Enhancing IoT Security: Proactive Phishing Website Detection Using Deep Neural Networks

Case study: Smart Home

Habiba Bouijij
SSL Lab, ENSIAS, Mohammed V University in Rabat, Morocco

Amine Berqia
SSL Lab, ENSIAS, Mohammed V University in Rabat, Morocco

Abstract: The Internet of Things (IoT) has proven its utility across various domains, including healthcare, agriculture, industry, and finance. It comprises Internet-connected devices that offer remote control capabilities. However, this very connectivity exposes these devices to potential cyberattacks. Cybercriminals can exploit the vulnerabilities in these devices by sending deceptive emails or text messages containing malicious links leading to hacker websites or destructive applications. This allows them unauthorized access to connected devices and the acquisition of sensitive personal information. Such malicious tactics are collectively known as phishing and pose one of the most prevalent threats.

This article presents an innovative method that harnesses the power of a Deep Neural Network to accurately classify and proactively prevent phishing websites by analyzing their URLs. The method is demonstrated through a smart home use case, aiming to reinforce IoT security and safeguard users' sensitive data by proactively identifying and preventing phishing attacks. By harnessing the power of the Deep Learning model, this innovative technique seeks to enhance online safety and protect users from potential cyber threats.

Keywords: Smart Home, Phishing, URL Classification, Security, Deep Neural Network.

Introduction

IoT, short for the Internet of Things, encompasses a network of interconnected devices and appliances that are connected to the Internet. This technology has had a profound impact on our daily lives, enhancing the capabilities of machines to handle demanding tasks, addressing monotonous activities, and enriching our quality of life by providing healthier, more productive, and comfortable experiences.

http://doi.org/10.18080/jtde.v12n1.822

However, this connectivity also exposes us to cybercrime attacks, with one of the most common being phishing attacks. These attacks typically involve tricking users through emails or text messages, leading them to visit seemingly legitimate websites and click on malicious links. As a result, cybercriminals can exploit this technique to collect personal, intimate, confidential, and sensitive information (Abbas *et al.*, 2021; Atitallah *et al.*, 2020; Safi & Singh, 2023).

Phishing attacks pose a significant threat to IoT devices, often leading to data breaches. To mitigate these risks, it is essential to establish a strong security mechanism that can identify potential threats, vulnerabilities, and countermeasures to prevent phishing attacks on IoT devices (Abbas *et al.*, 2021; Atitallah *et al.*, 2020).

This work delves into the endeavours of researchers to improve the cybersecurity of IoT systems and the diverse models suggested for countering phishing attacks. Specifically, we will focus on detecting and preventing phishing attempts in smart homes, which are considered an integral part of the IoT ecosystem. As a part of our research, we will introduce our proposed model based on Deep Neural Networks (DNNs).

The paper presentation will follow a structured format comprising various sections. The Literature Review will provide an overview of the addressed problem and introduce key concepts related to smart homes, exploring existing literature on detecting phishing attacks in the IoT. The Vocabulary and Methods section will outline the terminology and methodologies employed. The Methodology and Experience section will detail the steps taken to develop the proposed model. Results and Discussion will focus on presenting and analyzing research findings, discussing their implications and potential applications. The Comparative Analysis section will compare results with other learning models. Finally, the Conclusion and Future Work section will succinctly summarize key findings, highlight their implications, and propose potential avenues for future research to advance the field.

Literature Review

IoT is a groundbreaking technology facilitating seamless connections between objects and the Internet, reshaping how we interact with our surroundings. Intelligent devices within this framework collect and transmit data through IoT applications, subject to analysis using advanced artificial intelligence techniques. This empowers the IoT system to make informed decisions, which are then conveyed back to devices for intelligent responses through automated actions. Users can easily manage and control these IoT devices through user-friendly graphical interfaces. Various IoT systems exist, such as smart cars, smart homes, smart cities, and smart buildings. This paper specifically focuses on highlighting the smart home as a prominent example of IoT application (Abbas *et al.*, 2021).

http://doi.org/10.18080/jtde.v12n1.822

Smart home technologies

A smart home is characterized by the presence of a communication network that connects various electronic products or services, enabling remote control or monitoring. Various objects contribute to this system, including light bulbs, blinds, outlets, thermostats, and TVs. Even refrigerators and vacuum cleaners can be integrated into this ecosystem.

