ISSN: 1304-4796
Manisa Celal Bayar Üniversitesi Sosyal Bilimler Dergisi
Manisa Celal Bayar University Journal of Social Sciences
2026; 24 (2);615-634

Machine Learning-Based Malicious Web Page Detection

Mustafa Bilgehan İmamoğlu[1, 2]

ABSTRACT

With the growing spread of the internet, malicious activities like phishing attacks have become a major cybersecurity threat. To fill the gap in the current research on identifying such attacks, which aim to steal personal and financial data by redirecting users to fake online sites, this study sought to develop a Machine Learning (ML) model that can detect malicious websites with high accuracy using URL information. For this, Convolutional Neural Network (CNN) architecture was used that analyzes URL strings with high precision. Training and testing the model resulted in a validation accuracy of 89.95%. The developed CNN-based model has the potential to be integrated into cybersecurity systems to proactively defend users against phishing, as this experimental result clearly shows. Hence, this study makes an important contribution to improving digital security by demonstrating the effectiveness of CNN in detecting malicious websites based on URLs.

Keywords: *Malicious URL Detection, Phishing, Machine Learning, Cyber security.*

Makine Öğrenme Tabanlı Kötü Amaçlı Web Sayfası Algılama

ÖZET

İnternetin kullanımının yaygınlaşmasıyla birlikte, oltalama (phishing) saldırıları gibi kötücül faaliyetler siber güvenlik alanında önemli bir tehdit unsuru haline gelmiştir. Kullanıcıları sahte çevrimiçi platformlara yönlendirmek suretiyle kişisel ve finansal verilerin ele geçirilmesini amaçlayan bu tür saldırıların tespitine yönelik mevcut literatürdeki eksikliği gidermek amacıyla, bu çalışmada URL bilgisine dayalı olarak zararlı web sitelerini yüksek doğrulukla tespit edebilen bir Makine Öğrenmesi modelinin geliştirilmesi hedeflenmiştir. Bu kapsamda, URL karakter dizilerini yüksek hassasiyetle analiz etme yeteneğine sahip Evrişimli Sinir Ağı mimarisi kullanılmıştır. Gerçekleştirilen eğitim ve test süreçleri neticesinde, %89.95 oranında bir doğrulama başarımı elde edilmiştir. Söz konusu deneysel bulgu, geliştirilen CNN tabanlı modelin, kullanıcıları oltalama girişimlerine karşı proaktif bir şekilde savunmak adına siber güvenlik sistemlerine entegre edilme potansiyeline sahip olduğunu açıkça ortaya koymaktadır. Sonuç olarak bu çalışma, URL tabanlı zararlı web sitesi tespitinde CNN yönteminin etkinliğini doğrulayarak, dijital güvenlik standartlarının yükseltilmesine yönelik literatüre önemli bir katkı sunmaktadır.

Anahtar Kelimeler: *Kötü Amaçlı URL Tespiti, Kimlik Avı, Makine Öğrenmesi, Siber Güvenlik.*

[1] Contact: bilgehan@ktu.edu.tr
[2] Asst. Prof. Dr., Karadeniz Teknik Üniversitesi, Trabzon/TÜRKİYE, ORCID: 0000-0002-3496-2959.

(Submission: 03.07.2025 / Published:30.06.2026)
Doi Number: 10.18026/cbayarsos.1733586
Type of Research: Research Article

1. INTRODUCTION

A comprehensive interlinked computer system exists online, enabling individuals to find shared information and maintain their relationships. It has developed into a vital tool for communication that enhances fast resource access and encourages direct user interactions through connections with one another. From early times, the Internet delivers several advantages like communication, sharing news, e-commerce, bill processing, and distributing information. However, the growth of Internet use has brought many difficulties that can cause problems for users. Many elements can harm this connectivity. The growth in technology has converted the Internet into an arena at risk for various illegal behaviors ranging from online fraud to cyber attacks. Internet innovations have caused major safety concerns since attackers apply cyber-activities. Phishing schemes and trojans often use Internet addresses or URLs to carry out their destructive actions (Torska 2023).

Phishing attacks aim to steal sensitive and private data by tricking Internet users into revealing personal information, such as passwords and credit card numbers. The attackers typically aim to gain access to financial accounts and profit from these accounts. As technology advances, users need to become more aware of potential threats. Attackers using URLs often deploy malicious software, which are programs designed to access a machine and degrade its performance. Users often cannot distinguish whether the websites they visit are legitimate or fake. By logging into these sites, they inadvertently activate malicious programs without realizing it.

Users typically become suspicious of malicious software only when they notice a decline in system performance. By this time, attackers have usually succeeded in executing their plans (Hong, 2012). Therefore, it has become very important to identify the URLs of malicious websites before they reach the end user. Machine learning methods, which classify URLs based on information and behavior, are widely used to detect malicious websites. The classification processes of URLs have improved overall performance by automating these tasks without human intervention. At the forefront of these threats are attacks that create URLs for phishing purposes. Number of malicious websites poses a growing threat to users. Attacks conducted by cybercriminals, such as phishing and other cyber attacks, lead to dangerous situations for Internet users, including financial fraud and the theft of personal data. Users must become more aware and vigilant about cyber threats posed by malicious websites (Chaudhry et. al, 2016).

1.1. The Spread of The Internet and Increasing Cyber Threats

The widespread adoption of the internet has significantly facilitated digital communication and information sharing, while simultaneously leading to a rapid increase in cyber threats. These threats encompass various dangers that severely jeopardize the security and privacy of individuals and organizations. Cyber threats such as malware (viruses, trojans, and ransomware), phishing attacks, distributed denial-of-service (DDoS) attacks, identity theft, and social engineering can result in financial losses, theft of personal and sensitive information, privacy breaches, and psychological trauma. These attacks can also damage

the reputation of individuals and organizations, leading to job and career losses. Furthermore, the rise in cyber threats undermines users' trust in digital platforms, causing them to act cautiously and suspiciously in online transactions and interactions.

Preventative measures against cyber threats include using strong and unique passwords, implementing two-factor authentication (2FA) systems, regularly updating software and hardware, effectively utilizing security software, increasing education and awareness about cybersecurity, and regularly backing up data. These measures are critical in ensuring that internet users can navigate the digital environment safely and consciously. Informed users and robust security measures will minimize the adverse effects of cyber threats and contribute to creating a more secure digital world (Abomhara & Køien, 2015).

