Scientific Research Publishing

Journal of Information Security, 2024, 15, 474-493
https://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

PhishGuard: Integrating Fine-Tuned Large Language Models (LLMs) into Password Management

Smith Patel, Vijay K. Madisetti

School of Cybersecurity & Privacy (SCP), Georgia Institute of Technology, Atlanta, Georgia, USA
Email: smith.patel@gatech.edu, vkm@gatech.edu

How to cite this paper: Patel, S. and Madisetti, V.K. (2024) PhishGuard: Integrating Fine-Tuned Large Language Models (LLMs) into Password Management. *Journal of Information Security*, **15**, 474-493.
https://doi.org/10.4236/jis.2024.154027

Received: July 2, 2024
Accepted: October 7, 2024
Published: October 10, 2024

Abstract

In the digital age, phishing attacks have been a persistent security threat leveraged by traditional password management systems that are not able to verify the authenticity of websites. This paper presents an approach to embedding sophisticated phishing detection within a password manager's framework, called PhishGuard. PhishGuard uses a Large Language Model (LLM), specifically a fine-tuned BERT algorithm that works in real time, where URLs fed by the user in the credentials are analyzed and authenticated. This approach enhances user security with its provision of real-time protection from phishing attempts. Through rigorous testing, this paper illustrates how PhishGuard has scored well in tests that measure accuracy, precision, recall, and false positive rates.

Keywords

Phishing Attacks, Password Management, Phishing Detection, AI, BERT Algorithm, Real-Time Protection, Cybersecurity, URL Authentication

1. Introduction

The dependence on digitization is rising when activities pertaining to personal as well as professional matters are being digitized, thus creating a larger quantum of the necessity of effective cybersecurity measures. Among several cyber threats, phishing is unique in nature, deceptive, and targets sensitive data such as login credentials and financial details. When phishing techniques become sophisticated, they are evasive in nature and can easily pass through the conventional means of detection, leaving the existing defenses inadequate.

Even when password managers are well-adopted to secure and manage

authentication credentials, they inherently lack the functionality to determine the legitimacy of such sites where authentication credentials are used. Such a gap exposes a user to a high risk of phishing attacks because a password manager, during input of sensitive information, does not usually inspect or validate the URLs. Consequently, even a careful user that relies on the tools for security can fall victim to such well-crafted phishing sites.

The current tools and approaches, such as password managers, are designed with the sole focus of securely storing and managing passwords, irrespective of the context in which those passwords may be used. This gap allows phishing attacks to remain very potent since most tools do not function outside of the broader security landscape that includes website verification.

Here, we introduce a novel solution, PhishGuard, to bake-in phishing detection directly into the password management process. PhishGuard goes ahead and boosts password managers with real-time phishing detection, thus demonstrating a great stride in user-centric cybersecurity measures. The following will be covered in this paper:

1) Current techniques in password management and phishing detection: their limitations.

2) A discussion of PhishGuard system, focused on the integrated approach and technologies to enable such an approach.

3) Detailed analysis of PhishGuard performance, regarding the phishing attack detection in comparison to previous methods.

PhishGuard tends to be a proactive security tool that does not just manage the password effectively but also assures that the credentials are used in a safe and verified environment. The integration of sophisticated AI models capable of carrying out real-time detection of phishing URLs by Phish-Guard should drastically reduce instances of phishing attacks and thus protect the critical information of the users more effectively than ever before.

2. Existing Work

Currently, many methods and approaches have been developed to address this problem, ranging from very simple security features built into web browsers to quite sophisticated AI algorithms. We discuss existing phishing detection methods and their integration into password management systems, highlighting both the pros and cons of these methods.

2.1. URL Blacklisting

URL blacklisting keeps a record of phishing URLs or domains that are identified as malicious after the same characteristics have been identified before in other URLs. These blacklists are mostly compiled and updated by cybersecurity organizations, companies, or communities keeping track and reporting phishing activities. As and when any URL is accessed, it can be checked against the blacklisted URL, and in case of a match, it would block access and give a warning to the user.

Widely used methods for this are blacklists, among which the most prominent is Google Safe Browsing. It maintains a blacklist of URLs and warns the users in case a URL is reported under suspicion of a phishing attack. Prakash *et al.* [1] introduced Phishnet, a predictive technique to detect phishing URLs. It used to judge phishing URLs by inspecting features like directory structure, corresponding IP address, and brand names of the already blacklisted URLs. In a related approach, Felegyhazi *et al.* [2] have proposed a method that compares the domain name and the name server information of the newly suspected URLs with those on the blacklist for classification. Sheng *et al.* [3] found that after a long delay, a bogus domain was being blacklisted, and between 50% - 80% of these were listed post-attack. Being that new fraudulent websites are created in thousands daily, this makes the blacklist have to be frequently updated by its source, making it in turn more effective in using the AI detection-based approaches in fighting the offenses of phishing.

The major weakness of blacklisting is that it is a reactive approach in the sense that the human users or systems must find and report the phishing sites. Integrating URL blacklisting within the password management system provides an additional layer of security, to protect the automatic entry of credentials within known phishing sites. Yet the strength of any such method within a password manager is directly proportional to the strength and timeliness of the blacklist; users and developers may rely on the high update rate and great effective scope to which the method is effective.

2.2. Two-Factor Authentication (2FA)

Two-factor authentication (2FA) is a process in which the user provides two different types of authentications to access an account or system: usually what they know (a password or pin) and something they own or are (a smartphone or fingerprint). The integration of 2FA in phishing detection, especially through password management systems, comes as a robust security level, meant to reduce related risks of the compromise of credentials.

