ENHANCING PHISHING SITE DETECTION AND PREVENTION THROUGH MACHINE LEARNING WITH IMAGE CLASSIFICATION

Kaloyan Manev (1), Milen Petrov (1), Adelina Aleksieva-Petrova (2)*

(1) Sofia University "St. Kliment Ohridski", Sofia; (2) Technical University of Sofia, Bulgaria

* Corresponding Author, e-mail: kpmunev@uni-sofia.bg

Abstract: Distinguishing expertly designed phishing sites from genuine ones is difficult, likely contributing to the continued victimization of users by these attacks. The challenge of creating a universal algorithm to detect these malicious pages has resulted in the development of various tools to protect users. This paper introduces an innovative method for detecting and preventing phishing websites using machine learning, which is implemented in a newly developed browser plugin called NotPwned. Additionally, tools have been created for data collection and training computer vision models. A classifier has been designed to identify login forms based on submitted screenshots. The most well-known algorithms for object detection in images have been analyzed and compared. Furthermore, a model has been trained to recognize different logos. The research concludes with a comparison of the system's capabilities against those of competing solutions, demonstrating the effectiveness of the proposed method and the resulting product.

Key words: anti-phishing, computer vision, object detection, plug-in, phishing.

1. INTRODUCTION

One of the most serious problems in computer security is website phishing, where the victim is redirected using various techniques to a web page that is a fake copy of a website, most often a fake login or payment page. The malicious page may look exactly like the original, but the moment data is entered into it, they are sent to the attacker instead of being sent to an official server [1]. A new redirect or misleading message follows, and so the user does not even realise that they have become a victim of phishing, while at the same time, the attacker exploits the data obtained.

Several previous studies have been done in the field of phishing protection, and numerous products are already available [2]. Modern browsers have built-in security mechanisms, which most often check whether the URL of the visited website is part of a predefined list of malicious websites [1]. This approach has proven effective, but attackers can easily bypass it by using new websites with URLs that have not been blacklisted yet.

Some anti-phishing solutions are in the form of browser plug-ins, but most of them are no longer relevant, as the set of technologies that can be used for development has undergone significant improvement. There are some limitations, but browser plug-ins remain the only solution that is so easily accessible. They can monitor in real time all the websites visited and notify the user of any danger, even when there is a new phishing attack that has not been detected before.

The machine learning (ML) techniques significantly improve the effectiveness of phishing prevention systems. The researchers suggest that ML-based approaches provide a cost-effective, precise, and adaptive means of enhancing cybersecurity by learning from behavioral patterns and facilitating real-time threat detection and response [3].

This paper proposes a new anti-phishing method and implements and validates the plug-in prototype to protect users in this rapidly developing field. The main contributions of the paper are as follows:

- Proposing a flexible and proactive anti-phishing method to be used for detection and prevention based on ML.
- Implementing a prototype for protection against the theft of login credentials in the form of a browser plug-in that uses the proposed method.
- Validating the cases in which the proposed anti-phishing method would be effective and address specific problems related to the development of such a system.

The paper has the following structure. Section 2 contains an analysis of the scientific research conducted in the field of phishing protection. In Section 3, an anti-phishing method has been proposed. Section 4 outlines a prototype design. Section 5 presents the different approaches and tools for collecting the training examples used in the study. In Section 6, the results obtained from testing the prototype have been described and discussed. Section 7 contains the conclusion.

2. RELATED WORKS

Nowadays, phishing is one of the most common cyberattacks because to commit the crime, the attacker does not need a system to have any serious security flaws [4]. The life cycle of a phishing attack most often consists of five stages: reconnaissance, weaponisation, distribution, data exfiltration, and data exploitation [4]. Not all attacks are the same, and sometimes some of the stages are executed in parallel or multiple times.

Each different type of phishing attack creates the need for a new defence mechanism. Over time, many approaches have been developed to deal with the problems, but not all attacks can be stopped at the same stage. The two stages where the methods are most effective are distribution and data exfiltration. This is the reason why most studies in the field focus on these two stages.