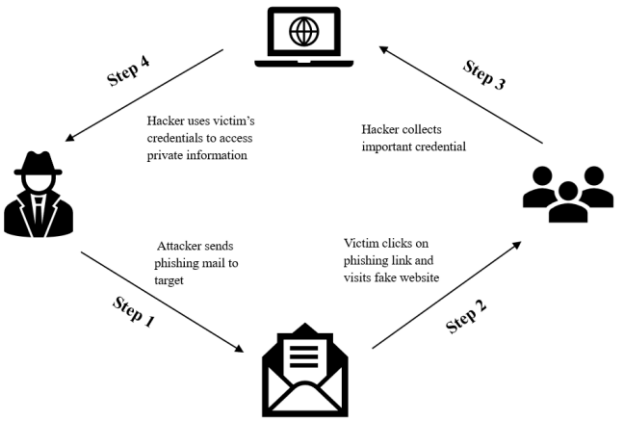
1.2. Cyber Threats and Phishing Attacks

As seen in Figure 1, Phishing attacks aim to steal sensitive and private data by tricking Internet users into revealing personal information such as passwords and credit card numbers. The attackers typically aim to gain access to financial accounts and profit from these accounts. As technology advances, users need to become more aware of potential threats. Attackers using URLs often deploy malicious software, which are programs designed to access a machine and degrade its performance.

Users often cannot distinguish whether the websites they visit are legitimate or fake. By logging into these sites, they inadvertently activate malicious programs without realizing it. Users typically become suspicious of malicious software only when they notice a decline in system performance (Verma & Shri, 2022). By this time, attackers have usually succeeded in executing their plans. Identifying URLs has become crucial. Machine learning methods, which classify URLs based on information and behavior, are widely used to detect malicious websites. The classification processes of URLs have improved overall performance by automating these tasks without human intervention.

Various methods can be used to protect against malicious websites. Firstly, using up-to-date antivirus software is crucial for detecting and blocking harmful software. Regularly updating web browsers and operating systems helps close known security vulnerabilities, reducing risks. Using strong and unique passwords makes accounts more resistant to brute force attacks. Being cautious of phishing attacks and avoiding suspicious email attachments and links helps protect users from such traps. Employing two-factor authentication (2FA) provides an additional layer of security for accounts. However, these measures are user-based precautions and do not address the detection or elimination of malicious websites themselves (Kumaraguru et al., 2007).

Figure 1. Phishing attack steps



Source: Oest et. al. (2018)

One of the methods to protect against malicious websites is to determine whether a website is fake or not. Various techniques can be used to determine whether a website is fake. DNS queries can analyze the site's reliability by checking the domain's registration details and IP addresses. The site owner's identity and contact information can be verified through various databases; fake sites often have hidden or incomplete information. The script structure also provides important clues; fake sites typically contain malicious code or use counterfeit versions of popular content management systems. The URL structure is another effective way to identify fake sites; legitimate sites usually have well-known domain names, whereas fake sites use complex or spoofed URLs. When these methods are used together, the authenticity of a website can be determined more effectively (Korkmaz & Büyükgöze, 2019).

1.3. Types of Cyber Threats

The Anti-Phishing Working Group (APWG) is a group that keeps an eye on phishing attacks happening all over the world and works to spread the word about this problem. They track things like how many unique phishing sites are out there, the different subject lines used in phishing emails, and how many brands get targeted by these attacks.

When they talk about unique phishing sites, they're looking at how widespread phishing really is by checking out the reported phishing URLs. The unique email subjects show just how varied these attacks can be by counting all the different subject lines scammers use to trick people. And when it comes to the number of brands targeted, they're measuring just

how many different companies are getting hit by these phishing incidents. All of these metrics help paint a clearer picture of how common and diverse phishing attacks are.

According to the APWG (2023) report, nearly five million phishing attacks were observed in 2023, with a total of 4,987,809 attacks recorded (see Table 1). This made 2023 the worst year on record for phishing, surpassing the 4.7 million attacks seen in 2022. In the fourth quarter of 2023, APWG detected 1,077,501 phishing attacks. This number was slightly higher than the 999,956 attacks in the third quarter but lower than the 1,286,208 attacks in the second quarter and significantly below the 1,624,144 attacks in the first quarter. The first quarter of 2023 marked the highest level of phishing attacks in APWG's historical observations.

Table 1. Phishing Activity Trends Report

	Number of unique phishing Web sites (attacks) detected	**Unique phishing email campaigns**	**Number of brands targeted by phishing campaigns**
Jan.	495,690	40,863	561
Feb.	509,394	45,259	549
Mar.	619,060	40,742	576
Apr.	597,789	41,083	544
May	381,572	30,717	521
Jun.	306,847	22,610	498
Jul.	327,294	29,110	477
Aug.	331,962	33,162	499
Sept.	340,700	32,740	508
Oct.	356,538	22,750	477
Nov.	350,776	24,621	442
Dec.	370,187	20,642	420

Source: APWG (2023).

Phishing attacks decreased in the second quarter of 2023 due to the closure of Freenom's free domain program. Freenom is a program that has long been widely used by phishers.

However, for years it has offered free domain name registrations on five country code top-level domains: .TK, .ML, .GA, .CF and .GQ. According to historical data, Freenom domains were used in 14% of all phishing attacks worldwide. It was responsible for 60% of phishing domains reported across all country code top-level domains in November 2022. In January 2023, Freenom stopped offering free registration. This ended phishers' free domain inventories as of mid-2023, and phishing in these domains ended. In February 2024, Freenom announced that it was exiting the domain business entirely. Settled a lawsuit filed by Meta, which alleged that Freenom monetized traffic to phishing websites while ignoring abuse complaints about those domains.

APWG observed 963,994 phishing attacks in the first quarter of 2024; This is the lowest quarterly total since the fourth quarter of 2021 and well below the record high of 1,624,144 attacks observed in the first quarter of 2023. From June 2023 to March 2024, the monthly number of Attacks remained stable. Although the number of reports received decreased, the number of unique email campaigns increased by 64% compared to the fourth quarter of 2024; This suggests that phishers vary their email subject lines to bypass email filters. Additionally, the number of brand names reported to APWG has decreased recently.