A very effective tactic in phishing prevention is being challenged by a specific technique of cyber-attack—Reverse Proxy [4]. This method appears to compromise all 2FA protections in the tracking and generation of traffic that is exchanged between a user and the original website. During this kind of attack, the attacker places a reverse proxy server that acts like an intermediary between the user and the actual website. The user is supposed to request the website from the proxy server when he wants to access the website. The actual website is requested in turn by the proxy and the website's response is captured by the proxy server and sent to the user.

This allows the reverse proxy to intercept and capture the user's login credentials, including any 2FA tokens, before passing them on to the real website. The attacker can now log in with those stolen credentials, masquerading as the real user, completely defeating the 2FA protection. The user is totally unaware

because, for all intents and purposes from the user's point of view, it is as if they are interacting with the real website, but in reality, the proxy server.

So, in short, 2FA provides very strong additional security and could go a long way in reducing risks with phishing. However, it is not free of its challenges and limitations: it may add complexity to user experiences and raise points of potential failure.

2.3. Browser Extensions

Most of the phishing detection browser extensions operate by the user's URL visits and checking the URLs against real-time databases for known phishing sites. Heuristic analysis is also performed to identify some suspicious patterns or features in websites that typically would be associated with phishing activities, for example, the disparity in the URL, a form requesting sensitive information, and a security certificate mismatch.

Most browser extensions make use of a long list of very stale blacklists and therefore, they will not block access to the latest cases of phishing until new ones are discovered and added to the list of known websites [5]. As with any other automated system, it may sometimes notify legitimate sites as phishing (false positives) or fail to notify genuine phishing sites (false negatives). Sometimes, browser add-ons require access to information on all pages that the users visit, which might sound inappropriate to privacy concerns, particularly in the case of an add-on that gathers or shares such information with other parties without clear user agreement. Browser extensions could affect system performance, especially when many are running because of the processing power and memory used.

Therefore, phishing detection by browser add-on is subject to out-of-date blacklists and opens serious concerns of privacy and performance. These are serious concerns. Therefore, users must make sure to choose trusted add-ons and be aware of these limitations. Ideally, use in concurrence with other security practices.

2.4. AI Models

The design of algorithms that can separate phishing websites from the legitimate ones is a task that has adopted several approaches based on AI techniques. In general, the feature collection, quality of the training dataset, and performance of the applied classification algorithms are the three major reasons for the success of such methods. Features are collected from a multitude of sources that include URLs, webpage contents, and third-party services. Therefore, access to some heuristic features is challenging and time-consuming; thus, some of the machine-learning strategies end up demanding large computational resources for the extraction of such features.

Jain and Gupta [6] proposed an anti-phishing mechanism that takes its features from both the URL and the source code of a webpage in a third-party service-independent manner. The approach has shown great accuracy in the identification

of phishing webpages while using a relatively small dataset (2141 phishing and 1918 legitimate webpages). The same researchers [7] developed another anti-phishing method that checks for phishing attacks by analyzing hyperlinks that are part of the HTML page. Since this technique is implemented on the client side and is language independent, it can be applied universally. However, it depends only on the HTML content of the webpage and may classify webpages as phishing if an attacker changes all the webpage resource references, such as Javascript, CSS, and images. Rao and Pais [8] have come up with an anti-phishing technique called BlackPhish, which works in a two-tiered manner. Blackguard is a two-tier approach: the first tier builds a blacklist through the use of visual similarity features, more specifically, file names, paths, and screenshots, instead of the classical URL blacklist. The second tier extracts heuristic features both from the URL and from the HTML to find phishing websites that could go through the first filter. In all cases, though, legitimate websites pass through the two-level scrutiny. Some other studies [9] implemented a search-engine-based approach in the first level of authentication. So, the authenticity of a webpage is first ensured at the first level, and in the second level, the investigation is done using different hyperlinks inside the HTML of the page to find phishing. While the search engine-based approaches generally improve the accuracy of identification of legitimate sites, these are prone to more misclassification of legitimate sites as phishing, in particular when newly established genuine sites do not turn up at the top in the search results. All these search-based approaches assume that a legitimate website should appear at the forefront in the results of a search engine.

Recently, Rao *et al.* [10], proposed a new method for the detection of phishing websites based on plain text and domain-specific text word embeddings in HTML source code. They had employed a lot of word embedding techniques to test their model using the ensemble and multimodal approaches. In this work, however, is based only on textual content, which could not work for a page where images are used in place of text. At the same time, other works focus on the identification of a phishing attack by analyzing various hyperlink relationships of web pages. Guo *et al.* [11] proposed a method of detection, named HinPhish, which is based on a constructed Heterogeneous Information Network (HIN) based on domain nodes and resource-loading nodes. The method also defines four hyperlink categories and three types of relationships among them—external, empty, internal, and relative links. They then used the authority ranking algorithm to evaluate the strength of these links and to attach an absolute numerical weight to each node.

In another study developed by Sahingoz *et al.* [12], they implemented an approach that used distributed representation of words in some URL-specific and passed through seven diverse AI classifiers to ascertain whether a URL can be a phishing site. In a similar vein, Rao *et al.* [13] proposed an anti-phishing approach, CatchPhish, extracting hand-crafted features and Term Frequency-Inverse Document Frequency (TF-IDF) from URLs, after which a classifier was used and trained with the random forest algorithm. Though these methods perform adequately, they

still suffer from the following shortcomings: 1) they perform inadequately for unrecognized characters in the URLs, which usually consist of meaningless and unknown words not seen in the training data; 2) they fail to capture the actual content of the website, hence some of these unique URLs but closely resembling known legitimate sites will not be flagged by the classifiers. The basis of this intuition is that it's not always possible to properly detect the phishing site just based on the features of the URL. Still, we compensate for this drawback by introducing a novel approach, which employs three different types of features to properly detect phishing websites. More precisely, URL character sequences are used with information conveyed by different hyperlinks and features based on textual content to get a hybrid feature set.

