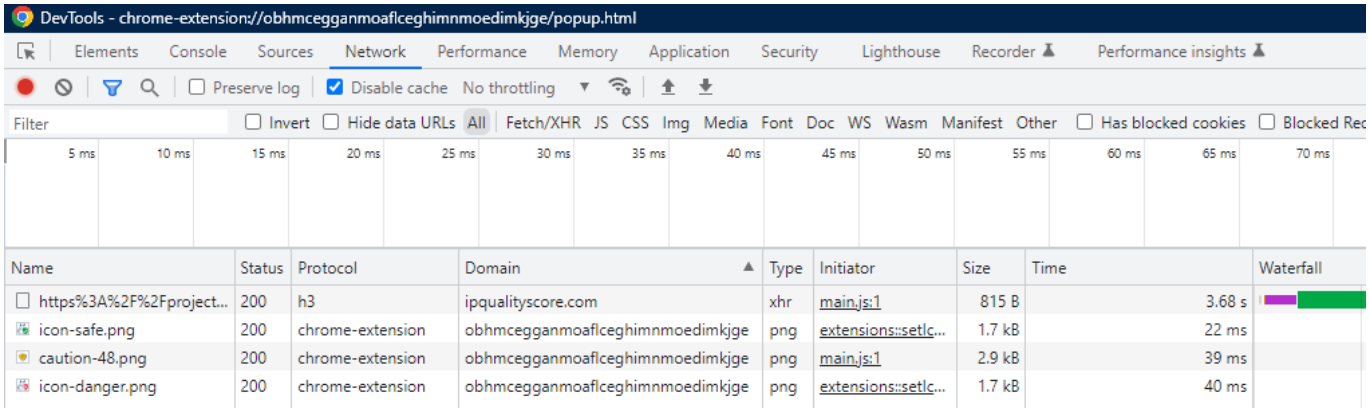

Fig. 16. *Google Devtools network analysis.*

V. Conclusion And Future Studies

Phishing is one of several types of cyber-attacks that use fraudulent web pages that appear legitimate to deceive users. Several approaches have been developed to detect this attack with varying level of success. In this study, a new phishing-detection system is proposed. The system is built as a desktop application using Electron.js, integrating both the frontend and backend, with the Google Maps API utilized for domain location visualization. The frontend of the web browser application was developed using Bootstrap, while Node.js served as the backend. Both applications interact with IPQualityScore's Malicious URL Scanner API, leveraging machine learning algorithms and querying up-to-date databases of phishing URLs to detect webpage risks and zero-day phishing threats. The design adheres to software engineering principles and best practices, ensuring accessibility and strong performance. Extensive testing was conducted on the developed system, applying it to multiple datasets. Experimental results demonstrated that automated and real-time detection of phishing attempts is feasible. The dataset used for testing comprised 20 phishing websites and 3 legitimate ones, with the system achieving an 80% accuracy rate in detecting phishing websites and 100% accuracy in identifying legitimate ones. Overall, the Catch-Phish system accurately classified 22 out of 23 URLs tested, resulting in an impressive overall accuracy of 95.6%. This affirms the system's potential to enhance internet security and accountability through automated phishing detection and information gathering. It is important to mention that the analysis and visualization of risk scores for URLs and webpages not hosted online may lead to misleading results; therefore, it is crucial for users to test against accessible websites to avoid false positives. This forms a potential future study.

References

[1] R. Alabdan. "Phishing attacks survey: Types, vectors, and technical approaches," Future Internet, vol. 1, pp. 1-38, 2020.

[2] Wallarm Inc (2023). What is phishing attack? Types and examples. [Online]. Available from: https://www.wallarm.com/what/types-of-phishing-attacks-and-business-impact. Accessed: [12/04/2023].

[3] J. Aljabri, N. Alzaben, N. NEMRI, S. Alahmari, S.D. Alotaibi, S. Alazwari, A.O. Khadidos, and A.M. Hilal. "Hybrid stacked autoencoder with dwarf mongoose optimization for phishing attack detection in internet of things environment." Alexandria Engineering Journal, vol. 106, pp. 164-171, 2024.