1.4. Phishing Attacks

Phishing attacks are a type of cyber-attack typically aimed at stealing sensitive information such as personal identification details, account passwords, and credit card numbers. These attacks are carried out by impersonating a trustworthy entity or person. Today, phishing attacks can manifest in various forms, with the most common methods being email, SMS, phone calls, and social media channels. Originating in the 1990s and relatively simple compared to current cyber threats, these attacks saw a rapid increase in the 2000s due to the rise of e-commerce and digital banking activities (Chanti & Chithralekha, 2022). Indeed, with the advancement of technology, these attacks continue to grow and particularly victimize individuals who lack digital awareness. Given the sheer volume of reported incidents, as documented by organizations like the Anti-Phishing Working Group (APWG), phishing remains one of the most significant and persistent threats to customer security in the digital landscape.

Phishing attacks are often constructed around a narrative. These narratives are crafted to appeal to the interests of the target audience, with attackers casting a bait through emails, messages, and other communication channels. Attackers mimic legitimate channels, such as emails and websites, to communicate with individuals. The content of the emails, messages, etc., typically includes actions like clicking a link, opening an attachment, or providing personal information. These contents are prepared using social engineering techniques and are designed to be difficult to distinguish from genuine ones. When the victim is deceived and clicks on the link in the message, they are directed to a fake website created by the attackers, where malicious code might be executed on their devices, malicious files may be downloaded, or they might be prompted to enter their personal information. This process serves the attacker's purpose and leads to the theft of sensitive

information, causing both financial and emotional harm to the victim (Abdillah et al., 2022).

Phishing attacks, a serious cybersecurity threat, are continually evolving with new scenarios rather than sticking to traditional methods. Consequently, these attacks continue to harm individuals and organizations. The increasing frequency of phishing attacks indicates that suitable environments for these attacks still exist. Experts have been working for years to develop various methods to counteract phishing attacks.

Historically, many security experts have focused on removing the end-user from the equation and making the system flawless and secure. However, it is not feasible to completely eliminate the end-user from the system (Arachchilage & Love, 2014). In an empirical study conducted by Dodge et al. (2007), students' awareness levels regarding phishing attacks were measured, revealing that students disclosed information that should not be shared with unauthorized users and fell victim to phishing attacks by opening attachments. Vishwanath et. al. (2011) demonstrated that an increase in the number of phishing emails received by individuals also increased their risk of being deceived. Another finding of the study was that users were more likely to respond to phishing emails from familiar individuals or organizations rather than from unknown sources. The same study also found no positive correlation between an individual's technological proficiency or previous experience and their susceptibility to phishing attacks. In this context, various studies have suggested that user security training designed for end-users could be effective (Kumaraguru et al., 2007).

1.5. Machine Learning for Detecting Malicious Websites

On the sustained growth of malicious websites; the internet is used more and more every day, so malicious websites wanting to deceive users are also developing. It is important to identify these websites as a means to raise awareness of people who constitute the weakest link in cyber security, to protect them from the attacks. In areas where there is the use of artificial intelligence, machine learning is taking its course to the use of malicious websites detection. Underlying this type of approach are complex algorithms that correlate a large volume of previously gathered big data on web traffic to uncover malicious activities and predicted concerns. The use of such information for learning the patterns of types of normal and malicious websites and detecting whether or not a new website with unknown content, structure and links is a malicious one is the task of such machine learning models (Ahsenali et al., 2016). The labels on the data they are being trained on are helpful to the extent that especially supervised learning algorithms understand malware distribution websites and other cyber threat websites, especially because they also can adjust labels. In some cases, such as with anomaly detection, unsupervised learning techniques can identify threats with new behaviours different from known threats. With these machine learning algorithms, cyber security specialists will be able to build proactive and powerful resource for the resistant against malicious websites (Bhavsar et al., 2018).

In their study, Anıl Utku and Ümit Can (2022) worked on creating methods to spot dangerous websites by looking at URL attributes. They tested a bunch of popular machine learning models, including Decision Trees, k-Nearest Neighbours, Light Gradient Boosting, Logistic Regression, Multilayer Perceptron, Random Forest, Support Vector Machine, and Extreme Gradient Boosting. Their efforts paid off as they effectively used these techniques to accurately detect threats based on the features of the URLs.

On another note, Siddharth Singhal and Utkarsh Chawla (2020) found that the Gradient Boosting Algorithm, set to a maximum depth of 4, nailed it with an accuracy of 96.4% when classifying malicious and benign URLs. They came up with a solid method to help users identify harmful websites and URLs. The dataset they used for training and testing came from various sources, and again, the algorithm hit that impressive 96.4% accuracy on the test dataset. This research showed how machine learning can be a game-changer in classifying harmful versus safe websites, leading to some great results.

Jitendra Kumar et al. (2020) utilized randomization of the dataset, feature engineering, linguistic analysis, and host-based features for feature extraction and statistical analysis. They compared different classifiers for the phishing URL classification task and achieved the highest accuracy of 98% with a Naive Bayes Classifier, demonstrating precision=1, recall=0.95, and F1-Score=0.97.

Manjeri et al. (2019) conducted a study using various machine learning algorithms such as Support Vector Machine (SVM), k-Nearest Neighbors (KNN), Random Forest, Decision Tree, and Naive Bayes for URL classification. They adopted a dataset and model usage approach, supported by different machine learning algorithms, achieving high accuracy results. Additionally, they employed association rule mining algorithms such as Decision Tree Rule, FP Growth, and Apriori to predict malicious URLs.

The identification of URLs is crucial as they can potentially harm systems or secretly store passwords without user knowledge (Manjeri et al., 2019). Traditional methods are insufficient against modern attack types; instead, advanced solutions using machine learning methods are effective. One such solution is the CNN algorithm. Machine learning is widely used in various products, including online searches, social networks, and website recommendations on consumer smartphones (LeCun et al., 2015). CNN algorithms are applied in diverse fields such as image and audio processing, natural language processing (NLP) and biomedicine (Şeker et al., 2017). A model called ConFully was developed based on the CNN algorithm to solve this problem through classification.