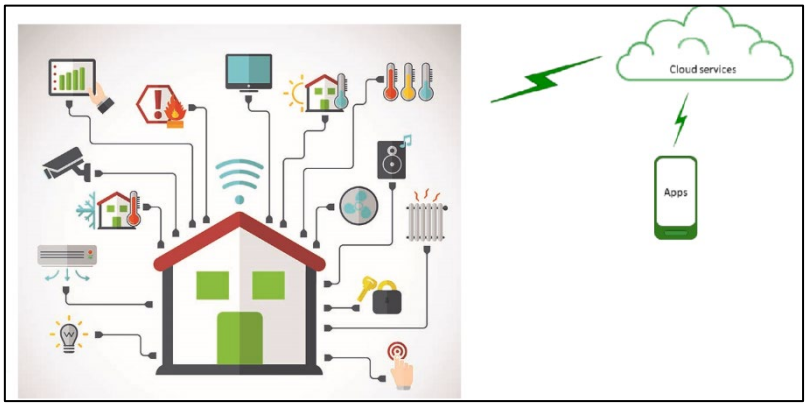

Figure 1. Smart Home

The smart home can be divided into three key areas: devices, cloud services, and consumer. In the devices sector, diverse smart devices are interconnected to establish the fundamental framework of the smart home ecosystem. The cloud services zone acts as a centralized hub facilitating communication and data exchange between these devices and cloud-based services, enhancing functionality and accessibility. Finally, the consumer zone serves as the interface where end-users interact with and manage their smart home system, providing a seamless and personalized experience. This tripartite division allows for a clear delineation of the essential components and their respective roles within the smart home framework.

Smart home security and risks

A smart home encounters various security threats, with the primary issue being the inadequate protection of individual devices. Some IoT home devices are rushed to market without properly addressing security concerns. User manuals often overlook privacy matters or lack sufficient information to guarantee the safety of the devices. Experts stress the importance of anticipating device breaches rather than dismissing them as unlikely events, as most IoT devices are easily exploited and provide minimal protection.

Furthermore, the integrity of a smart home network may be jeopardized, providing unauthorized individuals with the ability to retrieve stored data. Malevolent actors could observe patterns in device usage to identify periods when the homeowner is absent, posing a significant security risk. Administering a smart home network through the primary Internet account exposes not only data from IoT devices but also places various personal information

http://doi.org/10.18080/jtde.v12n1.822

at risk, such as emails, social media accounts, and even banking details. Among the primary vulnerabilities that can lead to data breaches, phishing remains a significant threat.

In this study, we propose a deep learning (DL)-based model aimed at identifying phishing URLs. This model can be integrated into the smart home system by configuring a proxy at the modem level to utilize the model for redirecting web traffic. The proxy is thus capable of leveraging the phishing detection model to assess URLs before transmitting them to connected devices.

Phishing

Phishing is a cybercrime where malicious actors impersonating legitimate institutions communicate with individuals through telephone, email or text messages. The intention behind these deceptive communications is to deceive individuals into revealing sensitive information, including personally identifiable passwords, credit card and banking authorizations, and data. Once these details are acquired, the perpetrators can access crucial accounts, leading to potential identity theft and financial losses (Abbas *et al.*, 2021; Safi & Singh, 2023).

In the cyber world, phishing attacks have experienced a significant surge over the last few years. As per the Cybermalveillance.gouv.fr platform, the main trends observed in cybermalveillance during 2021 are as follows (Cybermalveillance, n.d.):

- Phishing continues to be the most significant cyber threat faced by individuals and has seen a notable increase (+82%);
- Online account hacking (+58%) and fraudulent activities related to fake technical support (+18%) are also on the rise.

Additionally, according to APWG (Anti-Phishing Working Group) report, during the fourth quarter of 2022, a significant number of 1,350,037 phishing attacks were recorded. Figure1 illustrates a slight increase compared to the previous quarter, which had already been identified as the worst quarter for phishing attacks ever recorded by the APWG (n.d.).

http://doi.org/10.18080/jtde.v12n1.822

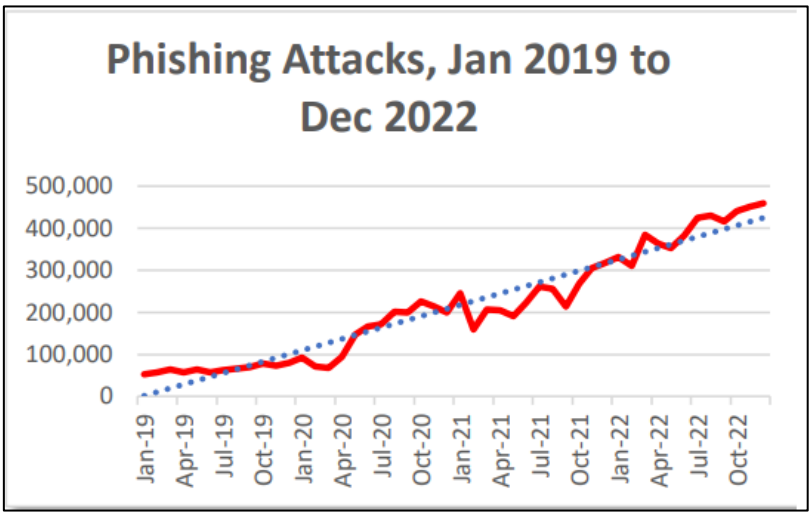

Figure 2. Phishing Attacks, 2019–2022 (APWG, n.d., 4th Quarter 2022)

This work will focus on phishing websites that employ malicious URLs, which are among the most commonly used techniques in phishing attacks. These fraudulent URLs are crafted to trick users into believing they are accessing a reliable website; however, in reality, users unwittingly share their information with cybercriminals.