The distribution phase of phishing attacks begins when malicious actors disseminate deceptive messages to potential victims, most commonly via email, phone, or social media. The most prevalent protective mechanism is the spam filter, which identifies and redirects suspicious messages, particularly email-based phishing, using content-based classification. Although widely adopted, spam filtering remains an evolving field due to its relevance in mitigating various message-based cyber threats.

The first spam filters were based on rule-based systems which use predefined rules to identify deviations from normal behaviour [5], for example checked the origin of messages and whether they contained certain words or phrases. Technological advancements have made it possible to create more efficient systems based on machine learning, which deliver better results and make fewer errors [6].

In the field of social networking sites, significantly less research has been done on whether spam filters are used for comparison. Recent studies again use methods from machine learning and more specifically from deep learning, such as the phishing detection method, which combines natural language processing functions with web feature extraction and Neumann similarity through a deep trust network [7]. To improve accuracy, oversampling is applied to increase the data, and phishing detection is ultimately achieved with a fuzzy deep neural-ordered autoencoder.

Phishing distributed via mobile devices and text messages is called smishing, and it is a growing problem. Research has been done on the subject, and like email phishing, there are two main approaches to combating it. The first one is the use of blacklists and a set of rules to check whether a message or call comes from a legitimate source [8]. Also, a mobile application with a rule-based SMS service designed to detect and prevent smishing attacks is introduced. Incoming messages are intercepted, analysed through a machine learning model via API, and classified as spam or ham, while the final decision remains with the user after notification [9].

Also, for real-time smishing (SMS phishing), a detection model is proposed that integrates Natural Language Processing and Machine Learning to effectively identify fraudulent messages in mobile financial environments, achieving high performance, thereby enhancing user security and trust [10].

The final stage of phishing attacks, where users can still protect themselves, is the data exfiltration stage. Data exfiltration starts when the victim visits a phishing page, and the best defence at this point is vigilance. At this stage, attackers wait for someone to fall into their trap and compromise their data. This means that whatever security mechanism is used, it must operate on the user's device. Examples of such security systems are anti-virus software and browser plug-ins that monitor activities and give a warning in case of danger.

Generally, the methods for protection employed at the data exfiltration stage are divided into two types: feature-based and content-based [11]. There is also an additional categorisation according to the kind of external resources used [12]. Hardware has become more and more potent over the years, and is now capable of performing the calculations that were previously performed on a remote server. Therefore, more and more solutions use only local resources. There are also hybrid systems, notwithstanding the classification of the approaches.

Feature-based methods use information about web pages such as URL address, domain, and ranking according to user ratings [11]. They often use external databases for reference. This category also includes whitelists and blacklists that restrict user access. Nevertheless, they are often used by administrators because they provide a high level of security. Machine learning and features are successfully used to create classifiers that effectively recognise phishing pages.

Computer vision is widely used in phishing protection. Systems that employ it perform well, but sometimes classify safe sites as malicious. They also rely on web page content, but instead of using code, they use a screenshot or extract all the images contained on a site. A common approach to detecting potential attacks is measuring the similarity between suspicious and legitimate web pages by comparing their visual content, including webpage layouts and logo-containing regions [13]. Other methods try to detect inconsistencies between the visual characteristics and the URL of the visited site. Brand names or logos are often checked, and therefore, heavy algorithms for image and text recognition are applied.

This necessitates the use of remote resources, and no matter how practical these approaches are, they remain unsuitable for some users.

A perfect technique for phishing detection does not exist. Each method has its strengths and weaknesses, so it is always necessary to perfect these approaches. The products available on the market are hybrid, combining techniques that can be effective in achieving good results.