1.6. Detection of Malicious Websites with Convolutional Neural Networks (CNN)

CNN algorithm depends on a function called “Contained” and is a type that uses multi-layer perceptrons. Some basic design principles of CNNs are drawn from neuroscience. It was also developed by examining the functioning of the visual center in the animal brain. In the visual center, the image is first divided into subregions. Simple cells identify local features in the image, such as edge features of objects, while complicated cells reply to

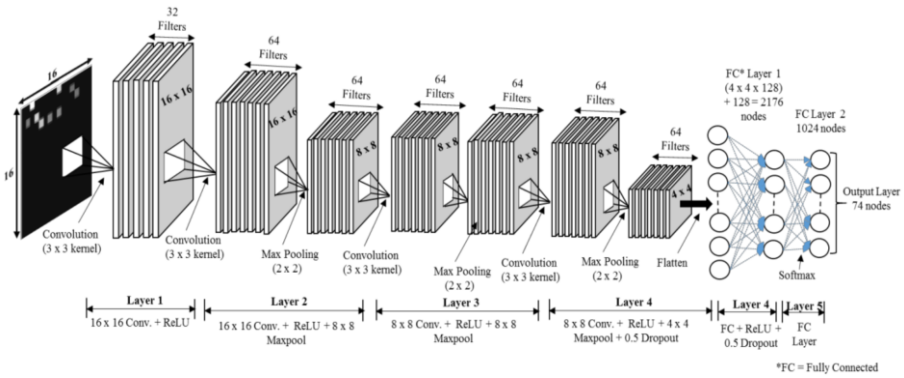
features like those detected by elementary cells. However, they remain invariant to small changes in feature location and focus on the entire visual field (LeCun et al., 2015; Goodfellow et al., 2016).

The CNN architecture consists of multiple layers and the stages consist of two layers, the first one is the convolutional and pooling layers. The units in the convolutional layer are organized by feature maps connected to the local feature maps from the previous layer and done through a filter bank structure of each unit (Özmen, 2022).

1.6.1. Convolutional Neural Networks (CNN)

CNN is a deep learning algorithm that uses images as input data. By leveraging the features of these images, CNNs can distinguish between different images, effectively performing classification tasks. The structure and working principle of CNNs are illustrated in Figure 2. Compared to other classification algorithms, the preprocessing required for data used in CNNs is significantly lower. This is because, unlike other classification algorithms, CNNs have the ability to learn filters (Kim, 2017).

Figure 2. Layers forming the CNN structure



Source: Sinha & Hwang (2019).

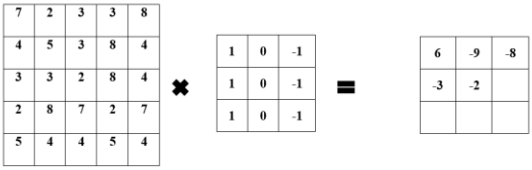
The architecture of CNNs consists of three layers: convolutional layer, pooling layer, and fully connected layer. An image input passes through these layers and goes through various processes until it becomes suitable for the deep learning model (Sayracı, 2021). For the specific task of malicious URL detection, CNNs are highly important because they can automatically learn hierarchical features from the sequential character data of a URL string. Unlike traditional machine learning models that rely on hand-crafted features, CNNs treat the URL as a one-dimensional "image" and use convolutional filters to identify local patterns, such as suspicious keywords, domain structure anomalies, or character-level obfuscation techniques, making them robust against evolving attack methods.

1.6.2. Convolutional Layers

The Convolutional Layer is the first layer in CNN algorithms that operate directly on images. It can be considered the fundamental building block of CNNs. Images consist of matrices made up of pixels carrying specific values. In this layer, the process involves moving smaller filter matrices across the image matrices to extract specific features from the images. These features can be low-level or high-level (Sayracı, 2021).

For example, a filter relevant to the topic in Figure 3 could detect edges on the image. Let's assume a 5x5 matrix for the image and a 3x3 matrix for the filter to be applied. When applying the filter to the image, it is initially placed over the top-left corner of the image. The indices between the two matrices are multiplied and all results are summed, with the outcome then being written as the first element of the output matrix. Subsequently, the filter matrix is shifted one pixel to the right, and the process is repeated. After completing operations in the first row, it moves to the second row and repeats the process. Once all operations are done, the output matrix is formed. This process enables the detection of features in the image.

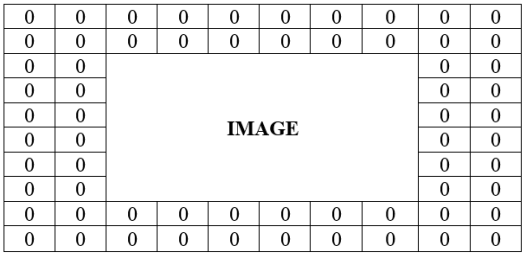
Figure 3. A 5x5 matrix



In CNN algorithms, the parameters learned by the model are the values within the filters. The model continuously updates these values and strives to better detect features. Moreover, filters can be applied to images for operations like sharpening, adding noise, and edge detection. Since images consist of multiple features, it is generally necessary to apply multiple filters in CNNs. This leads to the conclusion that CNN algorithms typically contain multiple convolutional layers. Two commonly used terms in filter operations are Stride and Padding, which can be summarized as follows (Kim, 2017):

Stride is a parameter in CNN models that can be adjusted. This parameter determines how many pixels the filter moves across the main image. Referring back to our previous example, we observed a stride value of one, meaning the filter shifts one pixel at a time across the image. If the Stride value were larger, the resulting feature map would be smaller due to the filter moving according to the Stride value on the image. If a filter is applied to an image, the output will be smaller than the original image due to the dimensions. To prevent this, we can use a method called Padding. Padding involves adding zeros around the image as if creating a frame. Depending on the filter size, these zero-padding layers can be increased. The padding process is illustrated in Figure 4.

Figure 4. Filling process in 5x5 matrix

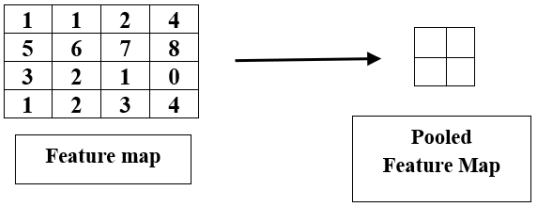


1.6.3. Pooling Layers

In the architecture of Convolutional Neural Networks (CNNs), alongside multiple successive convolutional layers, there exists a pooling layer. The purpose of the pooling layer is to reduce the size of the image by decreasing the number of parameters when images are very large. This process both preserves important information and reduces the spatial dimensions of feature maps. There are two types of pooling methods: maximum pooling and average pooling (Goodfellow et al., 2016). Generally, maximum pooling is preferred in this layer.

Maximum pooling is used to reduce the spatial dimensions of the input data after the convolutional layer. It sits between two convolutional layers. In maximum pooling, the height and width are reduced by taking the maximum value without changing the rotational or positional aspects, thereby extracting dominant features and effectively continuing the model's training process. Figure 5 illustrates the visualization of the pooling layer on a matrix.