Phishing website detection

Detecting phishing websites can be challenging due to their deceptive nature. However, several approaches and techniques can help identify them (Barraclough *et al.*, 2013; Berqia & Nacsimento, 2004; Ding *et al.*, 2019; Sahingoz *et al.*, 2019; Yuan *et al.*, 2021):

URL Analysis: Examine the URL for misspellings, extra characters, or variations from the legitimate website. Phishing sites often use URLs that mimic the original domain to deceive users.

Webpage Content Analysis: Analyze the content of the webpage for inconsistencies, poor design, or suspicious elements like fake login forms or requests for personal information.

Website Reputation Services: Utilize reputation services or blacklists that maintain databases of known phishing sites and malicious domains.

DNS Monitoring: Monitor DNS records for changes that may indicate the creation of a phishing site.

Detection through Deep Learning or Machine Learning: Employ deep learning models and machine learning algorithms to analyze features of known phishing sites and create models to detect new ones based on patterns.

Otherwise, numerous literature reviews have been conducted on the topic of phishing detection methods. Researchers have compared various approaches, such as Heuristic, Visual Similarity, Lists Based, Deep Learning models and Machine Learning techniques. The research involves a comprehensive investigation into the algorithms, datasets, and methodologies employed for the detection of phishing websites. Notably, Deep Learning and

http://doi.org/10.18080/jtde.v12n1.822

Machine Learning skills are the most prevalent and have demonstrated remarkable accuracy, surpassing 99%, in detecting phishing websites (Bouijij *et al.*, 2022; Liu *et al.*, 2022a; Safi & Singh, 2023; Wei *et al.*, 2020).

Manuel Sánchez-Paniagua and his team focus on anti-phishing techniques utilizing machine learning. They make use of HTML, URL, and web technology features. Their experiments illustrate that by employing a LightGBM classifier alongside a comprehensive set of 54 selected features, phishing websites can be accurately detected with a precision of 97.95%. This evaluation was conducted on the Phishing Index Login Websites Dataset (PILWD) (Sánchez-Paniagua *et al.*, 2022).

Dong-Jie Liu and Guang-Gang Geng introduce three semantic-based phishing detection models leveraging diverse deep learning techniques. These models, namely the Multi-scale In-depth Fusion (MIF) model, Multi-scale Feature-layer Fusion (MFF) model, and Multi-scale Data-layer Fusion (MDF) model, have undergone rigorous testing on a complex dataset, showcasing exceptional recognition capabilities. With an accuracy exceeding 99%, these models demonstrate high effectiveness in detecting phishing attacks (Liu *et al.*, 2022b).

Hamzah Salah and Hiba Zuhair propose a phishing predictive model that that employs a dual deep learning-based architecture, specifically combining Bi-directional Long Short-Term Memory (CNN-BiLSTM) with Convolutional Neural Network. The model utilizes a feature space comprising 60 mutual features from Uniform Resource Locators (URLs). Through experimentation, the model achieves impressive performance with an accuracy of 99.27% (Salah & Zuhair, 2023).

M. A. Adebowale and his team introduce an Adaptive Neuro-Fuzzy Inference System (ANFIS) constructed robust scheme that integrates features from images, text and frames for phishing website detection. By using related features of authentic and non-authentic websites and employing artificial intelligence procedures, this combined approach achieves an impressive accuracy of 98.3% (Adebowale *et al.*, 2019).

Syed Ghazanfar Abbas and his team (Abbas *et al.*, 2021) suggest a threat modelling methodology to classify and mitigate cyber threats that could result in phishing attacks on IoT devices. The study centres around two significant IoT use cases: smart homes and smart autonomous vehicular systems. Applying the STRIDE threat modelling approach, the paper identifies potential threats that have the potential to initiate phishing attacks in both of these use cases.

Sujatha Rajkumar and his team, introduce Deep Learning (DL) techniques, specifically a layered approach called Stacked Long Short-Term Memory (SLSTM), to detect and distinguish between normal and malicious traffic data in real-time IoT environments. The

http://doi.org/10.18080/jtde.v12n1.822

SLSTM method shows promising results in identifying most IoT attacks and outperforms existing classification methods in terms of real-time detection rates. The approach can be applied to different IoT scenarios, making it valuable for improving the security and reliability of IoT systems (Rajkumar *et al.*, 2023).

Vocabulary and Methods

In this section, we will begin by defining the vocabularies and techniques used to implement our model. Subsequently, we will describe our model in detail.

Deep Neural Network

A Deep Neural Network (DNN) takes inspiration from the neurons in the human brain, which generate outputs in response to external stimuli. Specifically, it is a type of artificial neural network that comprises numerous hidden layers positioned between the input and output layers. Each layer in a DNN contains neurons or nodes that perform mathematical operations on the data they receive. The deeper the network, the more hidden layers it has, and the more complex patterns it can learn. The term "deep" in Deep Neural Networks refers to the multiple layers used to process and learn from the data.

