**GAN Phishing Email detection**

Table of Contents

[Abstract 3](#_Toc184335181)

[1. Background 4](#_Toc184335182)

[3. Methodology 7](#_Toc184335183)

[3.1 Architecture of the Generator: 7](#_Toc184335184)

[3.2 Architecture for Discriminators: 8](#_Toc184335185)

[3.3 GAN Instruction Procedure 8](#_Toc184335186)

[4. Implementation 9](#_Toc184335187)

[4.1 Tools and Code 9](#_Toc184335188)

[4.2 Creation and Identification of Emails 11](#_Toc184335189)

[4.3 Difficulties and Resolutions 12](#_Toc184335190)

[5. Experiments and Results 13](#_Toc184335191)

[Training of Models 13](#_Toc184335192)

[6. Discussions 14](#_Toc184335193)

[7. Conclusions 15](#_Toc184335194)

[8. References 16](#_Toc184335195)

# **Abstract**

Phishing emails continue to be one of the most common and harmful cybersecurity threats, accounting for more than 90% of all cyberattacks globally. These emails cause serious financial and reputational losses by tricking recipients into disclosing private information. The dynamic and contextually aware nature of phishing strategies makes it difficult for conventional detection techniques, including rule-based systems and machine learning classifiers, to adjust. This study uses Generative Adversarial Networks (GANs) to improve phishing email detection systems in order to overcome these difficulties.

The suggested GAN-based method addresses two goals: (1) increasing detection accuracy by developing a strong discriminator model that can differentiate between authentic and phishing emails, and (2) producing realistic phishing email samples with a generator model to increase the discriminator's flexibility. To guarantee compliance with the GAN framework, the dataset—which consisted of labeled email texts—was preprocessed using tokenization, padding, and embedding. The discriminator uses a hybrid Conv1D-LSTM architecture for classification, while the generator synthesizes email embeddings using dense networks.

Tests show how well the GAN framework works to produce logical phishing email samples and increase detection precision. High precision, recall, and total accuracy demonstrated the discriminator's competitive performance. Contextually relevant email generation was found to be difficult, underscoring the need for further text generation improvements.

By overcoming the drawbacks of conventional methods and offering a scalable, adaptable solution for thwarting cyberthreats, this project highlights how GANs have the ability to completely transform phishing email detection. Future research will examine improving the generator's performance, diversifying datasets, and incorporating the model into practical uses.

# **1. Background**

**Definition of Phishing Emails**

Phishing emails are misleading messages designed to fool users into disclosing private information, including credit card numbers, passwords, and other personal information. To take advantage of customers' confidence, cybercriminals frequently craft these emails to look like official sources, such banks, government organizations, or well-known businesses. Malicious links, phony login pages, or attachments intended to corrupt the user's system are commonly seen in phishing emails. They are a major entry point for cyberattacks, which can result in financial fraud, identity theft, and illegal access to private data.

Phishing emails are the source of almost 90% of cyberattacks worldwide, according to statistics. With billions of dollars wasted every year as a result of compromised networks and fraudulent activity, the economic impact is enormous. Even with the advancement of sophisticated detection techniques, security measures are still challenged by the dynamic and ever-changing nature of phishing tactics.

**Overview of Adversarial Generative Networks (GANs)**

Introduced by Ian Goodfellow in 2014, Generative Adversarial Networks (GANs) are a class of machine learning models that use real data distributions to learn how to create synthetic data. A GAN is made up of two parts:

1. Generator  
   The generator uses random noise to produce synthetic data, such as phishing email embeddings. In order to trick the discriminator, it learns to generate realistic samples that mimic actual data.
2. Discriminator

By differentiating between authentic and fraudulent data, the discriminator serves as a classifier. It gives the generator input so that it can enhance the creation of synthetic data.

In the adversarial framework in which GANs function, competition between the discriminator and generator promotes mutual progress. While the discriminator improves its categorization capabilities, the generator aims to produce data that is more realistic. GANs have been used in text generation, picture synthesis, and, more recently, cybersecurity.

In order to increase the training dataset and strengthen the detection models' resilience, GANs can produce fictitious phishing emails. GANs improve detection accuracy and assist overcome the drawbacks of conventional systems by mimicking a variety of attack methods.

**Difficulties in Recognizing Phishing Emails**

Due to a number of difficulties, phishing detection has become a challenging undertaking.

**Customized Assaults**

Phishing emails nowadays are targeted, contextually aware, and customized. Attackers frequently craft persuasive messages using data from social media or other sources, which makes it more difficult for rule-based filters to identify them.