Figure 5. View of the Pooling Layer on the matrix



1.6.4. Fully Connected Layers

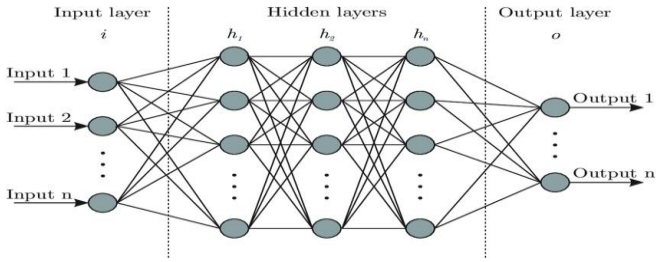
The fully connected layers in CNNs are the last part of the CNN setup. Think of these layers like a big web where every node connects to all the nodes in the layer before it. They're super popular in areas like computer vision and deep learning because they do a great job at spotting and classifying tricky visual patterns.

In these layers, the feature maps from earlier convolutional and pooling layers get combined after a flattening process. Flattening takes the data from a matrix format and squishes it into a flat vector, which makes it easier for traditional neural networks to work with during the final learning steps (Kim, 2017; Atıcı & Kocer, 2023). This setup allows CNNs to handle larger datasets using the features they've picked up along the way. But there are some drawbacks to fully connected layers. They can easily overfit, which means they might do great on the training data but struggle when faced with new data. Plus, they require a lot of computational power, making it tricky to generalize the network. So, when using these layers in CNNs, it's crucial to plan things out carefully, especially based on how complex the model is and how big the dataset is. These layers play a big role in advanced image processing and deep learning tasks. They help figure out how features are extracted and classified, ensuring that CNN models run smoothly and accurately when everything is set up correctly (Dua, M. et al., 2019).

To picture this better, think about how these fully connected layers operate. They take those feature maps from earlier layers and flatten them into one long vector. These feature maps act like matrices that highlight different parts of an image. Once flattened, this vector moves into fully connected layers. Each neuron in these layers connects to every feature in that input vector, doing a weighted sum followed by an activation function. The last fully connected layer usually serves as the output layer where you get results like classifications or predictions.

Fully connected layers (Figure.6) allow for deep analysis of learned features and help the model grasp complex relationships in the data. But because they can overfit and require a lot of computing power, it's smart to use regularization techniques to keep the model flexible and ready for new data (Atıcı & Kocer, 2023).

Figure 6. Fully Connected Layer working principle



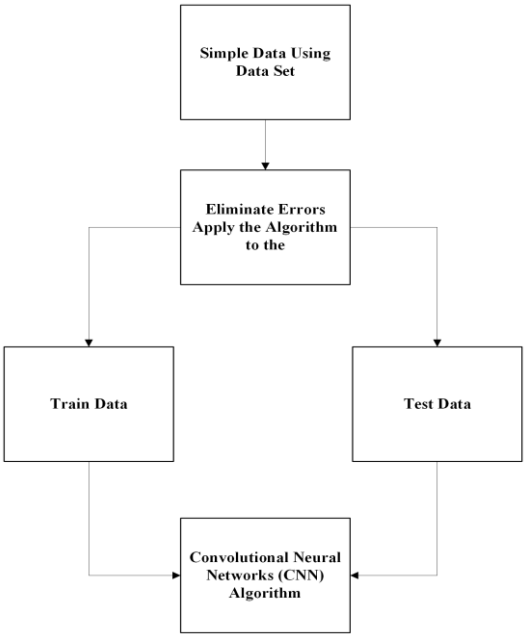
Source: Dua, M. et al. (2019).

2. EXPERIMENTS

The flowchart of this study, which facilitates the identification of malicious websites through URL information using a CNN algorithm, is presented in Figure 7. In the first step, a dataset containing URL addresses was prepared. This dataset includes both the URL and category information indicating whether the website associated with the URL is malicious

or not. After preprocessing the URL information in the dataset to remove unnecessary expressions, the data is divided into two parts: training and testing datasets based on predetermined ratios. The accuracy of the CNN model, trained using the training data, is then assessed by evaluating its performance on the test data, thereby calculating the algorithm's accuracy rate.

Figure 7. Training and testing the dataset using a CNN algorithm



2.1. Dataset

The study is all about figuring out if certain web addresses lead to malicious websites, using machine learning to do it. They got their data from Kaggle (Kumar, 2019), a pretty popular place for datasets in studies like this. The dataset includes 450,176 rows, with two main pieces of information: the URL of the site and whether it's flagged as malicious or not. Out of all these URLs, 345,738 are safe, and 104,438 are marked as malicious. So, if you break that down, about 77% of the web addresses are fine, and 23% are the ones to watch out for.

2.2. Data Pre-Processing

Since the dataset consists of URL information for websites to be determined as malicious or not, it needs to be preprocessed for the machine learning algorithm and prepared in a format suitable for the algorithm to process. CNN is a deep learning model generally used in the field of image processing and is particularly successful in visual data analysis and

classification. Since the data that CNN can process is usually in the form of 2D matrices (images), it is necessary to convert the URL information into numerical values and provide it to the CNN in an appropriate format. This transformation can be done by representing the URL characters with numerical values and feeding these numerical values to the CNN as a sequence or matrix. In this way, the CNN algorithm can learn the characteristic features of the URLs and classify whether they are malicious or not.

In the study, the conversion of URL information into numerical values is carried out by considering whether the characters forming the URL are printable. The printable element contained in the string module in Python includes all printable (viewable) ASCII characters as a subset of the ASCII character set. When converting the URL to numerical data, it is checked whether each character it contains is in the printable array. Characters that are in the printable array obtain their numerical counterparts by adding 1 to their positions in the array. The characters in the printable array are shown in Figure 8.

Figure 8. Printable array elements.

```
0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&'()*+,-
./:;<=>?@[\]^_`{|}~ \t\n\r\x0b\x0c
```

Assuming the example URL in the dataset is http://www.ktu.edu.tr, the conversion of the URL into numerical data using the printable array will be carried out in the following steps:

The index values of the characters forming the URL in the printable array are provided in Table 2.