Dense layer

The dense layer, also referred to as the fully connected layer, constitutes a fundamental element in neural networks. Within this layer, every neuron is connected to all neurons in the preceding layer. The dense layer performs matrix-vector multiplication, using the output from the preceding layer as the row vector and the neurons in the dense layer as the column vector. The dense layer assumes a critical role in artificial neural networks, as it is extensively employed and crucial for learning patterns and making predictions grounded on input data.

Dropout

Dropout is a regularization method ordinarily used during the training of neural networks to prevent overfitting (Sabiri *et al.*, 2022). It includes randomly zeroing a portion of input units (neurons) during each update in the training process. This technique prevents the model from overly relying on particular neurons and promotes the learning of more robust features within the network. The decision regarding which neurons to drop out is made randomly during each training iteration. The dropout rate determines the fraction of neurons to be dropped out, typically ranging from 0.2 to 0.5.

http://doi.org/10.18080/jtde.v12n1.822

Activation function

The activation function plays a vital role in artificial neural networks by introducing non-linearity. It transforms input data and passes it as output to the next layer, enabling the neural network to learn complex patterns and relationships within the data. Common activation functions like *ReLU* (Rectified Linear Unit), sigmoid, tanh (hyperbolic tangent), and softmax each have unique strengths and weaknesses. The choice of an activation function can greatly impact the neural network's performance across different tasks.

In this work, we use:

- *ReLU* function: it retains the positive input values without altering them, while converting all negative values to zero. Mathematically, *ReLU* is presented as:

$$f(x) = \max(0, x) \tag{1}$$

- *Sigmoid* function: a widely adopted non-linear activation function in artificial neural networks, it maps input values to a range of 0 to 1. This attribute makes it particularly suitable for tasks that entail binary classification and probability predictions. Mathematically, the sigmoid function is presented as:

$$f(x) = \frac{1}{(1+e^{-x})} \tag{2}$$

Dropna

"Dropna" refers to a function used to remove missing or NaN (Not a Number) values from a dataset. It is one of a prevalent data preprocessing procedure in data analysis to handle missing data before feeding it into a model.

In Python, "dropna" is commonly associated with Pandas, a standard library for data analysis and manipulation ("Pandas Tutorial", 2024). Precisely, the "dropna" function is used to drop rows or columns containing missing values from a Pandas DataFrame.

Tokenizer

Tokenization involves the procedure of dividing a given text into reduced segments or tokens. These tokens can encompass various elements, ranging from individual words and characters to subwords. In the Keras tokenizer class (Keras, n.d.), the "*fit_on_texts*" method is employed to update the internal vocabulary based on the texts list provided. It is essential to call this method before utilizing other functions like "*texts_to_sequences*" or "*texts_to_matrix*". This ensures that the tokenizer is properly prepared to handle the text data and perform subsequent tokenization tasks accurately. In this study, we employ the "*texts_to_sequences*" method, which simplifies the transformation of tokens in a text corpus into a sequence of integers.

http://doi.org/10.18080/jtde.v12n1.822

Pad sequences

"Pad sequences" is a function frequently used in natural language processing and sequence-modelling tasks to guarantee that all sequences in a dataset have the identical length, by padding or truncating them as needed. The purpose of this function is to guarantee that all sequences in a list have equal lengths. By default, it achieves this by padding each sequence with 0 at the beginning until they all match the length of the longest sequence.

In Python, "*pad_sequences*" is typically part of libraries such as Keras (Keras, n.d.) or TensorFlow (TensorFlow, n.d.), specifically within their sequence or preprocessing modules. It simplifies the data preprocessing step before feeding the sequences into the neural network, as most models require fixed-length input sequences.

Evaluation metrics

In a classification problem, the objective is to allocate samples to particular categories or classes. Various evaluation metrics are employed to gauge the efficiency of a classification model. Here are several commonly utilized evaluation metrics for classification tasks:

Conf_Matrix: presents the model's predictions in a tabular format, comprising four values: TN, FP, FN and TP. This matrix offers a comprehensive overview of how samples were accurately or inaccurately classified for each class.

		Predicted class	
		Class=0 (Benign)	Class=1 (Phishing)
Real class	Class=0 (Benign)	True Positive (TP)	False Negative (FN)
	Class=1 (Phishing)	False Positive (FP)	True Negative (TN)

Figure 3. Confusion Matrix

Precision: is a metric that signifies the proportion of true positives among all positive predictions made by the model:

$$precision = \frac{TP}{TP+FP} \tag{3}$$

Accuracy: serves as a performance measure for the model, indicating the percentage of correctly predicted samples (including true positives and true negatives) out of the total sample size. It offers a comprehensive evaluation of the model's accuracy in predicting both positive and negative samples:

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{4}$$

F1_score: is calculated as the harmonic mean of precision and recall, offering a balanced metric that takes into account both false positives and false negatives. This metric proves valuable, particularly in scenarios with class imbalances:

http://doi.org/10.18080/jtde.v12n1.822

$$F1 - score = 2 * \frac{precision * recall}{precision + recall} \quad (5)$$

Recall: referred to as sensitivity or true positive rate, measures the proportion of true positive predictions relative to the total number of actual positive samples. It indicates the model's capability to correctly identify positive samples among all the true positive samples:

$$recall = \frac{TP}{TP + FN} \quad (6)$$

Loss metric: is employed to assess the model's performance throughout the training phase. It measures the difference between the model's predicted values and the actual target values in the training data. Throughout training, the deep neural network strives to minimize this loss metric by iteratively adjusting its parameters (weights and biases), typically using techniques such as gradient descent or its variations. As the model learns from the data, the loss metric should ideally decrease over epochs, indicating that the model is converging to a solution that performs well on the training data.

Hardware and framework:

For this project, we opted for a machine running on the 64-bit operating system, Windows 10, with the following configuration: Intel® Core™ i5-8350U CPU @ 1.70GHz 1.90 GHz processor and 16 GB RAM. As for the framework, we chose Keras (n.d.), a high-level Python library, to interact with TensorFlow (n.d.), the machine learning framework developed by Google.

Methodology and Experience

In this section, we outline the systematic approach taken to execute the project. Python serves as the programming language, Jupyter Notebook functions as the web-based interactive computing platform, and Scikit-learn (n.d.) and Keras are employed as open-source libraries, serving as the implementation toolkit.

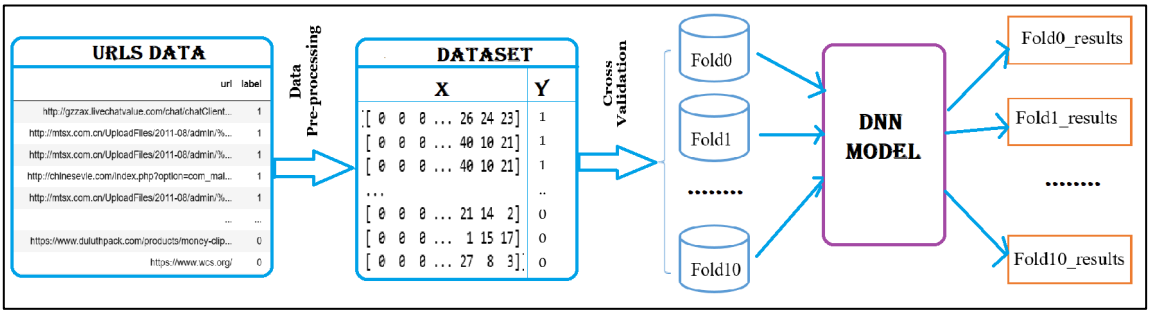

Figure 4. Summary of the proposed steps to implement our project.

URL data

To construct a Deep Neural Network (DNN) model designed for URL classification, it is essential to acquire a dataset comprising URLs along with their respective target labels or features. For this paper, we gather data on 40,000 URLs from two distinct sources: Mendeley

http://doi.org/10.18080/jtde.v12n1.822

Data (n.d.) and the Kaggle platform (Kaggle, n.d.). URLs from these sources are categorized into two classes, where 20,000 phishing URLs are labelled with 1, and 20,000 benign URLs are labelled with 0.

Data preprocessing

To achieve effective training of the neural network, it is crucial to ensure that the data is in a suitable format. The subsequent steps outline common data preprocessing procedures, defined earlier.

Table 1. Python script of the utilized methods

Step	Python script
Data Cleaning	Data_set = pd.read_csv('dataf.csv') data_set = data_set.dropna()
Tokenizer	Tokenizer_url = Tokenizer(char_level=True) Tokenizer_url.fit_on_texts(urls) Seq_url = tokenizer_url.texts_to_sequences(urls)
Pad_sequences	max_sequence_length = max(len(seq) for seq in seq_url) seq_url = pad_sequences(seq_url, maxlen= max_sequence_length)

Cross-validation

Cross-validation is a technique employed for model validation, assessing the generalization performance of a statistical analysis on an independent dataset. It is particularly useful in predictive scenarios, such as estimating the accuracy of a predictive model in real-world applications. A popular variant in machine learning is K-Fold cross-validation, where the dataset is divided into k equal-sized folds. The model is trained on k-1 folds, with the remaining fold reserved for testing. In this study, "Repeated Stratified K-Fold cross-validation" was chosen. Stratified K-Fold ensures a balanced representation of target classes in each subset, and the term "Repeated" signifies the process's iteration for enhanced robustness (Bouijij & Berqia, 2021; Scikit-Learn, n.d.). The Python script in Figure 5 demonstrates the implementation of Repeated Stratified K-Fold cross-validation.

```
from sklearn.model_selection import RepeatedStratifiedKFold
RepeatedStratifiedKFold(n_splits=5, n_repeats=2, random_state=1)
```