[4] N. Kamble, and N. Mishra. "Hybrid optimization enabled squeeze net for phishing attack detection." Computers & Security, vol. 144, p. 103901, 22024.

[5] A.K. Yamarthy, and C. Koteswararao. "MDepthNet based phishing attack detection using integrated deep learning methodologies for cyber security enhancement." Cluster Computing, vol. 27, pp. 6377–6395, 2024.

[6] E. Benavides-Astudillo, W. Fuertes, S. Sanchez-Gordon, D. Nuñez-Agurto, and G. Rodríguez-Galán. "A phishing-attack-detection model using natural language processing and deep learning," Applied Sciences, vol. 13, *pp.* 1-23, 2023.

[7] M.W. Shaukat, R. Amin, M.M.A. Muslam, A.H. Alshehri, and J. Xie. "A hybrid approach for alluring ads phishing attack detection using machine learning," Sensors, vol. 23, pp. 1-27, 2023.

[8] Anti-Phishing Working Group. (2020). Phishing activity trends report, 1st Quarter 2020. Online. Available from: https://docs.apwg.org/reports/apwg_trends_report_q1_2020.pdf. Accessed: [29/03/2023].

[9] H. Mohammad, and A. Gulzar. "A multivocal literature review on growing social engineering based cyber-attacks/ threats during the covid-19 pandemic: Challenges and prospective solutions," IEEE Access, vol. 9, pp. 7152-7169, 2021.

[10] X. Xu, Y. Liu, and Y. Xu. "Detecting phishing websites using visual similarity based on visual cryptography," Journal of Internet Technology, vol. 8, pp. 865-872, 2017.

[11] I. Bulakh. "Phishing: current state of problems and ways of combating it," In: proceedings of the IEEE 9th International Conference on Dependable Systems, Services and Technologies, 24-27 May, Kyiv, UA, pp. 216-220, 2018.

[12] European Union. (2016). General Data Protection Regulation (GDPR). Online. Available from:https://eurlex.europa.eu/legalcontent/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN. Accessed: [27/03/2023].

[13] K. Sood, P. Bansal, and A. Abraham. "A survey on phishing detection techniques," Journal of Network and Computer Applications, vol. 127, pp. 59-76, 2019.

[14] R. Arora, and N. Arora. "Phishing attack techniques," International Journal of Computer Science and Technology, vol. 5, pp. 4-6, 2014.

[15] B. Amro, A.H. Abusabha, I.I. Najjar, and A.H. Quneibi. "PAPG – Personalized anti-phishing guard," International Journal of Computing and Network Technology, vol. 7, pp. 8-9, 2019.

[16] K. Sahu, and S. Shrivastava. "Kernel k-Means clustering for phishing website and malware categorization," International Journal of Computer Applications, vol. 111, pp. 20-25, 2015.

[17] V. Shahrivari, M. Darabi, and M. Izadi. "Phishing detection using machine learning techniques," arXiv preprint arXiv:2009.11116, 1-9, 2009.

[18] A. Basit, M. Zafar, X. Liu, A.R. Javed, Z. Jalil, and K. Kifayat. "A comprehensive survey of AI-enabled phishing attacks detection techniques." Telecommunication Systems, vol. 76, pp. 139-154, 2021.

[19] A. Ahmadian, M. Abadi, and M. Shamsfard. "A novel algorithm for phishing URL classification," Journal of Ambient Intelligence and Humanized Computing, vol. 11, pp. 4359-4371, 2020.

[20] A.M. Ihaisoni, and S. Khan. "Phishing detection using machine learning classifiers," International Journal of Advanced Computer Science and Applications, vol. 10, pp. 101-105, 2019.

[21] A. Singh, A. Rathi, and P. Sharma. "Phishing website detection using machine learning," International Journal of Engineering and Advanced Technology, vol. 10, pp. 1133-1137, 2021.