3. PHISHING SITE DETECTION AND PREVENTION METHOD

The primary challenge with anti-phishing protection against new phishing attacks is not the lack of methods, but the complexity of existing solutions. The approach proposed in this paper addresses this issue by focused on four key attributes: accessibility (characterized by ease of installation and platform independence), confidentiality (ensuring that user data remains local and is not transmitted externally), protection against new attacks (capable of identifying previously unreported phishing attempts), and local execution (operating entirely on the user's device without reliance on remote servers). These principles are used as the base of the proposed anti-phishing system, which leverages computer vision techniques to replicate the intuitive, human-like perception users employ to recognize phishing websites, thereby enhancing detection effectiveness in real-world scenarios.

The workflow of the method is presented in the paper combines the advantages of different techniques and relies entirely on the appearance of the scanned page to achieve optimal results. It follows these steps (Figure 1):

1) The visited website is checked to see if it is on the whitelist which contains the top one million websites, and if so, the system considers the website as potentially legitimate.

2) A classifier is used to determine whether the current page contains a data entry form. If the visited address does not require the user to enter any data, then the system considers the site to be safe.

3) If the system detects a page resembling a login form, it uses an algorithm that looks for distinctive features of platforms selected by the user.

4) If features like matching logo are detected, but the site URL does not match the original address, the site is classified as dangerous, and the user receives a warning.

A crucial aspect of the second step is to assess whether the web page the user has accessed requires data entry. With this information, the system can avoid scanning a significant portion of the websites visited for phishing attempts. Implementing this optimization involves utilizing computer vision, as simply checking for forms in the source code can be evaded through various methods. For instance, malicious sites may omit password fields and instead use a plain text field with a specific font to mask the characters. Additionally, some phishing sites may rely entirely on JavaScript, lacking any forms or data entry fields at first glance.

Using a screenshot, the model needs to evaluate the likelihood that a web page contains a data entry form. This task is suitable for a classifier since it focuses on identifying characteristic features of login pages, such as having two text input boxes and a button, rather than searching for a specific object.

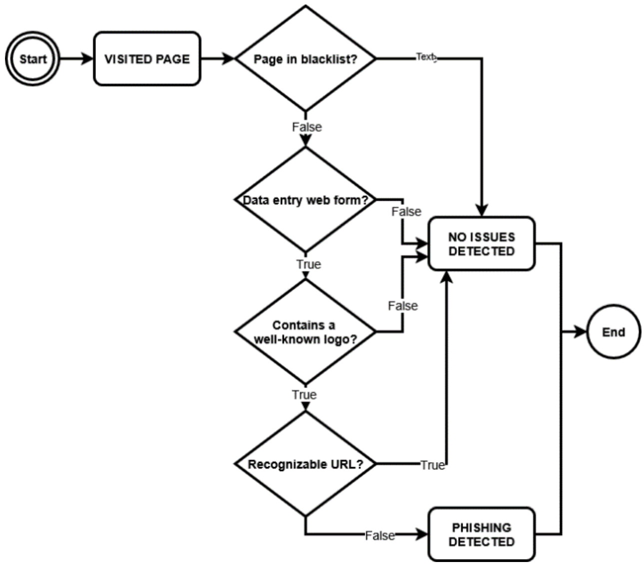

Figure 1. Workflow of the phishing site detection and prevention method

4. PROTOTYPE DESIGN AND IMPLEMENTATION

4.1. Prototype Design and Implementation Process

The prototype design and implementation process consists of several stages that will help to achieve the implementation and validation of the prototype (Figure 2).

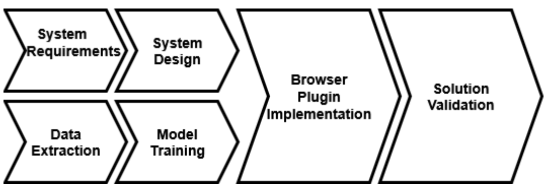

Figure 2. Stages in Prototype Design and Implementation

The stages are: collecting functional requirements for the new solution; designing the system based on the requirements; selecting algorithms and technologies for machine learning and tools for collecting and generating data for training the various models; training artificial intelligence (AI) using collected data, analysing its performance and evaluating training data on the influence of its performance results and their optimization; creating a browser plug-in that uses the trained models and testing the solution which checking whether it meets the requirements and analysing the cases where it performs better than existing solutions.