Figure 5. RepeatedStratifiedKFold python script

Deep Neural Network model

We have proposed a DNN model that consists of a sequence of dense (fully connected) layers with decreasing output sizes (256, 128, 64, 1) until it reaches an output of dimension (batch_size, 1). See Figure 6.

http://doi.org/10.18080/jtde.v12n1.822

Figure 7 illustrates the sequential model of the deep neural network that we have implemented.

```
# Import Keras library
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

# Create a sequential model
model = Sequential()

# Add a Dense layer
model.add(Dense(256, input_dim= max_sequence_length, activation='relu'))

# Add a Dropout layer with a rate of 0.2
# (each neuron has a 20 in 100 chances of being turned off in training)
model.add(Dropout(0.2))

# Add a Dense layer
model.add(Dense(128, activation='relu'))

# Add a Dense layer
model.add(Dense(64, activation='relu'))

# Add a Dense layer
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 6. DNN python script

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 256)               33024

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dense_2 (Dense)             (None, 64)                8256

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 74,241
Trainable params: 74,241
Non-trainable params: 0
_________________________________________________________________
```

Figure 7. Proposed sequential model

Results and Discussion

We used the previously discussed methods and techniques, including data preprocessing and cross-validation, and computed evaluation metrics for each fold. Figure 8 outlines the outcomes of our experimentation.

http://doi.org/10.18080/jtde.v12n1.822

We note that the average precision varies within the range of approximately 93% to 99.70%, accuracy from 92% to 99.69%, F1_score from 90% to 99.59%, and recall from 87.42% to 99.47%.

	Fold	Precision	Accuracy	F1_score	Recall	Conf_Matrix
0	Fold0	0.930143	0.927125	0.901337	0.874261	[[4754, 200], [383, 2663]]
1	Fold1	0.966243	0.968125	0.957760	0.949425	[[4854, 101], [154, 2891]]
2	Fold2	0.981636	0.980000	0.973510	0.965517	[[4900, 55], [105, 2940]]
3	Fold3	0.990716	0.989375	0.985976	0.981281	[[4927, 28], [57, 2988]]
4	Fold4	0.988177	0.991000	0.988177	0.988177	[[4919, 36], [36, 3009]]
5	Fold5	0.991359	0.988875	0.985301	0.979317	[[4928, 26], [63, 2983]]
6	Fold6	0.992452	0.994500	0.992777	0.993103	[[4932, 23], [21, 3024]]
7	Fold7	0.997038	0.996875	0.995890	0.994745	[[4946, 9], [16, 3029]]
8	Fold8	0.996040	0.995125	0.993580	0.991133	[[4943, 12], [27, 3018]]
9	Fold9	0.993115	0.995375	0.993929	0.994745	[[4934, 21], [16, 3029]]

Figure 8. DNN model results

The best results were achieved with the dataset of Fold 7, demonstrating exceptional performance:

- Precision: 99.70%;
- Accuracy: 99.69%;
- F1 Score: 99.59%;
- Recall: 99.47%;
- Confusion matrix as follows:
    - True Positives (TP): 3,029 (properly predicted positive samples);
    - False Positives (FP): 9 (negative samples erroneously classified as positive);
    - True Negatives (TN): 4,946 (correctly predicted negative samples);
    - False Negatives (FN): 16 (positive samples incorrectly classified as negative);

These high metrics and the detailed confusion matrix indicate that the model achieved near-perfect classification on Fold 7, with a very small number of misclassifications.

In a similar manner, on Fold 7, we provide the graphs (Figure 9 and Figure 10) illustrating the accuracy function and loss function during the training process.

In Figure 9, it is evident that, throughout the 100 epochs of training the DNN model with the dataset derived from Fold 7, the loss metric consistently declined from an initial value of 0.020

http://doi.org/10.18080/jtde.v12n1.822

to a final value of 0.009. The reduction in loss implies a continuous enhancement in the model's performance over the training period.

Figure 10 illustrates that the accuracy metric, monitored during the training process specifically on Fold 7, exhibited a significant improvement. Commencing at 99.30%, it steadily rose to attain a higher accuracy level of 99.75%. This upward trend in accuracy suggests that the model's ability to correctly classify instances improved during the training iterations on the Fold 7 dataset. The recorded increase in accuracy indicates a positive refinement in the model's performance, signifying its enhanced capability to make accurate predictions over the course of training.