[22] M.W. Shaukat, R. Amin, M.M.A. Muslam, A.H. Alshehri, and J. Xie. "A hybrid approach for alluring ads phishing attack detection using machine learning," Sensors, vol. 23, pp. 1-27, 2023.

[23] Y. Zhang, and L. Li. "Real-time phishing website detection based on a machine learning

algorithm," Journal of Network and Computer Applications, vol. 154, p. 102662, 2020.

[24] R. Kaur, and G. Kaur. "Phishing website detection system using machine learning algorithms," International Journal of Engineering & Technology, vol. 8, pp. 74-80, 2019.

[25] A. Gupta, and A. Jain. "Real-time phishing website detection using support vector machine," In: Proceedings of the 2020 International Conference on Computing, Communications and Networking Technologies (ICCCNT), 1-3 July, Kharagpur, India, pp. 1-7, 2020.

[26] S. Bhadoria, and N. Singh. "Real-time phishing website detection using machine learning algorithms," In: Proceedings of the 2020 International Conference on Innovations in Computer Science and Engineering (ICICE), 28-29 August, Hyderabad, pp. 1-7, 2020.

[27] A. Singh, A., Rathi, and P. Sharma. "Phishing website detection using machine learning," International Journal of Engineering and Advanced Technology, vol. 10, pp. 1133-1137, 2021.

[28] Z. Alshingiti, R. Alaqel, J. Al-Muhtadi, Q.E.U. Haq, K. Saleem, and M.H. Faheem. "A deep learning-based phishing detection system using CNN, LSTM, and LSTM-CNN. Electronics, vol. 12, p. 232, 2023.

[29] R. Zaimi, M. Hafidi, and M. Lamia. "A deep learning approach to detect phishing websites using CNN for privacy protection." Intelligent Decision Technologies, vol. 17, pp. 713-728, 22023.

[30] C. Kunndra, A. Choudhary, J. Kaur, A. Jogia, P. Mathur, and V. Shukla. "NTPhish: A CNN-RNN hybrid deep learning model to detect phishing websites." In International Conference on Cryptology & Network Security with Machine Learning. Springer Nature, Singapore, pp. 587-599, 2023.

[31] P. Korus, P. Wroblewski, and K. Dembczynski. "Combining convolutional neural networks and decision trees for improved phishing detection," In: proceedings of the 35th ACM/SIGAPP Symposium on Applied Computing, March 30 - April 3, Brno, CZ, pp 1046-1053, 2020.

[32] T.V. Nguyen, H.T. Pham, and T.H. Nguyen. "A real-time phishing website detection system using deep learning and blockchain," In: proceedings of the International Conference on Advances in Computer Science and Information Technology, June 27-28, Copenhagen, DK, pp. 1-7, 2020.

[33] X. Zhu, L. Li, and Y. Shen. "An improved phishing detection method based on convolutional neural network," in Proceedings of the IEEE International Conference on Artificial Intelligence and Computer Applications, Guangzhou, China, pp. 10-14, 2020.

[34] H. Kim, H. Park, and K. Lee. "A real-time phishing website detection system using deep learning," Computers & Security, vol. 77, pp. 138-149, 2018.

[35] N. Ahmadi, and A. Ghorbani. "A new approach to detect phishing websites using deep learning algorithms," Journal of Ambient Intelligence and Humanized Computing, vol. 10, pp. 221-230, 2019.

[36] J. Lee, S. Park, M. Kim, and B. Kang. "Phishing detection with recurrent neural networks and graph-based features," Applied Sciences, vol. 11, p. 3229, 2021.

[37] S. Mehmood, M. S. Khalid, and S. Saeed. "Detecting phishing emails using recurrent neural networks," in Proceedings of the IEEE International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Sukkur, Pakistan, pp. 1-6, 2020.

[38] T. Pham, T. Nguyen, and T. Nguyen, "Phishing emails detection based on recurrent neural network," in Proceedings of the International Conference on Advanced Computing and Applications (ACOMP), Ho Chi Minh City, Vietnam, pp. 110-114, 2021.