4.2. Architecture design and prototype implementation

The modern software system is composed of independent modules performing different tasks. This information is important to developers and is visualised using a "modular decomposition" diagram. Figure 3 shows the modular decomposition used in the development of the NotPwned plug-in.

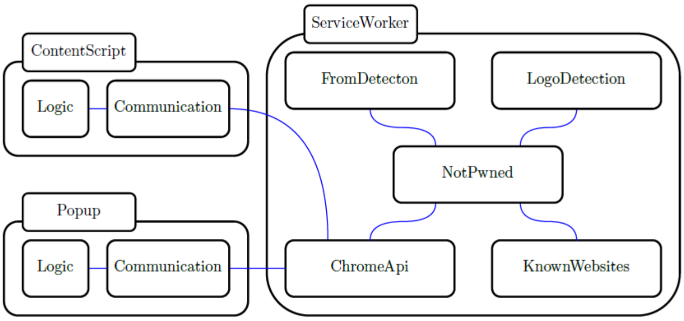

Figure 3. Modular decomposition of the NotPwned plug-in

During the planning of the NotPwned system, various development options were considered, and it was concluded that it would be most convenient for users if the system were to be distributed in the form of a browser plug-in. We selected a browser plugin architecture for two primary reasons that are critical to the method's efficacy and security: the necessity of analyzing rendered content and the preservation of user privacy through end-to-end encryption. This type of software is easy to distribute, easy to install, has almost no special requirements towards the user's system, and once installed, it can be synchronized with all of the user's devices.

The functional requirements define the system as a Google Chrome plug-in that activates its security mechanisms automatically when a user visits or reloads a web page (Figure 4). It classifies web pages as safe or dangerous based on a combination of heuristics and machine learning techniques.

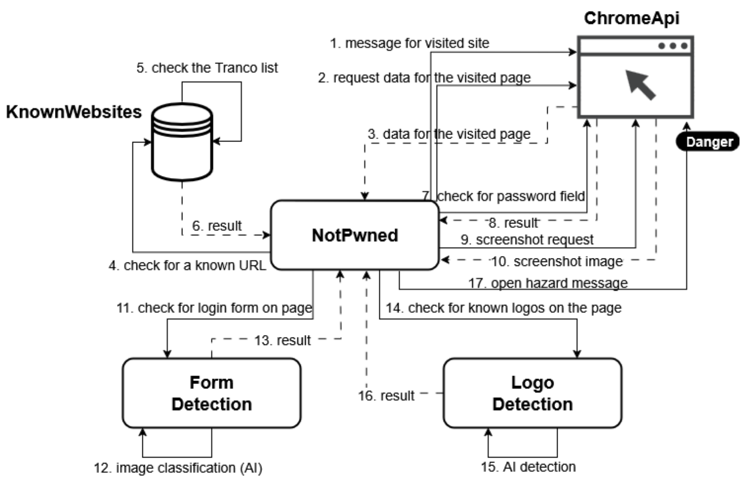

Figure 4. Implementation of the Proposed Anti-Phishing Method in the NotPwned Plugin

Initially, a domain is considered safe if it appears on the Tranco list [14] of the top one million websites or if it does not request login data. To enhance accuracy, the system integrates local machine learning models for detecting login forms and unauthorized logo use via computer vision, operating entirely within the user's browser to preserve privacy. It supports configurable security mechanisms with multiple activations per page and enables users to define their preferred protection sensitivity through three operational modes: light, balanced, and strict. Users can submit misclassification reports and override warnings to proceed to flagged pages, maintaining a balance between security and user autonomy. When a threat is detected, the system redirects the user to a secure page under its control. A configuration file allows users to specify trusted domains, and the system ensures real-time protection by combining static rules with dynamic content analysis.

5. TRAINING MODEL

This section presents the approaches used for collecting training examples as well as the training of the model used for the research. Some of the most famous neural networks were trained for the classification and detection of objects in images. A comparison was made between their results, and the two most suitable models were selected.