Table 2. Example of Converting URL to Numeric Data

Character	*h*	*t*	*p*	*:*	*/*	*w*	*.*	*k*	*u*	*e*	*d*	*r*
Array Index	*17*	*29*	*26*	*75*	*78*	*33*	*77*	*21*	*30*	*15*	*14*	*28*

According to these calculations, the numerical value of the http://www.ktu.edu.tr URL obtained with printable indexes will be as follows:

[18, 30, 30, 30, 26, 75, 78, 78, 33, 33, 33, 33, 77, 21,30, 31, 77, 15, 14, 31, 77, 30, 29]

2.3. Dataset Split Strategy

The next step is to train and test the dataset, and this step is performed after the URL information is converted to numbers. This process is performed with the “train_test_split” function in the “scikit-learn library”, which randomly splits the data according to the selected settings. Although there are different approaches to splitting the dataset, generally 25% is used for testing and 75% for training. This split is important because it lets the

model learn from a good chunk of data while also getting tested on new, unseen data. That way, you can really see how well it can generalize and perform in real-world situations.

3. DISCUSSION

The model created in the study consists of a total of 10 layers. The input layer takes the numerical representation of each URL as a parameter and transforms this data into 32-dimensional vectors in an embedding layer. These vectors are then passed through four 1-dimensional convolutional layers, each with a different "kernel_size" value. Each convolutional layer uses 256 filters and the ELU (Exponential Linear Unit) activation function. The ELU activation function reduces negative values towards zero and increases positive values linearly, but by allowing negative values to decrease exponentially, it enables more balanced learning.

Different "kernel_size" values allow the model to capture various patterns and relationships in the input data. The feature maps obtained from these layers are combined and processed with two fully connected (dense) layers. ELU activation functions and dropout layers are used in these layers to prevent overfitting. Finally, there is an output layer with a single neuron and a sigmoid activation function.

The model is compiled using the Adam (Adaptive Moment Estimation) optimization algorithm and the binary_crossentropy loss function (see Figure 9). The Adam optimization algorithm, a combination of momentum and RMSprop, adjusts the learning rate adaptively and uses separate learning rates for each parameter, providing a faster and more stable learning process. The "binary_crossentropy" loss function is a function used for binary classification problems. Measuring the difference between the predicted probabilities and the actual labels minimizes the error rate during the model's learning process.

Figure 9. Python code for the build phase of the CNN model

```
def build_model(self):
    main_input = Input(shape=(self.max_len,),
                       dtype='int32', name='main_input')
    emb = Embedding(input_dim=self.max_vocab_len, output_dim=self.emb_dim, input_length=self.max_len,
                    embeddings_regularizer=self.W_reg)(main_input)
    emb = Dropout(0.25)(emb)

    conv1 = self.get_conv_layer(emb, kernel_size=2, filters=256)
    conv2 = self.get_conv_layer(emb, kernel_size=3, filters=256)
    conv3 = self.get_conv_layer(emb, kernel_size=4, filters=256)
    conv4 = self.get_conv_layer(emb, kernel_size=5, filters=256)

    merged = concatenate([conv1, conv2, conv3, conv4], axis=1)

    hidden1 = Dense(1024)(merged)
    hidden1 = ELU()(hidden1)
    hidden1 = BatchNormalization()(hidden1)
    hidden1 = Dropout(0.5)(hidden1)

    hidden2 = Dense(1024)(hidden1)
    hidden2 = ELU()(hidden2)
    hidden2 = BatchNormalization()(hidden2)
    hidden2 = Dropout(0.5)(hidden2)

    output = Dense(1, activation='sigmoid', name='output')(hidden2)

    model = Model(inputs=[main_input], outputs=[output])

    adam = Adam(learning_rate=1e-4, beta_1=0.9, beta_2=0.999,
                epsilon=1e-08)
    model.compile(optimizer=adam, loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model
```

The training process for the model happens with the “fit” function. This function helps tweak the model's parameters based on the data to make it perform better. Basically, it trains the model using the training data for a set number of epochs and a specific batch size. An epoch is just one complete pass through all the training data. So, after each epoch, the model has seen everything once. Generally, the more epochs you run, the better the model gets, but you also risk overfitting, where it learns too much from the training data and doesn’t do well with new data.

Since training datasets can be pretty big, instead of feeding all that data to the model at once, it’s split into smaller mini-batches. The gradient descent algorithm works on each mini-batch, updating the model’s parameters along the way. This approach helps save memory and speeds up training.

While training is going on, the model also gets checked against validation data. This validation set helps see how well the model is doing overall during training. At the end of each epoch, they check how it performed on this dataset, which is a good way to keep an eye on overfitting.

Once training wraps up, they evaluate the model using test data with the “evaluate” function to look at its accuracy and loss values. After that, predictions are made on this test data using the “predict” function, turning those predictions into binary values like True or False. Finally, they use the classification_report function to compare these predictions with the actual labels and get a detailed look at how well the model performed.

Figure 10. Model test results

```
Epoch 1/10
10551/10551 ━━━━━━━━━━━━━━━━━━━━ 171s 16ms/step - accuracy: 0.7509 - loss: 0.5687 - val_accuracy: 0.8303 - val_loss: 0.4017
Epoch 2/10
10551/10551 ━━━━━━━━━━━━━━━━━━━━ 157s 15ms/step - accuracy: 0.8266 - loss: 0.4182 - val_accuracy: 0.8323 - val_loss: 0.3921
Epoch 3/10
10551/10551 ━━━━━━━━━━━━━━━━━━━━ 159s 15ms/step - accuracy: 0.8315 - loss: 0.4013 - val_accuracy: 0.8330 - val_loss: 0.3854
Epoch 4/10
10551/10551 ━━━━━━━━━━━━━━━━━━━━ 164s 16ms/step - accuracy: 0.8322 - loss: 0.3944 - val_accuracy: 0.8333 - val_loss: 0.3822
Epoch 5/10
10551/10551 ━━━━━━━━━━━━━━━━━━━━ 162s 15ms/step - accuracy: 0.8333 - loss: 0.3908 - val_accuracy: 0.8388 - val_loss: 0.3731
Epoch 6/10
10551/10551 ━━━━━━━━━━━━━━━━━━━━ 167s 16ms/step - accuracy: 0.8369 - loss: 0.3726 - val_accuracy: 0.8620 - val_loss: 0.3143
Epoch 7/10
10551/10551 ━━━━━━━━━━━━━━━━━━━━ 168s 16ms/step - accuracy: 0.8513 - loss: 0.3363 - val_accuracy: 0.8736 - val_loss: 0.2887
Epoch 8/10
10551/10551 ━━━━━━━━━━━━━━━━━━━━ 167s 16ms/step - accuracy: 0.8610 - loss: 0.3165 - val_accuracy: 0.8868 - val_loss: 0.2637
Epoch 9/10
10551/10551 ━━━━━━━━━━━━━━━━━━━━ 169s 16ms/step - accuracy: 0.8701 - loss: 0.2973 - val_accuracy: 0.8954 - val_loss: 0.2483
Epoch 10/10
10551/10551 ━━━━━━━━━━━━━━━━━━━━ 171s 16ms/step - accuracy: 0.8777 - loss: 0.2805 - val_accuracy: 0.8995 - val_loss: 0.2344
3517/3517 ━━━━━━━━━━━━━━━━━━━━ 23s 7ms/step - accuracy: 0.8994 - loss: 0.2353

Final Cross-Validation Accuracy 0.8995059728622437

3517/3517 ━━━━━━━━━━━━━━━━━━━━ 22s 6ms/step
              precision    recall  f1-score   support

       False    0.73747   0.71474   0.72592     20956
        True    0.93519   0.94178   0.93847     91588

    accuracy                        0.89951    112544
   macro avg    0.83633   0.82826   0.83220    112544
weighted avg    0.89837   0.89951   0.89890    112544
```