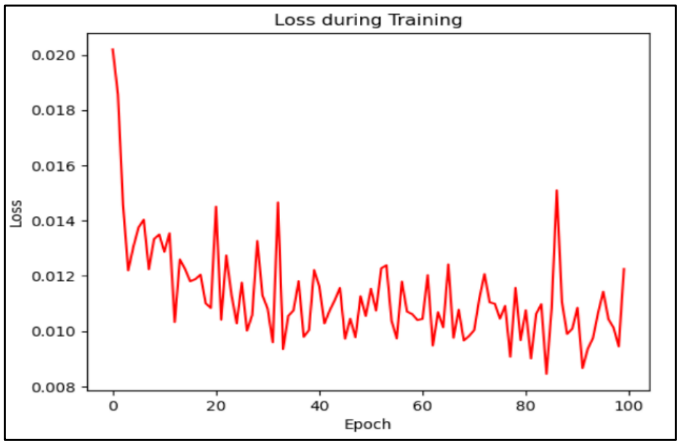

Figure 9. Loss metric recorded during the training process on Fold 7

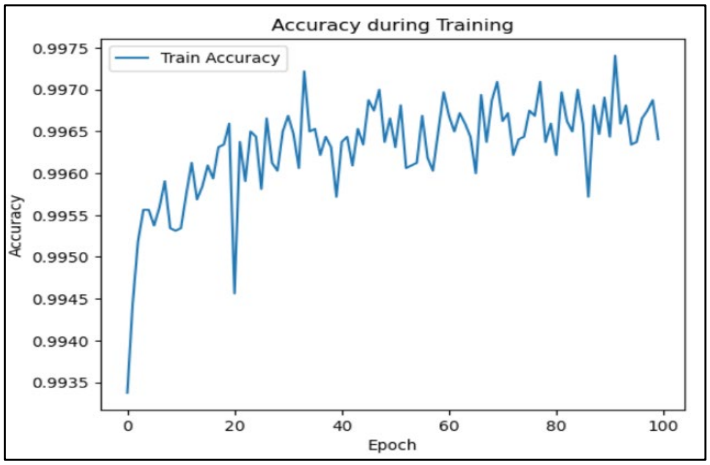

Figure 10. Accuracy metric recorded during the training process on Fold 7

Comparative Analysis

Table 3 presents a comparative analysis between our model and previously mentioned works, focusing on performance metrics related to the classification of phishing websites. The evaluated metrics encompass precision, accuracy, F1 score, and recall. Furthermore, the table provides insights into the utilized algorithms and the techniques applied for feature extraction in the respective studies.

http://doi.org/10.18080/jtde.v12n1.822

Table 3. Comparison of our results and other work

Authors	Approach	Results
Sánchez-Paniagua *et al.*	-HTML, URL and web technology features -ML (LightGBM)	Accuracy of 97.95%
Adebowale *et al.*	-Features of images, frames and text -Adaptive Neuro-Fuzzy Inference System	Accuracy of 98.30%
Salah & Zuhair	URL and CNN-BiLSTM	Accuracy of 99%
Our paper	URL and DNN	The average precision ranges from approximately 93% to 99.70%, accuracy from 92% to 99.69%, F1_score from 90% to 99.59%, and recall from 87.42% to 99.47%

Conclusion and Future Work

In this work, we addressed the pressing concern of phishing attacks in the Internet of Things (IoT), focusing specifically on the smart home environment. As IoT devices become increasingly interconnected, the risk of cybercrime, particularly phishing attacks, has grown substantially. These attacks exploit users' trust to gather sensitive information, making it crucial to develop robust security mechanisms to safeguard IoT systems.

Our research explored various models proposed by researchers to counter phishing attempts in the IoT ecosystem. To enhance the cybersecurity of smart homes, we presented our novel methodology centred on Deep Neural Networks, incorporating tokenizer and pad_sequences data pre-processing techniques.

Through a comprehensive literature review, we contextualized the concepts of phishing and smart homes, while surveying existing works on phishing detection in IoT. Our experiment and methodology section detailed the techniques employed to develop our proposed model.

The obtained results demonstrated the effectiveness of our DNN-based model in detecting and preventing phishing attacks in smart homes, showcasing high evaluation metrics. Our model's performance was further compared to other related works, highlighting its competitive advantages.

In summary, this paper makes a valuable contribution to the growing body of research in IoT security and phishing detection, offering valuable insights into protecting IoT devices and systems. As the IoT continues to evolve, our proposed model presents a promising step towards enhancing cybersecurity and safeguarding users' sensitive information in the connected smart home environment.

http://doi.org/10.18080/jtde.v12n1.822

Ultimately, it is crucial to maintain security and ensure the efficiency of the phishing detection model by persistently collecting data and regularly training the model with the most up-to-date information.

References

Abbas, S. G., Vaccari, I., Hussain, F., Zahid, S., Fayyaz, U. U., Shah, G. A., Bakhshi, T., & Cambiaso, E. (2021). Identifying and mitigating phishing attack threats in IoT use cases using a threat modelling approach. *Sensors*, *21*(14). https://doi.org/10.3390/s21144816

Adebowale, M. A., Lwin, K. T., Sánchez, E., & Hossain, M. A. (2019). Intelligent web-phishing detection and protection scheme using integrated features of Images, frames and text. *Expert Systems with Applications*, *115*, 300–313. https://doi.org/10.1016/j.eswa.2018.07.067

APWG. (n.d.). Phishing Activity Trends Reports. Retrieved May 17, 2023, from https://apwg.org/trendsreports/