5.1. Recognition of login form

A computer vision method is used that allows the training of a classifier for detecting login pages.

Data extraction for login form recognition

Both positive and negative examples are needed to train a classifier. For the current task, the positive examples must be in the form of pictures of login pages, and the negative examples must be images of other web pages. 2,500 positive and 2,500 negative examples were collected to train the classifier. Extracting so many screenshots manually is a labour-intensive task, so several helper scripts were created to automate the process.

Selenium was used to extract screenshots because it allows the running of automated actions in popular browsers. The negative examples were extracted by visiting each site on a list of websites and taking a screenshot. The URLs of the first 3,500 pages on the Tranco list of popular sites were fed as input into the script. Some pages were inactive, duplicate, or inappropriate, and after a manual check, the number of negative examples was narrowed down to 2,500.

Collecting positive and negative examples looks almost like the same task. The problems are similar and are solved in the same way, but only if a list of login page URLs is also available, but no such list was found in the public domain. For this purpose, a script was created that was used to extract information from HTML. It accepts as input the Tranco list of popular sites, visits the home pages of all websites, and looks for links to login pages. The found URLs are saved in a file so that they can be visited and screenshot later.

This script was a heuristic-based tool designed for the offline task of dataset creation, where performance speed was not a critical factor. Such scripts are ill-suited for a real-time security tool because they are both slow and brittle; they rely on finding specific HTML tags and structures (e.g., <input type="password">) that attackers can easily hide or obfuscate using JavaScript and other techniques. In contrast, our Machine Learning model offers superior robustness by learning the generalized visual patterns and spatial layout of a login page from its rendered image. This approach is resilient to code-level obfuscation, as it evaluates the final appearance rather than the underlying source code, forcing an attacker to create a visually distinct page to evade detection, which would defeat the purpose of the phishing attack itself.

Websites that cause errors are again skipped. After all the links on the web page are extracted, they are validated and cleaned of parameters. Then it is checked if they redirect to a login form, and if so, they are added to a list of detected URLs. Finally, the list is saved as a text file. The final step is to visit the URLs from the file and take screenshots of the web pages.

Training the model for login form recognition

Microsoft's Custom Vision was used to create the login form detection model. It supports two types of algorithms – for classification and object detection. The user selects one of the predefined configurations, uploads their data, and can start training models. Once the training is complete, the models can be downloaded in several formats compatible with the various machine learning libraries. The NotPwned plug-in relies on models using TensorFlow.js, because they can be used in a browser environment.

5.2. Logo detection

The third step of the proposed approach determines whether a web page contains a known logo that should not be there. This is a complex task because logos come in different sizes, in different colours, and sometimes they are rotated or animated. There are various approaches to solving these problems, but there is no universal solution. No limits on input data have been defined, so the detection algorithm must be able to handle all logo variations, which necessitates the use of machine learning.

Data extraction for logo detection

The only data that the NotPwned plug-in needs is whether a known logo appears on a web page. The position and size of the detected image are irrelevant to the anti-phishing algorithm. Hence, logo detection is possible both using a classifier and an object detection model. For this paper, small and light models working with screenshots are needed. Thus, for each supported website, it is possible to create an individual model that detects its logos.

Using machine learning requires a large volume of training examples, irrespective of the approach. The difference between the data used to train classifiers and image detection models is insignificant. In both cases, a large number of screenshots are needed, both containing and not containing a given brand name. The only difference is that training image detection models requires the use of annotation files that show the exact position where detected objects are located in images. Most sites do not have thousands of pages that can be screenshot, so it is necessary to generate synthetic data to be used for training.

Artificial examples in the study were created by taking different logo variations and placing them on top of random images. The data used to train the classifier to recognise landing pages was used as background. The logo variations were automatically generated by a script that accepted images as input and generated 481 new ones per image. The dimensions of each logo variation were between 16x16 and 496x496 pixels, and it was rotated between 0 and 359 degrees.