Some of the deep learning techniques for phishing detection include Convolutional Neural Networks (CNN), Deep Neural Networks (DNN), Recurrent Neural Networks (RNN), and Recurrent Convolutional Neural Networks (RCNN), since they worked in Natural Language Processing (NLP). Nevertheless, the large training times that are always part and parcel of deep learning normally go on to limit the application of such methods in phishing detection. Aljofey *et al.* [14] utilized a character-level CNN for URL analysis and thereby designed the phishing detection strategy. The approach has been experimented with on diverse machine and deep learning algorithms by using feature sets such as TF-IDF of characters, count vectors, and manually crafted features. Following this, Le *et al.* [15] then introduced a technique called URLNet, which extracts both character and word level features from a URL and uses CNNs for training and testing. Chatterjee and Namin [16] built a model for the detection of phishing URLs by applying deep reinforcement learning to figure out phishing URLs in their balanced dataset between benign and malicious URLs. Fourteen handcrafted features were extracted by the researchers to train their model. More recently, Xiao *et al.* [17] proposed another novel technique for phishing web page detection, CNN–MHSA, in which a CNN is applied to extract the character feature from the URL and an MHSA mechanism with multiple heads is applied to calculate the weights of the features extracted by the CNN. Zheng *et al.* [18] came up with a new method that employed the Highway Deep Pyramid Convolutional Neural Network (HDP-CNN) to combine the character and word embedding features for classifying the legitimacy of URL. However, these methods are effective in the sense that they overclassify a few benign sites as phishing, specifically for phishing sites that are hosted on the compromised server because these methods tend to mainly focus on the analysis of URL features alone.

Many techniques used in dealing with phishing have been scrutinized in this section, and it is shown that though each has some merit, they also have a significant level of limitations, which makes them unable to fully provide complete protection to users against new and sophisticated phishing attacks that are cropping up. In some of the early research, human intervention was required for feature extraction, which is an overhead process since the features have to be tuned

according to the data set. This can cause the anti-phishing solutions to become overfitted, hence lowering their effectiveness.

3. Proposed Approach

PhishGuard takes a new approach towards password management, integrating sophisticated phishing detection into its framework. PhishGuard is all about having that single cover on safe password management and, at the same time, providing proactive legitimacy checking of websites when doing logins. This all-in-one solution consequently fills the large gap in contemporary password management systems that can't give real-time, real-time phishing detection.

At the core of phishing detection in PhishGuard is a BERT (Bidirectional Encoder Representations from Transformers) LLM model fine-tuned for phishing detection. It works in real-time over the URL and web content to identify features of a phishing site, extensively trained over a wide array of datasets that cover many phishing attack vectors. The model runs through URLs as they are entered into the password manager, or while navigating in the browser. This now becomes a real-time analysis that enables PhishGuard to alert users whenever such suspicious activities are done, therefore avoiding the chance of phishing before sensitive information is leaked. The PhishGuard is fitted with a Command Line Interface (CLI) whereby it suits newbies and experts; it gives simple command support to manage one's password and real-time alerts about URLs showing suspicious activities. Data for the user is kept secure using cutting-edge encryption technology, even when being stored together with the AI models in use by PhishGuard. The hashing algorithm used in the derivation of the encryption keys includes a user's master password and a device-specific secret. This makes user data secure even if the data protection mechanism is breached.

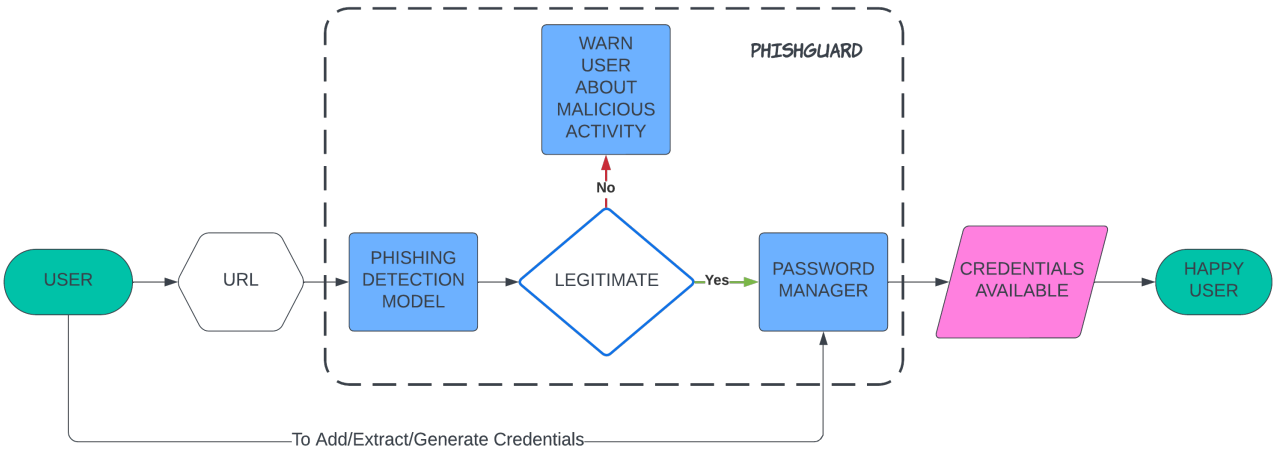

Figure 1. Model diagram.

The operational flow of PhishGuard is shown in (**Figure 1**):

- The BERT model checks at runtime, with the attached URL in the input, whether the addition or retrieval of the entry is a legitimate one.

- If a URL is detected as a phishing site, the user should be alerted, and the credentials being stored or auto filled should be stopped to prevent phishing attacks.
- PhishGuard updates its anti-phishing algorithms constantly through AI to always stay ahead of the changing techniques by those who conduct phishing.

