International Journal of Computer Applications Technology and Research
Volume 12–Issue 02, 32-41, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1202.1005

An Ensemble Learning Model for Predicting Social Engineering Pharming Attacks

Lilian Njuguna
School of Technology
KCA University
Kenya

Gabriel Kamau
School of Computing and Information Technology
Murang'a University of Technology
Kenya

Abstract: Pharming attack has a broad scope as social engineers can masquerade as anyone, particularly during the COVID-19 pandemic from health authorities or even organization executives getting in touch with their personnel. The study aims to develop an ensemble model for predicting social engineering-based pharming attacks from the client-side pharming attack. The target population for the study includes 1781 URLs, which are secondary and readily available on Kaggle having been compiled by Manu Siddhartha. The study focused on identifying URLs that facilitate Pharming attacks, a cybersecurity threat. Malicious URLs miss the protocol segmentation, certificate of authorization, and bait targets into becoming victims of pharming attacks. The research instrument used for data process and building a pharming attack model was CSV and Jupyter notebook. Data was collected from secondary data sources. A model for predictive pharming attacks was built utilizing Logic regression, Random Forest, and gradient boost as the model for boosting algorithms to reduce pharming malware attacks.

Keywords: *Ensemble learning, Pharming attack, Phishing, Social Engineering,*

1. INTRODUCTION

The widespread adoption of internet usage has subsequently led to an increased number of cyber-attack vectors through social engineering [1]. Social engineering attacks are a type of cybercrime where the attacker fools the target through impersonation, pretending to be someone the target knows. Considering cyber security, social engineering accomplishes its malicious goals by captivating human weaknesses. Social engineering is a grim security threat to nations, users, data and its operations and technological infrastructure [2].

Phishing is a type of social engineering whereby social engineers try to deceitfully acquire sensitive information from a target by pretending to be a reliable third party, which entails obtaining access credentials like username, and password [3]. The phishing attack is executed through email spoofing or messaging, and it instructs users to provide personal data on bogus websites. The interface of the bogus website is identical to the genuine website, the only dissimilarity is the URL of the website [4].

Pharming and phishing terms have been used concurrently and these can initiate the capacity for online identity theft. A Pharming attack will redirect the target to the bogus website even though the target is typing the right website address [5]. Pharming encompasses seizing the user's browser settings or running a background process that spontaneously readdresses targets to a bogus site. Most of the time, social engineers aim to access financial data or the authentication of the target credentials, so the redirect prompts when the targets circumnavigate to a financial or banking website [6].

Pharming is a superior kind of phishing attack or DNS poisoning whereby the target is redirected to a bogus website by altering the IP address at the DNS server thus obtaining personal, private, or confidential information or data through technical means [7].This could be an individual or a well-known organization thus gaining access to private data by manipulating the user's email and ads. The highly prevalent pharming aimed websites are online banking and e-commerce websites. As a result of the absence of security administration, desktops are also vulnerable to pharming attacks. An example of pharming is attackers replacing the phone book with a fake one they created [8].

This study aims to build a model to predict pharming malware attacks as a solution, to minimize fraud in cyberspace that is executed through pharming attacks. The develop an Ensemble model for predicting social engineering-based pharming malware attacks. The ensemble is found to be a superior answer to detecting malware pharming attacks. Since it can combine the resemblance in accuracy and several error-detection rate characteristics in picked algorithm[9].

Remarkably, ensemble learning model framework has been utilized globally to detect malwares and improve on prediction accuracy on malware prediction models. Nevertheless, the challenge has occurred in the selection of models that are more suitable for predicting pharming malware on the client side. Thus, a recommendation for further this work to be extended by going into depth of parameters of pharming attacks by finding more suitable features based on URL and website[5].

2. RELATED WORKS

Gajera, Jangid, Mehta, & Mittal, [7] used Artificial Neural Networks ANN for pharming prediction. Elements were picked based on URL parameters and ingested into neural networks for training. The identification amongst genuine websites. The prediction of malicious websites and the addresses were queried to local and global DNS, if they

equally return the same output then the website is genuine otherwise pharming attack was existing.

Ibrahim S. Alfayoumi, [8] presented a client-side approach for predicting pharming malware attacks on the base of the Authorized DNS serve IP Address matching approach. IPS are picked from local and genuine servers and in the subsequent, step their IP addresses are checked. If the IP address is mapped, then it is dispatched as a genuine website. The information collected from the ZONE file is verified and if discrepancies then it is stated that Pharming malware has transpired.