Classifier training data can be automatically generated that accepts a list of logos and multiple background images, then places the logos at random locations on the backgrounds and thus generates synthetic screenshots.

Model training

Logo detection models are one of the key elements of the anti-phishing algorithm used in the NotPwned plug-in, which utilizes techniques of computer vision. Several popular models were trained and tested as both classifiers and object detection models in images. The results are presented in Table 1.

Table 1. Test results of computer vision models

Model	Type	Logo	Accuracy, %	Loss	Size, MB
YOLOV7-E6E	object detector	WhatsApp	99.5	0.01	576
Custom Vision	object detector	Facebook	72.9	-	11
Custom Vision	object detector	Instagram	75.5	-	42
Custom Vision	object detector	WhatsApp	71.5	-	42
Custom Vision	object detector	Messenger	90.1	-	42
Custom Vision	object detector	Snapchat	90.3	-	42
Custom Vision	object detector	Twitter	87.2	-	42
Custom Vision	image classification	Facebook	63.9	-	4.7
Custom Vision	image classification	WhatsApp	91.6	-	4.7
EfficientNet-B2	image classification	WhatsApp	85	0.31	88.8
EfficientNetV2	image classification	WhatsApp	96.4	0.34	232
MobileNetV3	image classification	Facebook	71	0.81	17.6
MobileNetV3	image classification	WhatsApp	91.3	0.30	17.6

The training data used (the number of negative examples is approximately the same as the number of positive examples) include:

- Facebook – 14 logos, approximately 13,500 positive examples
- Instagram – 10 logos, approximately 9,600 positive examples
- Messenger – 5 logos, approximately 4,800 positive examples
- Twitter – 8 logos, approximately 7,700 positive examples
- Snapchat – 4 logos, approximately 3,800 positive examples
- WhatsApp – 4 logos, approximately 3,800 positive examples

The most difficult task was detecting the Facebook logo, because the company uses various logos in different colours, or sometimes even just text. On the other hand, WhatsApp has few logo variations and is very popular.

At the time of the study, YOLOv7 was the most widely used architecture for neural networks that detect objects in images. In just 20 epochs, the model had almost 100% success rate in handling test data. Unfortunately, it cannot be used in the NotPwned plug-in due to its large size, because even with a model pre-trained on hundreds of brands, it would demand a lot of processing and retraining efforts.

Unlike YOLOv7, the models trained on the Custom Vision platform are smaller in size, but they produced less satisfactory results. The classifiers did significantly better with logos without text, and the accuracy of the detection models was always between 70 and 90 percent.

The reason for the lower accuracy on logo recognition is a "big picture, small frame" problem. Our system captures high-resolution screenshots (e.g., 1920×1080), but for our AI to run efficiently in a browser, it must use a much smaller input (e.g., 224×224). When we shrink the screenshot to fit this frame, fine details are inevitably lost. A small logo, perhaps just 20×20 pixels, gets compressed into an unrecognizable blur, losing the distinct features needed for identification. This loss of critical information during resizing is the direct cause of the inaccuracy. This approach is a deliberate trade-off between accuracy and user experience. A larger model like YOLOv7, for instance, would be more accurate but its massive file size and computational demand make it impractical for a browser extension. Similarly, analysing the screenshot in tiles would be unacceptably slow. Our implementation therefore prioritizes speed and responsiveness to ensure a seamless user experience.

The size of the input data plays a crucial role in the performance of the models, particularly for larger datasets, such as screenshots. Therefore, each image must be scaled down to the required size for the neural network type. Models other than Microsoft's were also tested with different input sizes. Popular classifiers were examined because, in most cases, they are lighter than object detection models. MobileNetV3 performed best, but it is not universal either, as it does not cope well with the Facebook logo. It achieved results similar to the Custom Vision models, but is larger and trains significantly more slowly.

6. RESULTS AND DISCUSSION

Scenario testing involves establishing a series of steps for utilizing a specific system. These sequences, known as scenarios, are then used to create test cases. Each scenario represents a particular way in which the system can be operated.