Improvements upon Current Solutions:

- By building PhishGuard directly into the password-management workflow, the tool is moved to a proactive position of preventing phishing attempts as they are carried out in real-time, rather than to the current most reactive security measures.
- Unlike any solution based solely on blacklisting or heuristic analysis, the state-of-the-art machine-learning model used by PhishGuard provides broad coverage over a vast number of various phishing tactics, some being newly developed or unusual, without the reliance on user-reported attacks or updating of blacklists.
- PhishGuard packs in great security without detracting anything from the user experience. It embeds features of security seamlessly into an environment already comfortable to the user experience of a password manager, minimizing friction for the user while delivering advanced security to the lay audience.

4. Implementation and Test of Proposed Approach

4.1. System Architecture Overview

PhishGuard integrates a sophisticated architecture designed to enhance password management systems by incorporating advanced phishing detection capabilities. At the heart of this integration is the use of a BERT (Bidirectional Encoder Representations from Transformers) model, specifically fine-tuned for phishing detection. This section outlines the comprehensive architecture of PhishGuard, emphasizing the application of the BERT model in analyzing and authenticating URLs.

PhishGuard consists of several core components that work in tandem to provide a secure and user-friendly experience:

- **User Interface (UI):** The system employs a command-line interface (CLI), which facilitates direct interaction between the user and the password manager. This interface supports various commands for managing passwords and receiving real-time alerts on phishing attempts.
- **Encryption Mechanism:** Security is paramount in PhishGuard. The system uses AES-256 encryption, which is initiated by a Master Key derived from a combination of a user-inputted master password and a device-specific secret. This Master Key is crucial for encrypting sensitive data such as passwords, emails, and usernames, while less sensitive data, like site names and URLs, are stored in plaintext but monitored for authenticity.
- **Phishing Detection Module:** This is where the BERT model comes into play. PhishGuard utilizes a BERT model that has been fine-tuned on a dataset

specifically designed to identify phishing URLs. The model assesses each URL entered into the system to determine whether it is safe or potentially malicious.

Application of the BERT Model for URL Analysis and Authentication:

The BERT model used in PhishGuard is a pivotal element in the phishing detection module. BERT models are known for their exceptional performance in natural language processing tasks because they can understand the full context of a word by looking at the words that come before and after it, which is achieved through a mechanism called attention.

- **Model Details and Fine-tuning:** The specific BERT model integrated into PhishGuard, has been specially trained to recognize phishing URLs. The training involved adjusting the weights of the pre-trained BERT model on a labeled dataset comprising both phishing and non-phishing URLs, enabling it to predict the legitimacy of new, unseen URLs.
- **Real-Time URL Validation:** When a user attempts to store or access credentials in PhishGuard, the URL associated with these credentials is instantaneously sent to the BERT model. The model processes the URL and evaluates its features based on the learned patterns from its training. It then outputs a probability score indicating the likelihood of the URL being phishing or legitimate.
- **Integration with Password Management:** If the BERT model identifies a URL as phishing, PhishGuard automatically blocks the credential interaction and alerts the user with a warning. This integration ensures that users are proactively prevented from entering their credentials into potentially harmful sites, significantly enhancing the security of the password management process.

Technical Specifications and Data Flow:

- **Data Input:** Users input URLs through the CLI when adding or accessing stored credentials.
- **Data Processing:** The URL is encrypted and passed to the BERT model. The model uses tokenization to convert the URL into a format it can process, applying its fine-tuned knowledge to analyze the URL's structure and content.
- **Decision Making:** Based on the analysis, the BERT model classifies the URL and returns a decision that triggers an immediate response from the system—either continuing with the operation or blocking the transaction and issuing a security alert.

The architecture of PhishGuard represents a robust integration of traditional password management functions with advanced machine learning-based phishing detection. By leveraging the power of the BERT model, PhishGuard not only secures sensitive user data but also significantly enhances the overall security posture by preventing phishing attacks in real-time. This proactive approach marks a significant advancement in the domain of cybersecurity, setting a new benchmark for integrated security solutions.

4.2. Implementation Process

Key steps in the development of PhishGuard included:

- **CLI Development:** CLI, developed through Python, is robust yet simple so that it allows effective interaction between users and systems. Major emphasis was put on making the commands intuitive and the interface such that it can handle multiple kinds of user input without crashing.
- **Integration of the BERT Model:** The BERT LLM model was integrated into the password management workflow. At the time of a get or add operation, a user would try adding or retrieving an entry, and the URL would be passed through the BERT model, which evaluated its legitimacy and provided a risk assessment in a matter of milliseconds.
- **Encryption and Data Storage:** This will detail the protocols in place that ensure the user data in the system is encrypted and safe from unauthorized access. The system encrypts data at the point of entry and only decrypts when a user demands verification.
- **Setup:** A master password is requested from the user, securely hashed, and stored. A device-specific secret is also generated and stored locally in parallel to add security to the master key.
- **Entry Management:** Easily add credentials for various sites with just a few simple commands. Each entry is added by using the master password for authentication and then encrypting the provided credentials into a local database. The system also allows a search for entries through the name of the site, URL, or username.

4.3. Testing and Results

To ensure PhishGuard was powerful and dependable, it was developed with rigorous testing in mind:

- **Testing Environment:** The developed model was tested in a testing environment with a huge and varied dataset of 10,000 URLs. URLs within the dataset were categorized into those that were legitimate or phishing to simulate real-world usage of the classifier, providing a good measure of its performance across various attack vectors.
- **Performance Metrics:** The BERT model showed very good performance in terms of an accuracy of 98.39%, precision of 98.68%, and recall of 97.49%. The system has also been able to maintain a low rate of false positives standing at 0.95%, meaning a high level of reliability in locating phishing sites without the mislabeling of legitimate URLs.
- **User Experience Feedback:** Preliminary usability testing of the user interface, especially the CLI and alert messages for phishing detection, was conducted. The feedback from this session was then further used in the interface refinement and the clarity of instruction.