Manhas, Taterh, and Singh, [5] carried a study on the prediction of Pharming attacks on websites using the SVM Classifier. The study entails providing an additional safety Transport Layer Security/Secure Sockets Layer (TLS/SSL) was introduced. By defining a classifier that was able to predict the malicious websites up to great extent. The implementation was set up in MATLAB along with its different modules and libraries for the attribute .

Do Xuan, Nguyen, & Tisenko, [10] study on Malicious URL prediction based on Machine Learning. Machine learning algorithms are applied to classify URLs based on their attributes and behaviour's of URLs. The characters are mined from static and dynamic behaviour's of URLs thus new to the literature. Support vector machine (SVM) and Random Forest (RF) are the two supervised machine learning algorithms used.

Azeez, Oladele and Ologe, [6] conducted a study on Identification of pharming in communication networks using ensemble learning. Analyzed combinations' outcomes and each base learner model's results were compared in order to assess performance. The significance measurements were weighed, and calculated, including Accuracy (Initial Split Assessment), Mean Accuracy (Cross Validation Evaluation), Accuracy, Recall, and Log loss Positive, False Positive, Positive, True F-score, negative, and false negative.

Cohen, Nissim, and Elovici, [11] conducted a study titled Machine learning based solution for the detection of malicious JPEG (images) . In order to distinguish between legitimate and malicious JPEG images, MalJPEG statically extracts 10 straightforward yet discriminative properties from the JPEG file structure. The paper assessed MalJPEG using a real-world sample of 156,818 photos, of which 155,013 (98.85%) are benign and 1,805 (1.15%) are malicious. With an area under the receiver operating characteristic curve (AUC) of 0.997, a true positive rate (TPR) of 0.951, and a very low false positive rate (FPR) of 0.004, the results suggest that MalJPEG, when combined with the LightGBM classifier, exhibits the highest detection capabilities.

A Novel Machine Learning Approach for Malware Detection Malware analysis is the process of finding malware on a system. Static analysis and dynamic analysis make up both halves. A malicious file can be examined using static analysis, and a file can be observed being processed using dynamic analysis. The study presented a methodology for malware analysis that is based on dynamic malware detection and semi-automated malware detection, which is often machine learning. The framework employs the process of classification and displays the quality of experience to preserve efficiency trade-offs. Malware samples demonstrate that a robust detection mechanism was created by the framework.

3. PROPOSED APPROACH

Predictive analytics are techniques that help to make predictions. They analyze current and historical data to answer the question. All predictive analytics are probabilistic. Therefore, they do not indicate what will be the outcome in the future but anticipate what might happen in the future. Predictive analytics doesn't predict one likely future, but rather "multiple futures" based on the outcome actions. The predictive analytical process was utilized which consists of the phases explained briefly below.

Phase 1: Data discovery

Data discovery was achieved at the outset by collecting the data from all of the accessible sources. Understanding the data facilitated in selecting the algorithm. This was achieved by visualizing the data in in Jupyter notebook as shown in Figure 4.2. Understanding the essential information pertaining to the data assisted in making an initial decision on an algorithm.

The size of data some algorithms operate well with big chunks of data than others. The small training datasets, algorithms with low variance and high bias classifiers will run better. Compared to low bias/ high variance classifiers. Thus, for small training dataset, Naïve Bayes will perform well than KNN.

The characteristics of data this indicates how the dataset was formed. The dataset utilized in the study is linear. Thus utilized the logistic regression and random forest model fitted it best, since the data was more complex. The dataset behaviour, the features are sequential thus utilized the random forest.

The dataset from Malicious and Benign Websites | Kaggle set comprises 1781 websites and 21 columns.. These features facilitated predicting the malicious websites that lead to a pharming attack. There are 15 numerical variables, 4 categorical variables, and 2 Date variables. Table 3.1 detailed tabulation of the statistical and classified variables as derived from the Kaggle.