Atitallah, S. Ben, Driss, M., Boulila, W., & Ghezala, H. Ben. (2020). Leveraging Deep Learning and IoT big data analytics to support the smart cities development: Review and future directions. *Computer Science Review*, *38*, 100303. https://doi.org/10.1016/j.cosrev.2020.100303

Barraclough, P. A., Hossain, M. A., Tahir, M. A., Sexton, G., & Aslam, N. (2013). Intelligent phishing detection and protection scheme for online transactions. *Expert Systems with Applications*, *40*(11), 4697–4706. https://doi.org/10.1016/j.eswa.2013.02.009

Berqia, A., & Nacsimento, G. (2004). A distributed approach for intrusion detection systems. Proceedings. 2004 International Conference on Information and Communication Technologies: From Theory to Applications, 2004, pp. 493–494.

Bouijij, H., & Berqia, A. (2021). Machine Learning Algorithms Evaluation for Phishing URLs Classification. 2021 4th International Symposium on Advanced Electrical and Communication Technologies (ISAECT), pp. 1–5. https://doi.org/10.1109/ISAECT53699.2021.9668489

Bouijij, H., Berqia, A., & Saliah-Hassan, H. (2022). Phishing URL classification using Extra-Tree and DNN. 2022 10th International Symposium on Digital Forensics and Security (ISDFS), pp. 1–6. https://doi.org/10.1109/ISDFS55398.2022.9800795

Cybermalveillance Homepage. (n.d.). Retrieved July 30, 2023, from https://www.cybermalveillance.gouv.fr/tous-nos-contenus/actualites/rapport-activite-2021

Ding, Y., Luktarhan, N., Li, K., & Slamu, W. (2019). A keyword-based combination approach for detecting phishing webpages. *Computers and Security*, *84*, 256–275. https://doi.org/10.1016/j.cose.2019.03.018

Kaggle Homepage. (n.d.). Retrieved July 31, 2023, from https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning

Keras Homepage. (n.d.). Retrieved July 30, 2023, from https://keras.io/guides/

Liu, D. J., Geng, G. G., & Zhang, X. C. (2022a). Multi-scale semantic deep fusion models for phishing website detection. *Expert Systems with Applications*, *209*. https://doi.org/10.1016/j.eswa.2022.118305

Liu, D. J., Geng, G. G., & Zhang, X. C. (2022b). Multi-scale semantic deep fusion models for phishing website detection. *Expert Systems with Applications*, *209*. https://doi.org/10.1016/j.eswa.2022.118305

Mendeley Data Homepage. (n.d.). Retrieved July 31, 2023, from https://data.mendeley.com/datasets/n96ncsr5g4/1

"Pandas Tutorial". (2024). Available at https://www.w3schools.com/python/pandas/default.asp

Rajkumar, S., Sheeba, S. L., Sivakami, R., Prabu, S., & Selvarani, A. (2023). An IoT-Based Deep Learning Approach for Online Fault Detection Against Cyber-Attacks. *SN Computer Science*, *4*(4), 393. https://doi.org/10.1007/s42979-023-01808-y

Sabiri, B., Asri, B. E., & Rhanoui, M. (2022). Mechanism of Overfitting Avoidance Techniques for Training Deep Neural Networks. International Conference on Enterprise Information Systems, ICEIS - Proceedings, 1, pp. 418–427. https://doi.org/10.5220/0011114900003179

Safi, A., & Singh, S. (2023). A systematic literature review on phishing website detection techniques. *Journal of King Saud University - Computer and Information Sciences*, *35*(2), 590–611. https://doi.org/10.1016/j.jksuci.2023.01.004

Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from URLs. *Expert Systems with Applications*, *117*, 345–357. https://doi.org/10.1016/j.eswa.2018.09.029

Salah, H., & Zuhair, H. (2023). Deep learning in phishing mitigation: a uniform resource locator-based predictive model. *International Journal of Electrical and Computer Engineering*, *13*(3), 3227–3243. https://doi.org/10.11591/ijece.v13i3.pp3227-3243

Sánchez-Paniagua, M., Fidalgo, E., Alegre, E., & Alaiz-Rodríguez, R. (2022). Phishing websites detection using a novel multipurpose dataset and web technologies features. *Expert Systems with Applications*, *207*. https://doi.org/10.1016/j.eswa.2022.118010

Scikit-learn Homepage. (n.d.). Retrieved May 17, 2023, from https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

TensorFlow. (n.d.). Create production-grade machine learning models with TensorFlow. Available at https://www.tensorflow.org/

Wei, W., Ke, Q., Nowak, J., Korytkowski, M., Scherer, R., & Woźniak, M. (2020). Accurate and fast URL phishing detector: A convolutional neural network approach. *Computer Networks*, *178*. https://doi.org/10.1016/j.comnet.2020.107275

Yuan, J., Chen, G., Tian, S., & Pei, X. (2021). Malicious URL detection based on a parallel neural joint model. *IEEE Access*, *9*, 9464–9472. https://doi.org/10.1109/ACCESS.2021.3049625