Appendix section contains some of the high-quality screenshots of the CLI in action (See Figures A1-A4), which demonstrates the processes involved to add entries, real-time detection alerts of phishing, and retrieval of stored entries. These images show, in addition, the operation of PhishGuard and its pleasant-looking

user interface with an efficient operation.

The development and testing of PhishGuard in this chapter have shown that the system attains the proposed objectives; however, it is far beyond any known conventional password manager with respect to security and usability. This facility of real-time phishing detection within the process of password management is one highly significant improvement over other security tools and is the way which users of the current era can manage credentials online in a much more secure and reliable manner.

5. Comparison with Prior Work

The performance of PhishGuard cannot mean a thing unless that performance is measured against the existing cybersecurity solutions. This will be compared with the approach that PhishGuard uses to overcome the limitations that were found within traditional password managers and phishing detection systems. This approach clearly maps the previous work and clearly indicates the additional functionality that PhishGuard brings to the area of real-time phishing detection within a password management framework.

5.1. Comparison with URL Blacklisting

PhishGuard goes one step further than the usual traditional blacklisting of URLs. As described previously, URL blacklisting relies solely on how current the updated phishing databases are, and therefore is completely reactive: new phishing sites receive a routine "grace period" of time before they are tracked and then finally blacklisted. Whereas, PhishGuard uses a real-time analysis model that does not rely on known blacklists, it dynamically checks URLs with differences in characteristics, which, therefore, greatly reduces reliance on update frequencies and also increases the immediate detection capability.

5.2. Enhancements over Two-Factor Authentication (2FA)

Although the 2FA adds the critically important layer of security requiring an additional step of verification, it does not tackle the initial stage of deception when users give away their credentials to the phishing sites. PhishGuard integrates directly into the password entry process, where it analyzes URLs before sensitive information is submitted. This is an important form of preemptive detection, particularly in case 2FA could be bypassed through advanced mechanisms of phishing, such as Reverse Proxy attacks, against which PhishGuard is designed to guard by preventing the submission of credentials to suspicious sites.

5.3. Advancements beyond Browser Extensions

For instance, browser-based anti-phishing solutions are always limited as they entail the installation process by users and hence can have a substantial effect on browser performance. These solutions are also subject to issues related to privacy concerns and the lag with which it is possible to update their blacklist databases.

PhishGuard has no such browser constraints and instead works by giving a one-non-compromising solution within one "extension". URL checks happen in the background of the integration of the password manager, thus providing protection seamlessly without any extra actions from the user.

5.4. Superiority to Existing AI Models

Most of the state-of-the-art machine-learning phishing-detection approaches necessitate a huge consumption of computational resources and, as a consequence, are not directly implementable within password managers, highly limiting practicability. The second challenge is the need to provide an optimal balance between accuracy and performance, hence resource efficiency; the implementation of PhishGuard's AI model, especially with the BERT algorithm fine-tuned for phishing detection, within this password manager environment, ensures a seamless user experience without the much higher resource demands usually associated with sophisticated AI models.

6. Limitations and Areas for Improvement

While PhishGuard introduces a significant advancement in integrating phishing detection with password management, it is crucial to recognize potential limitations and areas for improvement. This analysis focuses on its performance in large-scale deployments and its ability to handle complex URL structures.

6.1. Performance in Large-Scale Deployment

- As PhishGuard relies on a fine-tuned BERT model, the computational resources required for real-time analysis can be substantial. While testing has shown impressive results, the scalability of such a system in a large-scale, real-world deployment could encounter challenges, such as increased latency and the need for substantial computational power, especially when handling multiple simultaneous users.
- Effective deployment of PhishGuard in larger environments may require dedicated hardware or cloud-based scaling solutions to handle the processing load. This could increase operational costs and complexity, potentially limiting accessibility for smaller organizations without the necessary infrastructure.

6.2. Handling of Complex URL Structures

- Advanced phishing attacks may use URLs that exploit ambiguities in the BERT model's training data, such as using new phishing tactics or encoding URLs in ways that mask their malicious intent. While BERT offers robust contextual understanding, it may still be fooled by highly sophisticated or novel phishing schemes that have not been sufficiently represented in the training dataset.
- URLs that dynamically generate content based on user interaction may pose a challenge. PhishGuard's real-time analysis is designed to assess URLs at the point of entry, but if the content or behavior of a site changes after this point,

a re-evaluation mechanism might be necessary to continuously ensure the safety of the site.

6.3. Areas for Improvement

- **Enhanced Machine Learning Models:** Continuing to evolve the training dataset to include the latest phishing tactics and exploring additional models that may complement BERT in handling edge cases or novel attacks could enhance effectiveness.
- **Hybrid Detection Techniques:** Integrating additional layers of security, such as heuristic checks alongside the BERT model, could improve detection rates, especially for new or complex phishing URLs that do not fit typical patterns.
- **User Behavior Analysis:** Incorporating user behavior analytics could add an extra layer of security by identifying anomalies in login patterns or credential usage that may indicate a phishing attack, thus providing another check against sophisticated phishing methods.
- **Performance Optimization:** Developing more efficient model architectures or employing model compression techniques could reduce the computational load, thereby enhancing scalability and reducing operational costs.

7. Summary and Conclusions

PhishGuard presents a groundbreaking step in enhancing password management with real-time phishing detection, a feature that is notably absent in many current password management systems. By leveraging a fine-tuned BERT model, PhishGuard significantly reduces the risk of users entering their credentials into phishing websites, thereby providing an added layer of security at the critical moment of credential input. This real-time detection and protection contribute to a more secure and user-friendly experience, as PhishGuard operates seamlessly in the background without disrupting the user's workflow.