International Journal of Computer Applications Technology and Research
Volume 12–Issue 02, 32-41, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1202.1005

Table 3.1 Data Understanding

Variable Name	Definitions of the variable
'URLs'	The unidentified URL analysed in the study
'URLs_LENGTH'	The URL length, the number of characters.
'NUMBER_SPECIAL_CHARACTERS'	The tally of the unique characters in the URL, for example (/, %, #, &)
'CHARSET'	The categorical variable - character encoding standard or character set
'SERVER'	The operative system of the server, from the packet response (categorical variable)
'CONTENT_LENGTH'	The content size of the HTTP header
'WHOIS_COUNTRY'	The nations we got from the server reaction, explicitly, our content utilized the API of Who is (categorical variable)
'WHOIS_STATEPRO'	The states we got from the server response, specifically, our script used the API of Who is (categorical variable)
'WHOIS_REGDATE'	Server registration date This variable has date values with format (DD/MM/YYY HH:MM)
'WHOIS_UPDATED_DATE'	The last update date from the analysed server
'TCP_CONVERSATION_EXCHANGE'	The number of TCP packets exchanged between the server and our honeypot client
'DIST_REMOTE_TCP_PORT'	The number of the ports detected and different to TCP
'REMOTE_IPS'	Total number of IPs connected to the honeypot
'APP_BYTES'	Number of transferred bytes
'SOURCE_APP_PACKETS'	Packets sent from the honeypot to server
'REMOTE_APP_PACKETS'	Packets received from server
'SOURCE_APP_BYTES'	The source of the app bytes
'REMOTE_APP_BYTES'	The remote app bytes
'APP_PACKETS'	Complete number of IP created while the correspondence between the honeypot and the server
'DNS_QUERY_TIMES'	DNS packets generated during the communication between the honeypot and the server
'TYPE'	Represent the type of web page analysed (1 is for malicious websites and 0 is for benign websites)

Data discovery involved the following steps

Step 1: Data pre-processing

Data pre-processing was performed once the data set was discovered. The collected datasets underwent dataset processing in order to further tailor them to the needs of the study. Pre-processing involved a number of processes, including

- Data cleaning is the task of smoothening filling missing values, noise, and solving discrepancies.
- Data feature selection
- Data Retrieval ingesting the data into the Jupyter notebook.
- Dataset transformation to modifying data so that can be ready for predictive analytics.
- Data selection identifying suitable data sources, analytics libraries/algorithms, as well as relevant variables data.

This data Kaggle data set comprises 1781 websites and 21 columns. These features facilitated predicting the malicious websites that lead to a pharming attack. There are 15 numerical variables, 4 categorical variables, and 2 Date variables. During the data understanding stage that it was detected the "URL" variable has 1781 unique values. This translated to a unique value for each website in the dataset, thus being the unique identifier for each URL.

The data was stored in the local disk, and later it was ingested into the Jupyter notebook. The data processing and analysis was conducted using Python programming language in the Jupyter notebook. The programme was utilized for data processing, data analysis, data mining and ensemble learning to model the predictive model. The data collected was hence visualized in graphs and charts. Highlighting the quantitative analysis with qualitative data was also stated. The statistical data mining and machine learning tools, like per cent, correlation, classification, and regression was utilized to interpret the study findings. The study utilized three ensemble learning method boosting Algorithm.

Step 2 Feature Selection

The main objective of the study was to develop a model to predict pharming malware attacks via malicious links. Pharming is a subset of phishing attacks even though pharming malware executes the DNS poisoning. However, the study core goal was not to explicitly differentiate pharming attack and phishing attack, but rather build a model that will enable prediction of malicious links which can propagate pharming attack. Thus, by extension enable post process malware analysis that can help to identify specific pharming attack from any other pharming attack. Figure 3.2 indicates the features manual select from the dataset.

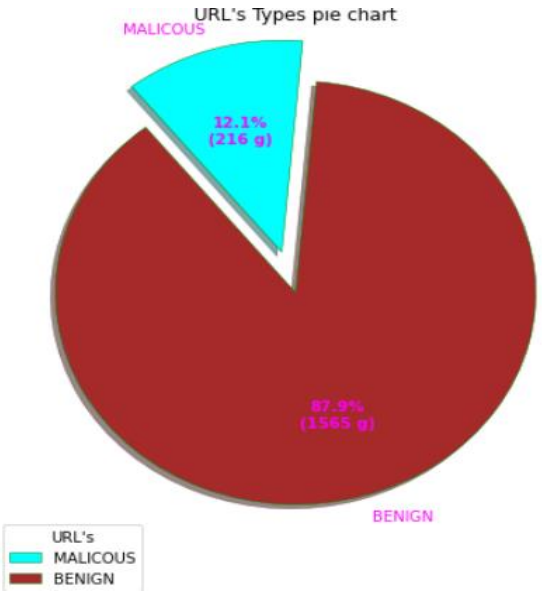


Figure 3. 1 Percentage of Malicious to Benign URLs

The length of the URL - Long URLs could be used to hide the questionable portion from view in the address bar. Even if there is a reliable scientific method for determining whether a website is malicious or benign, there are parameters that must be met. The dataset was used to pick the length of the URL value, which was done by manually comparing the lengths of the longest benign and malicious websites in the dataset.