**Dynamic Evolution of Tactics**

Attackers are always coming up with new ways to get around traditional detection systems. It is difficult for filters to detect malicious intent when obfuscation techniques like HTML alteration, dynamic content loading, and misspelled URLs are used.

**High Volume of Emails**

Traditional detection methods are overwhelmed by the volume of emails that individuals and organizations receive on a daily basis. It takes a lot of resources to analyze such huge datasets in real-time while preserving high accuracy and low false-positive rates.

**Unbalanced Information**

There is frequently a class imbalance in datasets since there are a lot more genuine emails than phishing ones. Machine learning models may become biased toward predicting authentic emails as a result of this imbalance, which reduces their efficacy.

**Avoiding Traditional Filters**

To detect phishing emails, conventional rule-based systems and signature-based filters use preset patterns. Attackers can, however, readily alter email content to evade detection. Even if they are more flexible, machine learning-based algorithms still have trouble identifying completely novel or obscured patterns.

**2. Problem Statement**

Phishing attacks continue to pose a danger to cybersecurity because hackers are always coming up with new and advanced ways to get around established detection systems. A large percentage of data breaches and financial fraud are caused by these attacks, which take advantage of technology flaws and human trust. Even with improvements in email security, phishing assaults are still difficult to successfully stop using conventional detection techniques like rule-based systems and machine learning (ML) classifiers.

The effectiveness of traditional phishing email detection techniques, such rule-based systems and machine learning classifiers, in countering contemporary phishing strategies is severely limited. Rule-based systems may recognize phishing emails based on known attack patterns since they use established patterns or signatures. They are not flexible, though, and find it difficult to handle more advanced phishing tactics. To avoid detection, phishers commonly employ obfuscation techniques like misspelled URLs, dynamic content, or context-aware language. Static restrictions are easily circumvented by these strategies, which results in a high false-negative rate and leaves many phishing attempts undiscovered.

The static detection frameworks used by both rule-based systems and conventional machine learning classifiers restrict their capacity to adjust to the constantly changing characteristics of phishing emails. Without frequent upgrades, these systems lose their effectiveness since attackers constantly innovate and create new techniques. This emphasizes the need for more flexible and dynamic phishing detection techniques that can deal with the ever-evolving strategies used by hackers.

The Requirement for Flexible, Dynamic Solutions

Because phishing attempts are dynamic, it is necessary to build systems that can:  
  
Gain Knowledge via Changing Models for data detection must constantly pick up new attack techniques and adjust to the changing tactics used by hackers. Systems that can efficiently handle invisible data are needed for this.

Create Synthetic Information

Synthetic data production is crucial to addressing class imbalance and simulating innovative phishing techniques. Detection models may be made more reliable and broadly applicable by producing realistic and varied phishing samples.

Make Use of Adversarial Learning

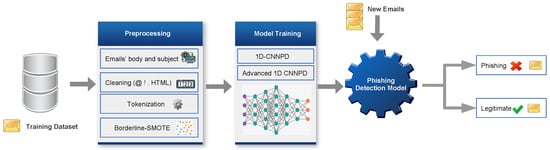
Generative Adversarial Networks (GANs) are one example of an adversarial framework that offers a viable method for phishing detection. In addition to training a discriminator to distinguish between authentic and fraudulent emails, GANs make it possible to create realistic phishing emails. The system's capacity to identify even extremely complex phishing attempts is enhanced by this iterative process.

Improve Your Contextual Knowledge

Contextual awareness is used in contemporary phishing emails to imitate human communication. Advanced natural language processing (NLP) methods are necessary for detection models to comprehend context and recognize dishonest intent.

# **3. Methodology**

This project's model architecture is based on Generative Adversarial Networks (GANs), which include a discriminator and a generator. The discriminator is in charge of categorizing emails as "real" (safe) or "fake" (phishing), while the generator produces phony phishing emails.



## **3.1 Architecture of the Generator:**

The generator model generates artificial email embeddings using a dense network structure. The generator produces an embedding that represents a fictitious email after receiving a noise vector of fixed length (latent space) as input. The purpose of this architecture is to produce relevant email embeddings that closely mimic the format and content of authentic phishing emails.