The integration of phishing detection directly into password management workflows is an important contribution to the field of cybersecurity. Existing solutions often rely on outdated blacklists or post-factum security measures, leaving a vulnerability gap that PhishGuard addresses. The innovation lies in PhishGuard's proactive stance: it identifies phishing threats in real-time, alerts users before they can submit sensitive information, and offers a more comprehensive solution to phishing.

There are several areas in which PhishGuard could be further developed to enhance its capabilities and broaden its application scope:

Intelligence Level of the Algorithm:

- While PhishGuard's BERT model is already highly effective, future versions could include an adaptive learning feature, allowing the model to evolve based on new phishing tactics and URLs encountered in real-time. This could be achieved through periodic updates of the model based on newly identified phishing sites or by enabling the model to learn from user feedback directly.

- Another direction for improvement would be to combine the BERT model with other detection methods, such as heuristic analysis or deep learning techniques. This hybrid model could offer enhanced detection rates by cross-referencing multiple methods of phishing detection, particularly for more complex and obfuscated URLs that may evade simpler models.

Scalability and Performance Optimization:

- As PhishGuard moves toward large-scale deployments, optimizing the BERT model for speed and resource efficiency will be critical. Implementing techniques like model quantization, pruning, or even migrating parts of the model to the cloud could reduce the computational burden and ensure that PhishGuard remains responsive in environments with thousands of simultaneous users.
- To handle larger datasets and multiple users simultaneously, PhishGuard can benefit from parallel processing techniques. By distributing tasks across multiple servers or using cloud-based solutions, PhishGuard could maintain its efficiency even in high-demand environments.

Expanded Scope of Application:

- Currently, PhishGuard operates primarily within a password management context. Expanding its scope to support mobile platforms, browser integrations, and enterprise-level security solutions could significantly increase its user base. For instance, integrating PhishGuard with enterprise password management systems or cloud services could offer protection for businesses dealing with large volumes of sensitive data.
- While the current focus of PhishGuard is on URL analysis, future versions could incorporate phishing detection for other vectors such as email and SMS phishing. This would involve adapting the model to analyze email headers, subject lines, and message content, thus providing users with a more holistic protection framework against phishing across different communication channels.

New Feature Suggestions:

- **Real-Time User Feedback System:** To improve the model's effectiveness, a real-time feedback mechanism could be added, allowing users to flag potential phishing sites that were not detected by the BERT model. This information could then be used to retrain the model and improve its detection capabilities.
- **Phishing Attack Reports:** PhishGuard could generate detailed reports on detected phishing attempts, including the type of phishing attack, associated URLs, and suggested security actions for users. This feature would provide greater transparency and help users better understand and manage their cybersecurity risks.

PhishGuard's innovative approach has already made significant contributions to the field of password management and phishing detection. By continuing to improve its intelligence, expanding its applicability across platforms and phishing vectors, and addressing performance concerns in large-scale environments,

PhishGuard can become an even more powerful tool in the fight against phishing. The future development of PhishGuard will focus on making the system smarter, faster, and more versatile, ensuring that it stays ahead of emerging phishing threats and remains a reliable safeguard for user credentials.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Prakash, P., Kumar, M., Kompella, R.R. and Gupta, M. (2010) Phishnet: Predictive Blacklisting to Detect Phishing Attacks. 2010 *Proceedings IEEE INFOCOM*, San Diego, 14-19 March 2010, 1-5. https://doi.org/10.1109/infcom.2010.5462216

[2] Felegyhazi, M., Kreibich, C. and Paxson, V. (2010) On the Potential of Proactive Domain Blacklisting. *Proceedings of the 3rd USENIX Conference on Large-Scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More*, California, 27 April 2010, 6.

[3] Sheng, S., Wardman, B., Warner, G., Cranor, L.F., Hong, J. and Zhang, C. (2010) An Empirical Analysis of Phishing Blacklists. *Proceedings of the 6th Conference on Email and Anti-Spam*, California, 16-17 July 2009.

[4] Siganevich, S. (2024) Why 2FA Multi-Factor Authentication Is No Longer Sufficient to Stop Phishing. Seraphic Security. https://seraphicsecurity.com/resources/blog/2fa-multi-factor-authentication-is-not-sufficient-to-stop-phishing/

[5] Bhopen Singh, O. and Tahbildar, H. (2015) A Literature Survey on Anti-Phishing Browser Extensions. *International Journal of Computer Science & Engineering Survey*, **6**, 21-37. https://doi.org/10.5121/ijcses.2015.6402

[6] Jain, A.K. and Gupta, B.B. (2018) Towards Detection of Phishing Websites on Client-Side Using AI Based Approach. *Telecommunication Systems*, **68**, 687-700.

[7] Jain, A.K. and Gupta, B.B. (2019) An AI-Based Approach for Phishing Detection Using Hyperlinks Information. *Journal of Ambient Intelligence and Humanized Computing*, **10**, 2015-2028.

[8] Rao, R.S. and Pais, A.R. (2019) Two Level Filtering Mechanism to Detect Phishing Sites Using Lightweight Visual Similarity Approach. *Journal of Ambient Intelligence and Humanized Computing*, **9**, 3853-3872.

[9] Jain, A.K. and Gupta, B.B. (2017) Two-Level Authentication Approach to Protect from Phishing Attacks in Real Time. *Journal of Ambient Intelligence and Humanized Computing*, **9**, 1783-1796.