Step 3 Data Retrieval

The data was then loaded on Jupyter notebook using panda's python library from the local disk where it was initially stored after the download from Kaggle website. The dataset for pre-processing, exploration, and the building of the predictive model. Part 1. Importing the libraries to facilitate dataset loading as indicated below.

```
In [3]: import numpy as np # linear algebra
        import pandas as pd

        from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
        from sklearn.base import TransformerMixin
        from sklearn.pipeline import Pipeline
        import string
        from sklearn import metrics
        import matplotlib.pyplot as plt
        import os
        import seaborn as sns

        import warnings
        warnings.filterwarnings('ignore')
```

```
C:\Users\Injuguna\AppData\Roaming\Python\Python37\site-packages\pandas\compat\_optional.py:138: UserWarning: Pandas requires v
rsion '2.7.0' or newer of 'numexpr' (version '2.6.9' currently installed).
  warnings.warn(msg, UserWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\feature_extraction\image.py:167: DeprecationWarning: `np.int` is a deprecat
d alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is sa
e. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review
our current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  dtype=np.int):
```

Part 2. Loading of data into the Jupyter notebook

```
In [4]: data = pd.read_csv('C:\pydatafiles\Kaggle_Dataset3.csv')
        data.head()
```

Out[4]:

	URL	URL_LENGTH	NUMBER_SPECIAL_CHARACTERS	CHARSET	SERVER	CONTENT_LENGTH	WHOIS_COUNTRY	WHOIS_STATEPRO
0	M0_109	16	7	iso-8859-1	nginx	263.0	None	None
1	B0_2314	16	6	UTF-8	Apache/2.4.10	15087.0	None	None
2	B0_911	16	6	us-ascii	Microsoft-HTTPAPI/2.0	324.0	None	None
3	B0_113	17	6	ISO-8859-1	nginx	162.0	US	AK
4	B0_403	17	6	UTF-8	None	124140.0	US	TX

5 rows × 21 columns

Step 3 Data cleaning

This entails the task of smoothening noise, filling missing values and resolving inconsistencies in a given dataset. Below are the data cleaning process undertaken to prepare the dataset. The column CONTENT_LENGTH has 50% of the value blank. The variable "APP_PACKETS" has duplication of values with "SOURCE_APP_PACKETS". The variable 'WHOIS_STATEPRO' also had values that were either initials or numbers. To understand the data below is a snippet of the statistical summary

```
In [5]: data.describe(include='all')
```

Out[5]:

	URL	URL_LENGTH	NUMBER_SPECIAL_CHARACTERS	CHARSET	SERVER	CONTENT_L
count	1781	1781.000000	1781.000000	1781	1780	966
unique	1781	NaN	NaN	9	239	
top	M0_109	NaN	NaN	UTF-8	Apache	
freq	1	NaN	NaN	676	386	
mean	NaN	56.961258	11.111735	NaN	NaN	11726
std	NaN	27.555586	4.549896	NaN	NaN	36391
min	NaN	16.000000	5.000000	NaN	NaN	0
25%	NaN	39.000000	8.000000	NaN	NaN	324
50%	NaN	49.000000	10.000000	NaN	NaN	1853
75%	NaN	68.000000	13.000000	NaN	NaN	11323
max	NaN	249.000000	43.000000	NaN	NaN	649263

11 rows × 21 columns

On further understanding the data the variable URL is totally unique. Also, as we preview the first 5 lines, we depicted it as an identifier that is mapping key and does not define the URL Hence the conclusion to drop it from the data set. Identifying if there were any missing values and thus clean up the dataset and fill the empty data points. Below is the result after this process.

```
In [15]: # Interpolate our data to get rid of null values
         data = data.interpolate()
         print(data.isnull().sum())
```