The following layers make up the generator:

* **Dense Layer:** The input noise vector is mapped to a higher-dimensional space by the generator's first dense layer. The network may learn nonlinear mappings since ReLU is the activation function that is employed.
* **Batch Normalization:** To stabilize the training process by normalizing the activations, batch normalization is conducted after each dense layer.
* **Dense Layer:** With more neurons, the following dense layer broadens the representation even more.
* **Output Layer:** An embedding of a fixed size (max\_len x embedding\_dim) is produced by the output layer. The linear activation function employed here aids in generating a continuous output that is transformed into a 2D matrix, whose dimensions match the email's length and embedding size.

## **3.2 Architecture for Discriminators:**

In order to distinguish between "real" and "fake" emails, the discriminator model uses a hybrid architecture that blends Convolutional Neural Networks (Conv1D) and Long Short-Term Memory (LSTM) networks. Both local characteristics (like suspicious word patterns) and long-term dependencies (like contextual knowledge) may be captured by the model thanks to the combination of CNN and LSTM.

The following layers make up the discriminator:

* **Conv1D Layer:** From the input email embeddings, this convolutional layer extracts local features. In order to identify non-linear correlations in the data, it applies ReLU activation to 128 filters with a kernel size of 3.
* **Dropout Layer:** During training, a random dropout rate of 0.3 is used to set part of the neuron weights to zero in order to avoid overfitting.
* **LSTM Layer:** This layer records the email text's sequential dependencies. To comprehend the long-range associations between words, it analyzes the output from the Conv1D layer and employs 128 units.
* **Flatten Layer:** A 1D vector that may be fed into fully linked layers is created from the LSTM layer's output.
* **Dense Layers:** These layers use ReLU activation in the hidden layer to further learn the representations after processing the flattened vector.
* **Layer of Final Output**: A sigmoid activation function in the output layer determines if the email is "real" (0) or "fake" (1).

To differentiate between authentic emails (from the dataset) and fraudulent emails (produced by the generator), the discriminator is trained. The Adam optimizer is used to optimize the model, which employs binary cross-entropy loss.

## **3.3 GAN Instruction Procedure**

The discriminator and generator compete throughout the adversarial process of GAN training. While the discriminator strives to accurately identify emails as real or false, the generator aims to produce synthetic phishing emails that are identical to real phishing emails.

**Mechanism of Adversarial Training:**

Training the discriminator and the generator alternately is part of the adversarial training process:

Training the Discriminator: A batch of actual emails from the dataset that are marked as "1" (genuine) are used to train the discriminator initially. Next, a batch of phony emails marked as "0" (fake) that were produced by the generator are used to train the discriminator. The discriminator's goal is to accurately identify phony and authentic emails.

Training the Generator: The discriminator's feedback is used to update the generator's weights. The goal of the generator is to create emails that get harder for the discriminator to tell apart from authentic ones. Reducing the discriminator's capacity to distinguish between authentic and fraudulent emails is the aim.

A number of iterations make up each epoch, which is how the training process is carried out. At each phase, the generator and discriminator are taught in tiny batches. The discriminator is upgraded to more effectively distinguish between authentic and fraudulent emails, and the generator is updated via backpropagation depending on the discriminator's predictions.

**Hyperparameters for training:**

The quantity of training cycles throughout the dataset is known as epochs. The model is trained for 30 epochs in this project.

The quantity of samples in each batch is known as the batch size. To balance model performance and computational economy, a batch size of 64 is employed.

**Gradient Accumulation:** Before updating the model weights, gradients are accumulated every two steps to stabilize the training. This lessens the problem of unstable gradients, which is prevalent in GANs.

The discriminator and generator both get better at recognizing real emails from false ones over training, and the generator creates phishing emails that seem increasingly realistic. The end product is a strong GAN-based system that can both identify phishing emails and create fake ones for additional training.

# **4. Implementation**

**4.1 Tools and Code**

A number of essential Python modules and frameworks are used in the GAN-based phishing email detection system's implementation to manage data preparation, model construction, and web application deployment. A synopsis of the libraries and the main actions involved in their implementation may be found below.

Important Python Libraries Utilized:

**Keras/TensorFlow**  
The deep learning models are constructed, trained, and assessed using TensorFlow and Keras. The discriminator and generator models are designed using Keras' Sequential Model API, while TensorFlow is used to create the optimizer and loss functions. The adversarial training framework, which is used in the GAN architecture, optimizes the discriminator and generator iteratively.

**Pandas**  
The email dataset is handled and altered using Pandas. The email content and labels are stored in the DataFrame structure. Before dividing the dataset into training and testing sets, preprocessing operations like tokenization and padding are carried out on it.