[10] Rao, R.S., Umarekar, A. and Pais, A.R. (2021) Application of Word Embedding and Machine Learning in Detecting Phishing Websites. *Telecommunication Systems*, **79**, 33-45. https://doi.org/10.1007/s11235-021-00850-6

[11] Guo, B., Zhang, Y., Xu, C., Shi, F., Li, Y. and Zhang, M. (2021) Hinphish: An Effective Phishing Detection Approach Based on Heterogeneous Information Networks. *Applied Sciences*, **11**, Article No. 9733. https://doi.org/10.3390/app11209733

[12] Sahingoz, O.K., Buber, E., Demir, O. and Diri, B. (2019) Machine Learning Based Phishing Detection from URLs. *Expert Systems with Applications*, **117**, 345-357. https://doi.org/10.1016/j.eswa.2018.09.029

[13] Rao, R.S., Vaishnavi, T. and Pais, A.R. (2019) Catchphish: Detection of Phishing

Websites by Inspecting URLs. *Journal of Ambient Intelligence and Humanized Computing*, **11**, 813-825. https://doi.org/10.1007/s12652-019-01311-4

[14] Aljofey, A., Jiang, Q., Qu, Q., Huang, M. and Niyigena, J. (2020) An Effective Phishing Detection Model Based on Character Level Convolutional Neural Network from URL. *Electronics*, **9**, Article No. 1514. https://doi.org/10.3390/electronics9091514

[15] Le, H., Pham, Q., Sahoo, D. and Hoi, S.C.H. (2018) URL Net: Learning a URL Representation with Deep Learning for Malicious URL Detection. https://dblp.org/rec/journals/corr/abs-1802-03162.html

[16] Chatterjee, M. and Namin, A. (2019) Detecting Phishing Websites through Deep Reinforcement Learning. 2019 *IEEE 43rd Annual Computer Software and Applications Conference*, Milwaukee, 15-19 July 2019, 227-232. https://doi.org/10.1109/compsac.2019.10211

[17] Xiao, X., Zhang, D., Hu, G., Jiang, Y. and Xia, S. (2020) CNN–MHSA: A Convolutional Neural Network and Multi-Head Self-Attention Combined Approach for Detecting Phishing Websites. *Neural Networks*, **125**, 303-312. https://doi.org/10.1016/j.neunet.2020.02.013

[18] Zheng, F., Yan, Q., Victor, C.M., Leung, F., Richard, Y. and Ming, Z. (2022) HDP-CNN: Highway Deep Pyramid Convolution Neural Network Combining Word-Level and Character-Level Representations for Phishing Website Detection. *Computers & Security*, **114**, Article 102584.

Appendix

Screenshots

Figure A1 shows the configuration of Password Manager, it creates a database called "pm" and two tables "secrets" and "entries". During the configuration the password manager asks to setup a Master Password for the password manager.

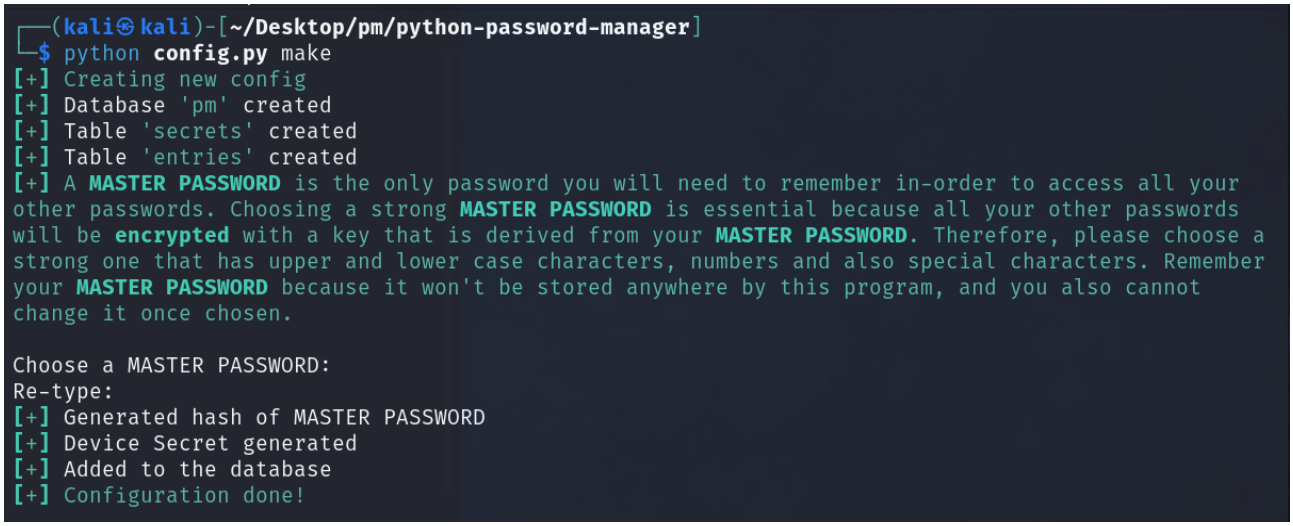

Figure A1. Configuration of password manager.

Figure A2 shows the help option of the password manager to see how to use it.