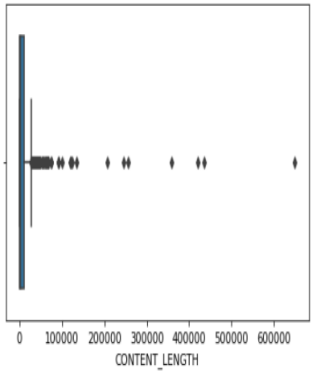
```
URL_LENGTH                 0
NUMBER_SPECIAL_CHARACTERS  0
CHARSET                    0
SERVER                     0
CONTENT_LENGTH             0
WHOIS_COUNTRY              0
WHOIS_STATEPRO             0
WHOIS_REGDATE              0
WHOIS_UPDATED_DATE         0
TCP_CONVERSATION_EXCHANGE  0
DIST_REMOTE_TCP_PORT       0
REMOTE_IPS                 0
APP_BYTES                  0
SOURCE_APP_PACKETS         0
REMOTE_APP_PACKETS         0
SOURCE_APP_BYTES           0
REMOTE_APP_BYTES           0
APP_PACKETS                0
DNS_QUERY_TIMES            0
Type                       0
dtype: int64
```

Step 4 Data integration

The next step was to drop outliers from the column content length using the linear method to facilitate model creation.

```
In [11]: # Handling missing values and outliers for CONTENT_LENGTH feature
         sns.boxplot(data.CONTENT_LENGTH)
         data.CONTENT_LENGTH.describe()
```

```
Out[11]: count       967.000000
         mean      11733.697001
         std       36429.131522
         min           0.000000
         25%         324.000000
         50%        1853.000000
         75%       11324.500000
         max      649263.000000
         Name: CONTENT_LENGTH, dtype: float64
```



```
In [12]: # dropping the outliers

         data = data.drop(data[data['CONTENT_LENGTH']>300000].index.values)
         data.CONTENT_LENGTH = data.CONTENT_LENGTH.interpolate(method ='linear')
         print(data.shape)
         data.isnull().sum()

         (1775, 21)

Out[12]: URL                         0
         URL_LENGTH                  0
         NUMBER_SPECIAL_CHARACTERS   0
         CHARSET                     0
         SERVER                      0
         CONTENT_LENGTH              0
         WHOIS_COUNTRY               0
         WHOIS_STATEPRO              0
         WHOIS_REGDATE               0
         WHOIS_UPDATED_DATE          0
         TCP_CONVERSATION_EXCHANGE   0
         DIST_REMOTE_TCP_PORT        0
         REMOTE_IPS                  0
         APP_BYTES                   0
         SOURCE_APP_PACKETS          0
         REMOTE_APP_PACKETS          0
         SOURCE_APP_BYTES            0
         REMOTE_APP_BYTES            0
         APP_PACKETS                 0
         DNS_QUERY_TIMES             0
         Type                        0
         dtype: int64
```

Step 5 Data Selection

The dataset was split into two datasets consist of the testing sets for evaluating the performance of the assembled model and training set for model assembling. The research implemented the train test procedure which entails having separate data for training and testing the performance of the predictive model. The classification was based on the variable 'TYPE' thus separating it into two different data frames.

```
In [23]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, r
         om_state = 10)
```

Separating training and test data

The dataset was split on a ratio of 70:30 where 70 percent of the data was utilized for training functions while else the 30 percent was utilized for testing. the random state was to ensure that this can be recreated test/training split and reinvent the results.

```
In [24]: X_train.shape, X_test.shape, y_train.shape, y_test.shape

Out[24]: ((1420, 16), (355, 16), (1420,), (355,))
```

Phase 2. Model Development

This phase is the task of building a model that was utilized to predict the pharming malware attack. Data Mining Task which was built and train using the training data set. This entailed defining the model object which refers to specifying the components of the model and their interconnections. Compile the defined model object converting the model into an executable model. Training the model which is performing machine learning to train the model by calling the model. fit () method and passing input attributes (X) sample and output attributes (Y) sample.

Ensemble learning was the selected algorithm utilized to building the pharming malware prediction model. This entails merging the strengths drawn from simpler base models. There are several techniques that form the basis of ensemble learning in this study we will utilize boosting ensemble mechanism with base model Random Forest, Logistic regression, and Gradient boosting. Random forests the method depend on multiple decision trees for training purposes. The predictions from the trees are assembled simultaneously in making a final prediction. This was achieved by using the mean prediction for regression based random forest. This technique of combining multiple trees is the basis for reference for an ensemble technique. Random Forests are extensively used to solve real world machine learning challenges that would require classification or regression-based solutions.

The baseline models were varied, and 10-fold cross-validation was used to determine which baseline models will be used at level 0 of the ensemble learning approach and which ones will have the best performance. The same datasets as the baseline models were used for a variety of machine learning algorithms. The best machine learning classifiers for detecting fraud were found to be Random Forest, Logistic Regression, MLP, and Gradient Boosting classifiers [9].