For the initial version of the NotPwned plug-in, several scenarios were developed, with the most critical one focusing on testing the anti-phishing features. This scenario included seven specific steps that needed to be followed during its execution: (1) opening a new tab in Chrome; (2) waiting for 15 seconds; (3) visiting a secure website; (4) visiting a dangerous website; (5) checking if the plug-in redirects the user to a warning page; (6) clicking on the "Back to safety" button and (7) checking if the plug-in redirects the user back to the secure website

Although the scenario seemed easy and quick to implement, step four proved to be a real challenge because all known malicious websites became inaccessible shortly after detection. To address this issue, 5 test web pages were created mimicking the Facebook platform. Using synthetic test data in this case is very useful because it guarantees that the pages will always look the same and that they will be accessible at all times. It is possible to create pages that are exact copies of websites, as well as fictitious ones.

An example of a phishing page that is not a copy of a website is the first Facebook test page. It is pretty different from the current one, and anti-phishing solutions that rely on visual similarity might likely be bypassed using a similar-looking phishing website. Furthermore, many people use the functionality that keeps them signed in to their accounts and may not know that the login page has been updated. This is the reason why NotPwned's performance was verified using such a page.

The following test examples were copies of the current Facebook login page. The first of them was an exact copy of the original, and the second used a replaced logo. Conventional anti-phishing solutions find it most challenging to detect entirely fictitious pages. To test the NotPwned plug-in, two such pages were created.

The final test phishing page used to verify the NotPwned plug-in was created by making a few minor modifications to Facebook's Forgotten Password page. Minor changes were made to the content, and one more text field was added. In this way, the page looked authentic because it met all visual requirements used on the platform.

For this study, eight widely used anti-phishing browser plug-ins were evaluated and compared against the initial version of the NotPwned plug-in: Avast Online Security & Privacy, Bitdefender TrafficLight, Malwarebytes Browser Guard, Microsoft Defender Browser Protection, Avira Browser Safety, ZoneAlarm Web Secure Free, AVG Online

Security, and ZecOps Anti-Phishing Extension. The assessment was conducted using a set of test scenarios involving artificially generated phishing pages designed to simulate zero-day attacks. NotPwned was the only plug-in to detect all five phishing attempts successfully. These results highlight a key advantage of NotPwned in addressing previously unknown threats, underscoring its potential effectiveness in real-time phishing protection where traditional solutions may fall short. In the experimental setup, it takes a little over 500 milliseconds to detect a completely unknown phishing page, and the size of the plug-in itself is 60.6 MB.

Several conclusions can be drawn from the conducted validation of this technology. Checking whether a web page contains a login form is a problem that is best solved using a classifier. Although a lot of data is needed, its collection can be automated. With enough training examples, it is possible to train a model to recognise different types of pages, such as registration or checkout pages. The presented classifier is reasonably sufficient for the creation of the first version of the NotPwned plug-in.

The difficulty in finding known logos is that the presentation of the models and the resources they require needs to be well-balanced. NotPwned was created using classifiers and object detection models created using the Custom Vision platform, because they perform well and require few resources, and they are also easy to train and use in a browser environment.

7. CONCLUSION

The paper describes an innovative method for detecting and preventing phishing websites using machine learning, which combines several aspects, namely computer security, machine learning, and user data privacy. These solutions are likely to gain popularity, as the advantages of combining these diverse aspects far outweigh the benefits of relying on conventional approaches in isolation.

The NotPwned plug-in is a new product that, even in its first version, manages to perform better than the competition in specific criteria. During the research, algorithms for one-shot and zero-shot learning for object recognition were taken into consideration. When they are ready for practical use, these approaches would allow NotPwned to detect phishing attacks without prior training of models.

ACKNOWLEDGEMENT

This study was funded by the project “Research and application of machine learning algorithms in the analysis and development of highly secure software”, fund with contract №: KP-06-N57/4 from 16.11.2021 by Bulgarian National Science.