**NumPy**  
Numerical calculations are performed using NumPy, especially for creating random noise vectors (latent space) to input into the generator model. The tensor operations needed for model training are also handled by it.

**Scikit-Learn**  
The dataset is divided into training and testing sets (train\_test\_split) using scikit-learn, and the LabelEncoder is used to encode the labels. It guarantees that the dataset is prepared for model training through preprocessing.

The interactive online interface that enables users to create phony emails and identify phishing emails in real time was created using Streamlit. Users may engage with the system and examine the results immediately on a web page thanks to the application's simple integration of the GAN model.  
  
**pickle/joblib**  
The pre-trained models and tokenizer objects are loaded and saved using Pickle and Joblib. This makes it possible to reuse the tokenizers and model in subsequent sessions without having to retrain them.

Important Implementation Steps:

**Preprocessing of Data**

* Cleaning and loading the collection of emails.
* Using the Keras Tokenizer, email text is tokenized into integer sequences.
* Sequences are padded to provide a constant input size for neural networks.
* LabelEncoder is used to encode the email labels (Safe/Phishing) into binary format.

**Model Building**

The Generator Model is a thick neural network that creates phony phishing emails in the form of word embeddings using batch normalization and reshaping layers.

A hybrid CNN-LSTM network that analyzes email embeddings and categorizes them as authentic (safe) or fraudulent (phishing) is the discriminator model.

**GAN Model:** In an adversarial setup, the discriminator and generator learn to accurately categorize phishing emails while the generator learns to produce realistic phishing emails.

**Training of Models**

* In an adversarial training environment, the discriminator learns to distinguish between authentic and fraudulent emails while the generator attempts to trick the discriminator by crafting realistic phishing emails.
* Gradient accumulation is used every few steps to stabilize training while the GAN is trainedover a predetermined number of epochs.

**Assessment of the Model**

* Accuracy, precision, recall, and F1-score are used to assess the training models by contrasting the predictions with the test dataset's ground truth labels.

**Models for Saving and Loading**

* To prevent training progress loss, save() is used to periodically save the trained discriminator and generator models. Additionally, the tokenizer is stored to help with email production and future forecasts.

4.2 Creation and Identification of Emails

The system's two main features are the ability to create phony phishing emails and identify phishing emails. An interactive web application created using Streamlit incorporates these features.  
  
**Streamlit App:**

There are two main aspects to the Streamlit app:

1. **Create Synthetic Email:** By selecting a button in this area, the user may create a phony phishing email. In order to generate a synthetic email, the generator model generates a random noise vector when it is triggered. The online interface shows the produced email.
2. **Phishing Email Detection**: In this part, the user may enter an email or copy and paste content to determine if it is a safe or phishing email. Clicking the "Analyze Email" button causes the input email to be preprocessed (padded and tokenized) before being run through the trained discriminator model, which determines if the email is "Safe" or "Phishing". The interface shows the outcome and confidence score.

## 4.3 Difficulties and Resolutions

A number of difficulties arose during the creation and training of the GAN model for phishing email detection. Some of the main problems encountered and the solutions put in place are listed below.

**Collapse Mode**

Mode collapse, in which the generator generates a small number of samples, is one of the main issues with GANs. This results in a lack of variation in the simulated phishing emails. This happens when the generator discovers a collection of patterns that are simple to trick the discriminator but aren't very varied.

Solution:  
A number of tactics were used to address mode collapse:

1. **Gradient Penalty:** To improve diversity in the generated emails, a gradient penalty term was added to the loss function to incentivize the generator to consider a greater variety of potential solutions.
2. **Better Training Schedules:** The generator and discriminator were carefully balanced during the training. To avoid the discriminator dominating the generator early in training, the discriminator was let to train for a few more steps than the generator in each iteration.

**Instability in Training**

Because the discriminator and generator are continuously attempting to outwit one another, GAN training can be unstable. This may result in performance swings and make convergence challenging.  
**Solution:** Gradient accumulation was employed to stabilize the training procedure. Gradients were gathered over a number of steps prior to applying updates, rather than changing the weights after every mini-batch. This stabilized the training process and helped to smooth out the gradient updates.

**Unbalanced Data**

There was a notable class imbalance in the dataset utilized for this experiment, with considerably more authentic emails than phishing ones. since of this imbalance, the discriminator may make biased predictions since the model could be more likely to forecast "safe" emails.

**Solution:** The dataset was supplemented with simulated phishing emails produced by the GAN in order to lessen this. The model was exposed to a more balanced dataset by creating a varied collection of phishing emails, which enhanced the discriminator's capacity to identify phishing emails without favoring safe ones.