The study utilizes the random forests is used into working together with random feature selection. Every new training set was drawn, with replacement, from the initial training set. Then a tree was developed on the new training set applying random feature selection. The reason for selecting boosting is it enhance accuracy when random features are utilized. Also boosting can be used to give continuing estimates of the generalization error (PE∗) of the merged ensemble of trees and estimates for the strength and correlation.

The output is numerical in character with the hypothesis of an independently drawn training set from the distribution of random vector Y, X (BREIMAN, 2001).

$E_{xr}(Y\text{-}h(X))^2$

The predictor of a random forest is achieved by attaining the average over k of the trees.

$(h\ (x, O_k))$

The proposed predictive framework of this research encompassed of boosting based ensemble learning algorithms the method inclined towards prediction pharming malware. The first step involved training three ensemble learning algorithms with their default parameters on the training dataset. Three models were created from this specific step by training the three algorithms on our training dataset.

Post-processing task preparing the model for deployment which is visualising the model by calling the plot model () method that creates a plot of the network model. Followed by testing which refers to assessing whether the model has achieved the expected functional requirement such as predicting or describing. Then model was validated by checking what it had achieved the non-functional requirement of accuracy. The model was saved after validation results were satisfactory.

International Journal of Computer Applications Technology and Research
Volume 12–Issue 02, 32-41, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1202.1005

Phase 3. Model Evaluation

Classification techniques have been applied to many applications in several fields of sciences. The training data are utilized for building a classification model to predict the class label for a new dataset. The outputs of classification models could be discrete as in the decision tree classifier. Though, the outputs of learning algorithms need to be evaluated and analysed meticulously and this analysis interpreted correctly, so as to evaluate different learning algorithms. The confusion matrix was used to assess the trained models as a method of solving a classification challenge. An observation's actual value is listed on the rows of a confusion matrix, while the observation's projected values are listed on the columns. A classification issue known as the binary classification has only two viable solutions (Tharwat, 2018).

Below are defined main lingo and metrics associated with a confusion matrix according to (Tharwat, 2018)

- ✓ True Positive: This indicates whether the observations are true. They are located in the matrix's bottom right cell.
- ✓ True Negative: This designates the observations "NO" that are found in the matrix's top-left cell.
- ✓ False Positive / Type I errors in a model are observations that were expected to be "YES" but were really "NO." They are found in the matrix's top right cell.
- ✓ False Negative/Type II Error: These observations were predicted as "NO" but ended up being "YES," according to the model. They are found in the matrix's bottom left cell.

- ✓ Recall: This shows how many of the dataset's actual positive observations we were successful in correctly predicting. Additionally, known as "Sensitivity of a model.

$$Recall = \frac{TruePositive}{TruePositive + FalsePositive}$$

- ✓ Precision: This tells us how many of the observations that we have predicted to be positive are actual positives.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

- ✓ Specificity: This indicates how many of the actual negative observations in our dataset we were able to predict correctly.

$$Specificity = \frac{TrueNegative}{TrueNegative + FalsePositive}$$

- ✓ Accuracy: This indicates how many observations we predicted correctly regardless of whether they are negative or positive.

$$Accuracy = \frac{\text{TruePositive} + \text{TrueNegative}}{\text{TruePositive} + \text{TrueNegative} + \text{FalsePositive} + \text{FalseNegati}}$$

- ✓ The F1-Score evaluates a model's simultaneous recall and precision balance. Since it might be difficult to compare models with high recall and low precision to those with high recall and low precision, it is an important metric. F1 Score was used in the study to assess the trained models.

$$F1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Phase 4: Model Deployment

This phase entailed using the model to predict real produce reports for supporting the decision-making process.

Loading the model - the model structure and weight data was loaded from the saved Jason files. Compile the loaded model was essential before it was used so that predictions made using the model can use the suitable efficient computation from the Keras backend. The model to predict classes of new data cases was used by selecting new data that is not classified and requires new classes to be predicted.

Phase 5: Model Monitoring

This phase entailed validating the deployed model to ensure it has attained non-functional requirements such as accuracy. Using the following

- Select a new data file with input attributes without known classes
- Utilize the deployed model to predict the unknown classes
- Select a file with a list of known classes for the new dataset
- Utilize the list of established classes to validate the performance of the model

Data was sourced from the online data science platform Kaggle. The downloaded URL file was stored on the local disk. The file is retrieved later into the Jupyter notebook platform using various python libraries. The URLs was analysed using the Jupyter notebook, first by pre-processing the ingested data and performing some feature extraction and labelling. The data was visualized using the necessary visualization libraries. The building of a model to predict pharming malware attacks, the model was evaluated, and results tabulated.