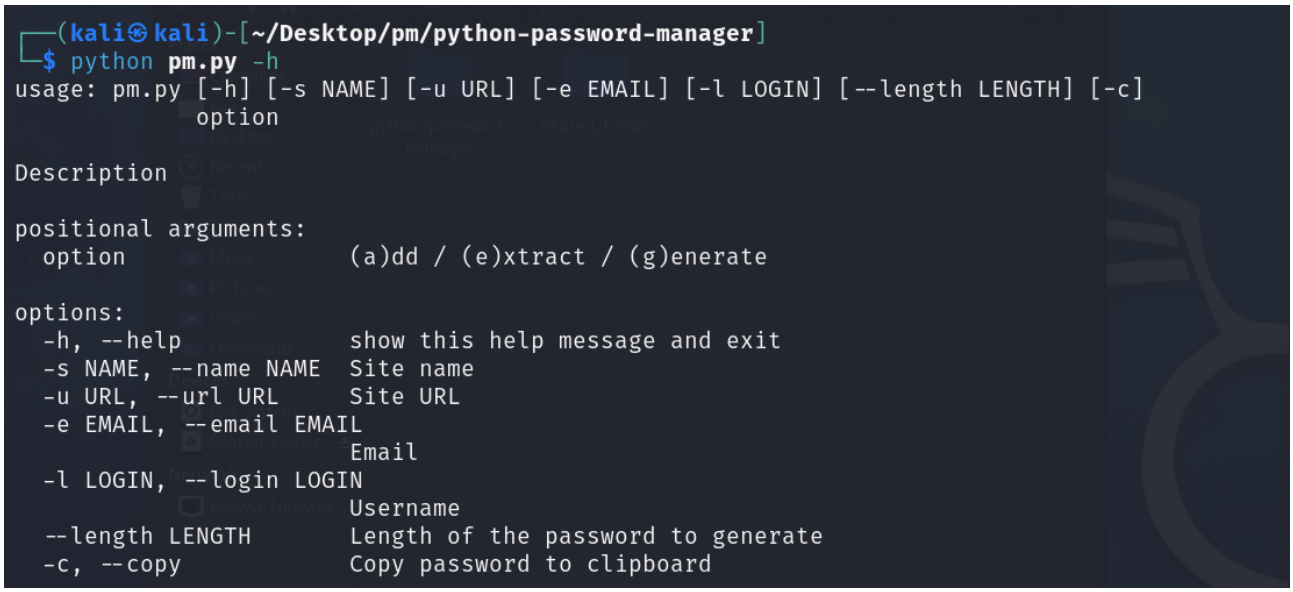

Figure A2. PM usage help.

Figure A3 shows that one can extract all the entries that are store in the database using this command.

```
┌──(.env)─(kali㉿kali)-[~/Desktop/pm/python-password-manager]
└─$ python pm.py extract
MASTER PASSWORD:
```

Results

Site Name	URL	Email	Username	Password
apple	http://www.apple.com	user1@apple.com	user1	{hidden}

Figure A3. Using extract option.

Figure A4 shows that one can successfully add a new entry to the password manager database as the URL used to store the entry was Legitimate.

```
┌──(.env)─(kali㉿kali)-[~/Desktop/pm/python-password-manager]
└─$ python pm.py add -s amazon -u http://www.amazon.com -e user2@gmail.com -l user2
MASTER PASSWORD:
Password:
[+] Added entry
```

Figure A4. Adding new entry with legitimate URL.

```
┌──(.env)─(kali㉿kali)-[~/Desktop/pm/python-password-manager]
└─$ python pm.py add -s instagram -u http://instagram-hosted-voting.blogspot.com -e sam@outlook.c
om -l sam
MASTER PASSWORD:
[!] WARNING: Phishing URL detected. Operation aborted.
```

Figure A5. Adding new entry with phishing URL.

```
┌──(.env)─(kali㉿kali)-[~/Desktop/pm/python-password-manager]
└─$ python pm.py extract -l user2
MASTER PASSWORD:
```

Results

Site Name	URL	Email	Username	Password
amazon	http://www.amazon.com	user2@gmail.com	user2	{hidden}

```
┌──(.env)─(kali㉿kali)-[~/Desktop/pm/python-password-manager]
└─$ python pm.py extract -s amazon
MASTER PASSWORD:
```

Results

Site Name	URL	Email	Username	Password
amazon	http://www.amazon.com	user2@gmail.com	user2	{hidden}

Figure A6. Retrieving entries using Username and Site name.

In Figure A5, we have deliberately used a phishing URL to check if the entry is being stored in the database or not. But the password manager throws a warning that the URL is Phishing, and it aborts the operation of adding the entry.

Figure A6 shows that I can also retrieve entries using Username and Site Name. This shows that the phishing detection model is not hindering the normal operation of the password manager.

Figure A7 shows the performance of BERT model when it is tested against 10000 URLs.

Figure A7. BERT performance.

Code Snippet

accuracy_find(json).py:

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch
from torch.nn.functional import softmax
import pandas as pd
from tqdm import tqdm

# Load the tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("ealvaradob/bert-finetuned-phishing")
model = AutoModelForSequenceClassification.from_pretrained("ealvaradob/bert-finetuned-
phishing")

def detect_phishing(text):
    # Encode the text using the tokenizer
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True,
max_length=512)

    # Predict using the model
    with torch.no_grad():
        logits = model(**inputs).logits

    # Convert logits to probabilities
    probabilities = torch.softmax(logits, dim=1).tolist()[0]

    # Returning the class with the highest probability
    return "phishing" if probabilities[1] > probabilities[0] else "legitimate"

# Load the dataset
```

```
df = pd.read_json('combined_reduced.json')
df = df.head(10000)  # Optionally limit the dataset for quicker testing

TP, FP, TN, FN = 0, 0, 0, 0

# Iterate over the dataset
for i, row in tqdm(df.iterrows(),bar_format='{l_bar}{bar:10}{r_bar}{bar:-10b}', to-
tal=len(df)):
    text, label = row['text'], row['label']
    status = "phishing" if label == 1 else "legitimate"
    result = detect_phishing(text)

    if result == "phishing" and status == "phishing":
        TP += 1
    elif result == "phishing" and status == "legitimate":
        FP += 1
    elif result == "legitimate" and status == "legitimate":
        TN += 1
    elif result == "legitimate" and status == "phishing":
        FN += 1

# Calculating the metrics
accuracy = (TP + TN) / (TP + FP + TN + FN)
precision = TP / (TP + FP) if (TP + FP) != 0 else 0
recall = TP / (TP + FN) if (TP + FN) != 0 else 0
fpr = FP / (FP + TN) if (FP + TN) != 0 else 0  # False Positive Rate

print(f'Accuracy: {accuracy * 100:.2f}%')
print(f'Precision: {precision * 100:.2f}%')
print(f'Recall: {recall * 100:.2f}%')
print(f'False Positive Rate: {fpr * 100:.2f}%')
```