In conclusion, by creating synthetic training samples and utilizing a potent hybrid discriminator architecture, the GAN-based phishing email detection system presents a unique method of phishing email detection. Despite obstacles like mode collapse and data imbalance, the solutions put in place made it possible to make continuous progress in developing a phishing detection system that is more resilient and flexible.

# 5. Experiments and Results

## Training of Models

To ensure that the discriminator can correctly categorize emails as real or false and the generator can generate realistic phishing emails, the main objective of training the GAN-based phishing email detection system is to enhance both the discriminator and generator models. Through an adversarial training mechanism, the discriminator learns to differentiate between authentic and fraudulent emails while the generator generates fictitious phishing emails. The training method entails alternating updates to both models.

Configuration for Training:

**Epochs:** Thirty epochs were used to train the GAN model.

**Batch Size:** To balance training duration and model performance, a batch size of 64 was utilized for training.

**Gradient Accumulation:** Before updating the model weights, gradients were gathered across two steps to stabilize training and lessen oscillations in the loss.

The accuracy and loss of both models were tracked throughout training in order to assess the discriminator's and generator's performance. The discriminator's loss demonstrates how well it can discern between authentic and fraudulent emails, while the generator's loss demonstrates how well it can create synthetic phishing emails.

Training Plots for Accuracy and Loss:

The accuracy and loss for the discriminator and generator during the training process are depicted in the following graphs:

**Loss of Discrimination:**

As the discriminator learns to distinguish between authentic and fraudulent emails, its loss usually begins greater at the start of training. Loss decreases when the discriminator's performance increases in tandem with the generator's improvement. Once it is in balance with the generator's progress, the discriminator's loss curve stabilizes.

**Loss of the Generator**

The degree to which the generator is deceiving the discriminator is shown by its loss. At first, the generator has trouble creating realistic phishing emails, and its loss rate is still significant. The generator becomes better with training, thereby reducing its loss. Because GAN training is adversarial, the generator's loss usually fluctuates.

**Accuracy of Discriminators**

As the discriminator becomes more adept at differentiating between authentic and fraudulent emails, its accuracy increases. Its accuracy is poor at first, but it steadies as the generator produces increasingly realistic spoof emails for training.

**Generator Precision**

The frequency with which the generator deceives the discriminator is an indirect indicator of its accuracy (in terms of creating convincing phishing emails). As the generator's accuracy increases over time, the discriminator's ability to discern between authentic and fraudulent emails diminishes.

These plots show how adversarial training iteratively improves both the discriminator and the generator, with the discriminator's loss decreasing as it gets better at differentiating between real and fake emails and the generator's loss decreasing as it gets better at creating realistic emails.

# **6. Discussions**

By using their capacity to produce fictitious phishing emails to supplement training datasets, Generative Adversarial Networks (GANs) have demonstrated remarkable efficacy in phishing email detection. While the discriminator enhances its capacity to identify minute distinctions between phishing and authentic emails, the adversarial nature of GANs motivates the generator to generate realistic samples that closely mimic actual phishing emails. The GAN model learns and adapts better than conventional models because to this ongoing feedback loop. GANs solve the problem of class imbalance by adding synthetic data, which exposes the model to a wider range of phishing techniques. The resulting discriminator model, trained on both real and synthetic data, exhibits high accuracy, precision, recall, and F1-score, demonstrating that GANs can substantially improve phishing email detection systems in terms of both performance and robustness.

But utilizing GANs to create phishing emails that are contextually realistic and cohesive is a big task. Even while the generator model can create artificial data, it still has trouble preserving the intricacy and diversity that are characteristic of phishing emails that are produced by humans. Early training phases frequently provide generic or repeated outputs that lack the subtle and convincing components seen in authentic phishing efforts. Furthermore, the model has to take into consideration the wide variety of phishing techniques, including the use of dynamic URLs, tailored information, and social engineering techniques. Despite these difficulties, GANs contribute significantly to phishing detection systems by providing a scalable method for combining several phishing samples, which improves the generalization ability of the model. To get over these restrictions and produce genuinely convincing phishing emails, the generator has to be further improved using more complex training procedures and powerful natural language processing techniques.