4. RESULTS AND DISCUSSION

Factors that lead to pharming malware attack

The relationship between the variables that lead to pharming malware attack was visualized as visualised in Figure 4.1 According to this view, as URL length increases, the likelihood that a website is harmful decreases until the URL length significantly increases. The likelihood that the webpage is dangerous then rises once more.

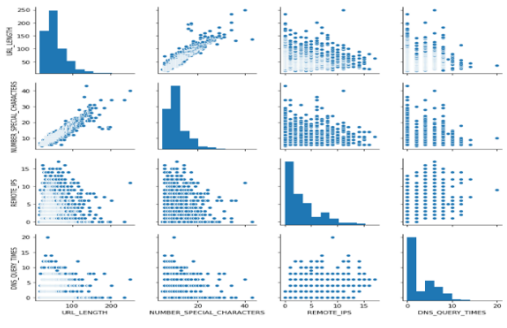
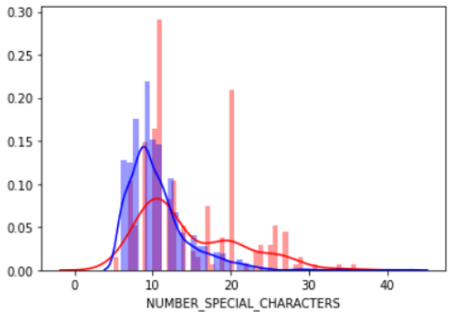


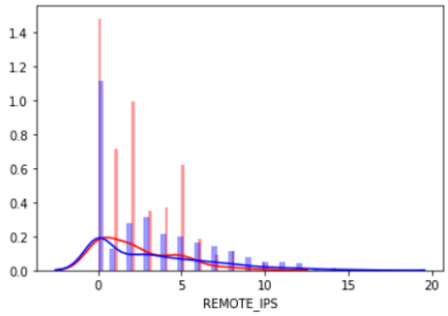
Figure 4.1 Variable Visualization

International Journal of Computer Applications Technology and Research
Volume 12–Issue 02, 32-41, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1202.1005

The attributes demonstrated a meaningful link with the target variable, according to the visualizations we performed in the Exploratory data analysis portion. how many special characters are on a URL, the more special characters, the more harmful the website is. The red bars represent the malicious websites, there are some positive peculiar spikes.



The REMOTE_IPS implies that 'this variable has the total number of IPs connected to the honeypot'. Thus malicious websites have a somewhat lesser grouping of remote IPs connected than benign websites.



The average URL_LENGTH of benign url denoted with 1 is higher than of malicious URLs denoted by O. The average CONTENT_LENGTH of benign URL at 17432 is higher than of malicious URLs at 12954. The shorter the URL length the high chances of it being malicious.

Pharming malware predictive model by use of ensemble learning

Logic Regression

A machine learning algorithm called logic regression is used to train classifiers. It is essentially a linear regression model with the logistic/sigmoid function on top. The following is how it is modelled mathematically:

$z = wx + b$

$y = sigmoid(z)$

$$sigmoid(z) = \frac{1}{1 + e^{-t}}$$

This model assumes that all predictors are linearly associated with the log probabilities of the result, which in this case includes evidence of malice. Convergence problems for logistic regression were caused by the predictor variables' multicollinearity. Lasso regression, which only chooses one feature for highly correlated data and minimizes it to zero coefficient, was used to address the convergence concerns. Figure 4.2 shows the Logic Regression ROC Curve

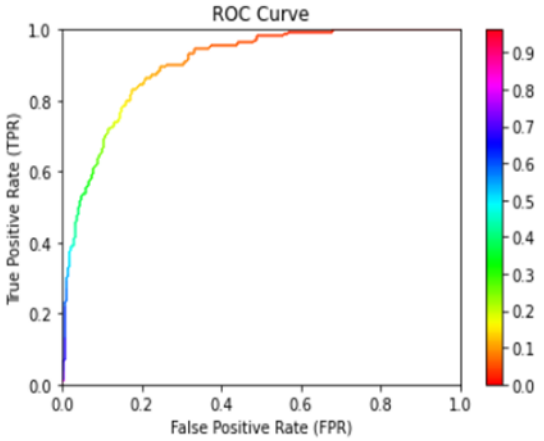


Figure 4. 2 Logic Regression ROC Curve

The Logistic Regression model has a recall of 98 percent overall, an AUC of 0.985, and an accuracy of 91 percent after being trained on the entire train dataset containing labelled malicious and benign traffic. The high recall demonstrates that the model could reliably categorize the majority of malicious traffic. A requirement for the prediction of pharming malware is the strong recall.