REFERENCES

[1] Thapa, J., Chahal, G., Gabreanu, S. V., Otoum, Y. Phishing Detection in the Gen-AI Era: Quantized LLMs vs Classical Models. *arXiv preprint arXiv:2507.07406*, 2025.

[2] Abdolrazzagh-Nezhad, M., Langarib, N. Phishing Detection Techniques: A review. *Data Science: Journal of Computing and Applied Informatics*, vol.9, no.1, 2025, pp.32-46.

[3] Bonikela, H. R. Deep Learning for Cybersecurity: AI-Based Detection of Phishing and Fraudulent Web Pages. *International Research Journal of Modernization in Engineering Technology and Science*, vol.7, no.4, 2025, pp.2533-2559.

[4] Dou, Z., Khalil, I., Khreishah, A., Al-Fuqaha, A., Guizani, M. Systematization of Knowledge (SoK): A Systematic Review of Software-Based Web Phishing Detection. *IEEE Communications Surveys & Tutorials*, vol.19, no.4, 2017, pp.2797–2819, https://doi.org/10.1109/ COMST.2017.2752087.

[5] Birthriya, Santosh Kumar, Priyanka Ahlawat, Ankit Kumar Jain. Detection and prevention of spear phishing attacks: A comprehensive survey. *Computers & Security*, 2025, art.104317.

[6] Ji, F., Lee, K., Koo, H., You, W., Choo, E., Kim, H., Kim, D. Evaluating the effectiveness and robustness of visual similarity-based phishing detection models. In *34th USENIX Security Symposium (USENIX Security 25)*, 2025, pp. 3201-3220.

[7] Vidyasri, P., S. Suresh. FDN-SA: Fuzzy deep neural-stacked autoencoder-based phishing attack detection in social engineering. *Computers & Security*, vol.148, 2025, art.104188.

[8] Jain, Ankit Kumar, B. B. Gupta. Rule-based framework for detection of smishing messages in mobile environment. *Procedia Computer Science*, vol.125, 2018, pp.617-623.

[9] Akande, Oluwatobi Noah, et al. SMSPROTECT: An automatic smishing detection mobile application. *ICT Express*, vol.9, no.2, 2023, pp.168-176.

[10] Aaron Zimba, Katongo O. Phiri, Chimanga Kashale, Mwiza Norina Phiri. A machine learning and natural language processing-based smishing detection model for mobile money transactions. *International Journal on Information Technologies and Security*, vol.16, no.3, 2024, pp. 69-80. https://doi.org/10.59035/UEEN6450

[11] Zuraiq, AlMaha Abu, Mouhammd Alkasassbeh. Phishing detection approaches. *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*. IEEE, 2019.

[12] Maurya, Swati, Harpreet Singh Saini, Anurag Jain. Browser extension based hybrid anti-phishing framework using feature selection. *International Journal of Advanced Computer Science and Applications*, vol.10, no.11, 2019, paper 78.

[13] Singh, Tejveer, Manoj Kumar, Santosh Kumar. Walkthrough phishing detection techniques. *Computers and Electrical Engineering*, vol.118, 2024), art. 109374.

[14] Pochat, Victor Le, et al. Tranco: A research-oriented top sites ranking hardened against manipulation. *arXiv preprint arXiv:1806.01156*, 2018.

Information about the authors:

Kaloyan Manev – has MSc in Data Protection of Information in Computer Systems and Networks, at Faculty of mathematics and informatics, Sofia University, Bulgaria, and his main interests are in cybersecurity and machine learning, email: kpmunev@uni-sofia.bg

Milen Petrov – is a professor in Department of Software Technologies at the Faculty of Mathematics and Informatics in Sofia University with expertise in information security, serious games, and software learning platforms, email: milenp@fmi.uni-sofia.bg

Adelina Aleksieva-Petrova –is currently an associate professor at the Faculty of Industrial Technology in the Technical University of Sofia, and her research focuses on AI, semantic web, software design and validation, and agent-based technologies, aaleksieva@tu-sofia.bg

Manuscript received on 15 July 2025