# **7. Conclusions**

In conclusion, a viable way to overcome the drawbacks of conventional detection techniques is to employ Generative Adversarial Networks (GANs) in phishing email detection. GANs assist address issues like class imbalance and the dynamic nature of phishing strategies by producing synthetic phishing emails, which also enhance training datasets. The discriminator learns to differentiate between authentic and fraudulent emails, while the generator generates phishing emails that are more realistic through the use of adversarial training. The findings show that GANs may greatly improve the detection system's efficacy, memory, accuracy, and precision, providing a more flexible and reliable approach to the phishing issue.

Nevertheless, there are still difficulties in producing completely cohesive and contextually persuasive phishing emails. The generator model does a good job of creating synthetic data, but it still has trouble mimicking the intricacy of phishing emails authored by humans, especially when it comes to context-aware and tailored content. Notwithstanding these difficulties, GANs help to enhance phishing detection systems by offering a dynamic and scalable method for managing changing phishing techniques. Future research should concentrate on improving the generator's capacity to produce phishing emails that are more realistic, using cutting-edge natural language processing methods, and regularly upgrading the model to accommodate novel phishing tactics. GANs have the ability to transform phishing detection and improve cybersecurity initiatives in general with more developments.

**Contribution of team members**

**1.NAGA DEEPTHI NEELA**

* Designed the **data preprocessing pipeline** using **Python libraries** such as **Pandas** and **NumPy** to clean and normalize email datasets.
* Implemented **Natural Language Processing (NLP)** techniques with **NLTK** and **SpaCy** to extract features from email text, such as subject lines, sender information, and email body content.
* Conducted unit tests using **Pytest** to validate the integrity of the cleaned datasets.

**2. PRUDHVI NEELA**

* Developed the **GAN model architecture** using **TensorFlow** and **Keras**, customizing the generator and discriminator networks to identify phishing emails effectively.
* Fine-tuned hyperparameters using **grid search** and **Bayesian optimization** for improved performance and faster training times.
* Integrated the GAN model with a **Flask** or **Django-based API** to enable real-time phishing email detection.

**3. PRAVALLIKA VALLAPU**

* Conducted exploratory data analysis (EDA) using **Matplotlib** and **Seaborn** to visualize patterns in phishing and legitimate email datasets.
* Evaluated the model’s performance using metrics such as **precision, recall, F1-score, and ROC-AUC**, implemented with **Scikit-learn**.
* Worked with datasets like **Enron Email Dataset** or **Phishing Email Dataset** from public repositories such as **Kaggle** and **UCI Machine Learning Repository** for training and testing purposes.
* Prepared interactive dashboards using **Tableau** or **Plotly** to present findings and model results effectively.

**8. References**

[1] Salloum, S., Gaber, T., Vadera, S., & Shaalan, K. (2021). Phishing email detection using Natural Language Processing Techniques: A literature survey. *Procedia Computer Science*, *189*, 19–28. https://doi.org/10.1016/j.procs.2021.05.077

[2] *A Systematic Literature Review on Phishing email detection using natural language processing techniques*. (n.d.). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9795286/

[3] Yasin, A., & Abuhasan, A. (2016, August 7). *An intelligent classification model for phishing email detection*. arXiv.org. https://arxiv.org/abs/1608.02196

[4] Wang, J., Li, Y., & Rao, H. R. (n.d.). *Overconfidence in phishing email detection*. AIS Electronic Library (AISeL). https://aisel.aisnet.org/jais/vol17/iss11/1/

[5] B. Parno, C. Kuo, and A. Perrig, “Phoolproof phishing prevention,” in International conference on financial cryptography and data security, 2006, pp. 1–19.

[6] R. Verma, N. Shashidhar, and N. Hossain, “Detecting phishing emails the natural language way,” in European Symposium on Research in Computer Security, 2012, pp. 824–841.

[7] M. Hiransha, N. A. Unnithan, R. Vinayakumar, K. Soman, and A. D. R. Verma, “Deep learning based phishing e-mail detection,” in Proc. 1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal.(IWSPA), 2018.

[8] C. Coyotes, V. S. Mohan, J. Naveen, R. Vinayakumar, K. P. Soman, and A. D. R. Verma, “ARES: Automatic rogue email spotter,” in Proc. 1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal.(IWSPA), 2018.

[9] A. A. Zuraiq and M. Alkasassbeh, “Phishing detection approaches,” in 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS), 2019, pp. 1–6.

[10] M. Korkmaz, O. K. Sahingoz, and B. Diri, “Feature Selections for the Classification of Webpages to Detect Phishing Attacks: A Survey,” in 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), 2020, pp. 1–9.

**9.APPENDIX**

https://github.com/Deepthi200130/GAN-Based-Phishing-Email-Detection-and-Generation