The training process of the CNN model was conducted with an epochs_num value of 10 and a batch_size value of 32. At the end of each epoch, the accuracy and loss values for both the training and validation data were reported as in Figure 10. As the epochs increased,

the training accuracy and validation accuracy gradually improved, while the loss values continuously decreased. At the end of the 10th epoch, the training accuracy was calculated to be 87.77%, and the validation accuracy was 89.95%. These results indicate that the model performed quite well and learned the data successfully. From a managerial perspective, the high validation accuracy of 89.95% demonstrates that this model can be effectively deployed as a proactive security layer in organizational network infrastructures. This capability translates directly into reduced financial losses from data breaches, enhanced customer trust, and significant operational efficiency gains by automating the detection of malicious URLs before they reach end-users.

The model's accuracy on the test dataset was calculated to be 89.95%, and the loss was 23.53%. These results demonstrate that the model maintained high classification success on new and previously unseen data.

The values used to evaluate the model's performance include precision, recall, f1-score, and support. Precision measures how many of the examples the model classified (predicted) as positive (harmful) are actually positive. It shows how many of the sites classified as harmful are truly harmful. A low rate of false positives is aimed for in the precision value. The test results showed that the model's accuracy in identifying harmful websites was 93.52%.

Recall, on the other hand, indicates how many of the truly harmful websites were successfully classified as harmful by the model. The recall test results showed that 94.18% of harmful websites were correctly classified as harmful by the model.

The f1-score is the harmonic mean of precision and recall values. It summarizes the model's overall performance by considering both false positives and false negatives. The test results showed that the model achieved an f1-score of 0.9385 for harmful sites, indicating high success in identifying harmful websites.

The support value, which indicates the number of examples present in the dataset for each class, was 91,588 for harmful sites and 20,956 for harmless sites.

The precision (93.52%) and recall (94.18%) values obtained for the harmful (True) class show that the tested model is successful in identifying harmful sites. The high precision value indicates that most of the sites classified as harmful by the model are indeed harmful, while the high recall value indicates that most of the harmful sites are correctly identified.

4. CONCLUSION

A Statista report says there are over 5.44 billion internet users turning in the world, or 67.1% of the global population, as of April 2024 (Statista, 2024). Malicious websites aiming to cause trouble are all the rage, now that so many people are online. This means that, out of nearly 2 billion sites online, more than 20 million of them are bad, and we need to figure out if a site is safe.

In this study, we ask how we can detect if a web page is malicious by using a machine learning algorithm to verify any URL. One keyway to avoid getting pulled into those bad sites: using antivirus software. It's also important to stay up to date with your web browsers and operating systems so you can fix security issues as they become known and lower risks.

With strong passwords and avoiding suspicious email links, you can stop phishing attempts to keep your accounts safe from a brute force attacks. With everything just described, however, you can't guarantee that you will catch every malicious website.

More people are getting more interested in artificial intelligence, and more and more of us are using machine learning algorithms to solve cybersecurity problems. If this sounds like a problem that could be solved by supervised learning algorithms, then the point is taken because supervised learning algorithms can be trained with labelled data to spot phishing attempts, malware distribution, and harmful websites. And big roles are also played by Convolutional Neural Networks (CNNs). They are not only for image processing, they're also for classification and recognition. The layers in these networks are effectively built to extract the important features from the data and make the networks capable to detect and classify the cyber threats such as malware and bad URLs. And CNNs are themselves known for being extremely good at learning and scaling up, whatever you need.

The dataset used in the study consists of 450,176 labeled URL records. Of these, 345,738 are labeled as non-malicious URLs (77% of the total addresses), while 104,438 are labeled as malicious URLs (23% of the total addresses). Since CNN is generally a deep learning model used in image processing, in the preprocessing step, the URL information needs to be converted into numerical values and presented to the algorithm in an appropriate format as a 2D matrix. The numerical conversion of URL information is performed considering whether the characters it contains are printable.

After the numerical conversion of URL information, the dataset is divided into training (75%) and test (25%) sets. Each convolutional layer of the 10-layer model uses 256 filters. The feature maps obtained from these layers are combined and processed with two fully connected layers. Finally, there is an output layer with a single neuron and a sigmoid activation function.

During the training and testing processes, conducted with epoch and batch values of 10 and 32, respectively, it was observed that the training accuracy and validation accuracy gradually increased. The final epoch's training accuracy of 87.77% and validation accuracy of 89.95% indicate that the model has successfully learned the data.

Precision, recall, F1 score and support values were used to cheque model's performance. Any website was also attempted to be detected as malicious, and it achieved an accuracy of 93.52%, a recall of 94.18, and an F1 score of 0.9385. It turns out, these numbers do a great job in revealing what the model does best, which is to predict whether a site is harmful, given a URL.

True class precision of 93.52% and True class recall of 94.18% shows how well the model is at identifying those trouble sites. Overall, this is very promising indicating the strong performance of the model in keeping users safe online.

Statement of Research and Publication Ethics

All processes related to the manuscript have been conducted in accordance with the research and publication ethics principles of Manisa Celal Bayar University Journal of Social Sciences Institute.

Authors' Contribution Statement

The entire manuscript was written by the author.

Conflict of Interest Statement

There is no potential conflict of interest in this study.

REFERENCE

Abdillah R., Shukur Z., Mohd M. & Murah T. M. Z. (2022). Phishing Classification Techniques: A Systematic Literature Review. IEEE Access, 10(1), pp. 41574-41591. Doi: 10.1109/ACCESS.2022.3166474.

Abomhara, M., Køien G. M. (2015). Cyber Security and the Internet of Things: Vulnerabilities, Threats, Intruders and Attacks. Journal of Cyber Security and Mobility. Vol 4. Issue 1. https://doi.org/10.13052/jcsm2245-1439.414

Ahsenali, C. J., Chaudhry S. A., Rittenhouse R. G. (2016). Phishing Attacks and Defenses. International Journal of Security and Its Applications 10(1), pp.247-256 http://dx.doi.org/10.14257/ijsia.2016.10.1.23

APWG - Anti-Phishing Attacks Working Group (2023). Phishing Attack Trends Report. Erişim Adresi: https://apwg.org/trendsreports/

APWG - Anti-Phishing Attacks Working Group (2024). Phishing Attack Trends Report. Erişim Adresi: https://apwg.org/trendsreports/

Arachchilage, N. A. G. & Love, S. (2014). Security awareness of computer users: A phishing threat avoidance perspective. Computers in Human Behavior. 38(1), pp. 304-312. Doi: 10.1016/j.chb.2014.05.046

Atıcı, H. & Kocer H. E. (2023). "Mask R-CNN Based Segmentation and Classification of Blood Smear Images", GMBD, 9(1), pp. 128–143.

Bhavsar, V., Kadlak, A. & Sharma, S. (2018). Study on Phishing Attacks. International Journal of Computer Applications. Vol 182, No. 33.

Chanti, S. & Chithralekha T. (2022). A literature review on classification of phishing attacks. International Journal of Advanced Technology and Engineering Exploration (IJATEE). 9(89), pp. 446-476. Doi: 10.19101/IJATEE.2021.875031

Convolutional neural networks. https://stanford.edu/~shervine/l/tr/teaching/cs-230/cheatsheet-convolutional-neural-networks

Dodge R. C., Carver C. & Ferguson A. J. (2007). Phishing for user security awareness. Computers & Security. 26(1), pp 73-80. Doi: 10.1016/j.cose.2006.10.009

Dua, M., Gupta, R., Khari, M. et al. Biometric iris recognition using radial basis function neural network. Soft Comput 23, 11801–11815 (2019). https://doi.org/10.1007/s00500-018-03731-4

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

Hong, J. (2012), The state of phishing attacks. Communications of the ACM, Volume 55, Issue 1. Pages 74 - 81 https://doi.org/10.1145/2063176.2063197

Kim, P. (2017). Convolutional Neural Network. In: MATLAB Deep Learning. Apress. Berkeley, CA. https://doi.org/10.1007/978-1-4842-2845-6_6

Korkmaz, A., & Büyükgöze, S. (2019). Sahte Web Sitelerinin Sınıflandırma Algoritmaları ile Tespit Edilmesi. Avrupa Bilim ve Teknoloji Dergisi (16), pp. 826-833. https://doi.org/10.31590/ejosat.598036

Kumar, S. (2019) Malicious and Benign URLs. (https://www.kaggle.com/datasets/siddharthkumar25/malicious-and-benign-urls)

Kumaraguru, P., Rhee, Y., Acquisti, A., Cranor, L. F., Hong, J., & Nunge, E. (2007). Protecting people from phishing. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI. Doi:10.1145/1240624.1240760

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. Nature 521, pp. 436–444. https://doi.org/10.1038/nature14539

Manjeri, K. R., M.N.V. A. & Nair, P. C. (2019). "A Machine Learning Approach for Detecting Malicious Websites using URL Features," 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, pp. 555-561, doi: 10.1109/ICECA.2019.8821879.

Oest, A., Safei Y., Doupé A., Ahn G. -J., Wardman B. and Warner G., "Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis," 2018 APWG Symposium on Electronic Crime Research (eCrime), San Diego, CA, USA, 2018, pp. 1-12, doi: 10.1109/ECRIME.2018.8376206.

Özmen, N. E. (2022). Derin Öğrenme Yöntemleri ile Görüntü Sahteciliği Tespiti. Tekirdağ Namik Kemal University, Department of Computer Engineering. [Master Thesis]. (https://acikerisim.nku.edu.tr/xmlui/bitstream/handle/20.500.11776/11474/746987.pdf?sequence=1&isAllowed=y)

Şeker, A., Diri B., & Balık H. H. (2017). "Derin Öğrenme Yöntemleri ve Uygulamaları Hakkında Bir İnceleme", GMBD, 3(3), pp. 47–64.

Singhal, S., Chawla U. & Shorey R., "Kötü Amaçlı Web Sitesi Tespiti için Makine Öğrenimi ve Kavram Kaymasına Dayalı Yaklaşım", 2020 Uluslararası İletişim Sistemleri ve Ağları Konferansı (COMSNETS), Bengaluru, Hindistan, 2020, s. 582-585, doi: 10.1109/COMSNETS48256.2020.9027485.

Sinha R.S, Hwang S-H. Comparison of CNN Applications for RSSI-Based Fingerprint Indoor Localization. Electronics. 2019; 8(9):989. https://doi.org/10.3390/electronics8090989

Statista (2024). Number of internet and social media users worldwide as of April 2024. https://www.statista.com/statistics/617136/digital-population-worldwide/#:~:text=As%20of%20April%202024%2C%20there,population%2C%20were%20social%20media%20users

Torska, Nina (2023). 250+ Kimlik Avı İstatistiği: Türler, Maliyet ve Daha Fazlası. Erişim Adresi: https://marketsplash.com/tr/oltalama-istatistikleri/#link8

Utku A. & Can Ü. (2022), "Makine Öğrenmesine Dayalı Etkili Kötü Amaçlı Web Sayfası Tespiti", IJISS, 11(4) pp. 28–39.

Verma, A., & Shri, C. (2022). Cyber Security: A Review of Cyber Crimes, Security Challenges and Measures to Control. Vision. https://doi.org/10.1177/09722629221074760

Vishwanath A., Herath T., Chen R., Wang J. & Rao H. R. (2011). Why do people get phished? Testing individual differences in phishing vulnerability within an integrated, information processing model. Decision Support Systems. 51(3), pp. 576-586. Doi: 10.1016/j.dss.2011.03.002