RANDOM FOREST

A combination of matplotlib and seaborn was utilized to offer customized themes and give additional plot types. Matplotlib is hence a superset of seaborn make the two important for visualization.

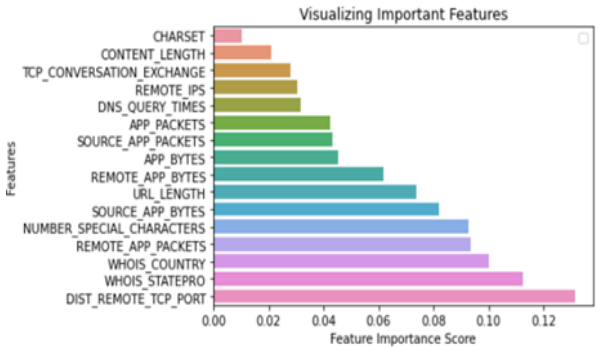


Figure 4.5 Visualizing Important Features

The study generated the model training set features perform predictions on the selected test set features and compare actual and predicted values. The modelling leveraged on

International Journal of Computer Applications Technology and Research
Volume 12–Issue 02, 32-41, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1202.1005

obtaining the best combination of hyperparameters tunning translating to improved performance. This process showed improved performance of Random Forest from training accuracy at 92% to 99% the testing accuracy at 91% to 94%.

Gradient Boosting

Each predictor aims on improving on its predecessor by reduction of errors, by fitting the new predictor to the residual errors created by the preceding predictor. The log odds conversion formula.

$$e * \log(odds) / (1 + e * \log(odds))$$

The modelling leveraged on obtaining the best combination of hyper parameters tuning translating to improved performance. This process showed improved performance of Random Forest from training accuracy at 97% to 99% the testing accuracy at 93% to 96%. The gradient boosting model had a 91 percent accuracy rate and a 98 percent total recall after being trained on the entire train dataset containing labelled malicious and benign traffic. An AUC of 0.985 was achieved for the area under the curve.

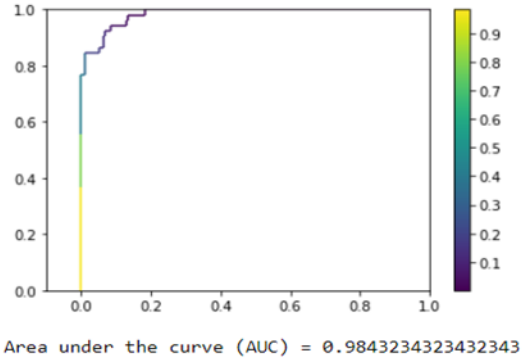


Table 4. 1 Testing Algorithm

Testing Algorithm	**Precision**	**Recall**	**F1-Score**	**Accuracy**
Logic regression	89%	90%	89%	90%
Random Forest	92%	91%	90%	91%
Gradient Boosting	90%	90%	90%	94%

The three models' training set performances were compared. shows that all models have excellent logical accuracy, but the Random Forest model had the highest final score. With the exception of logic. After the hyperparameter tuning the AUC was at 0.984

Below is the comparison of the outcome of three ensemble learning techniques.

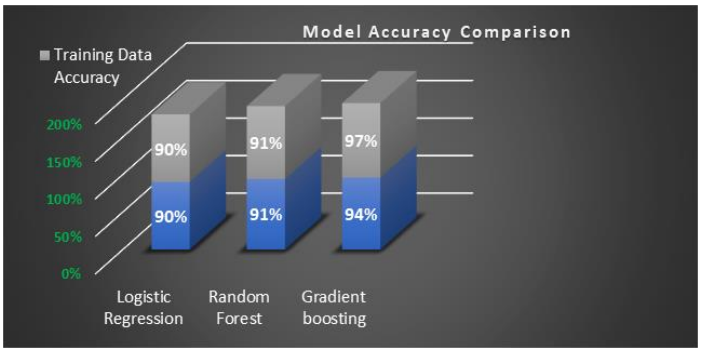


Figure 4. 7 Accuracy Bar graph

The Model Comparison

The three models' test set performances were compared. shows that all models have good rational accuracy, but the gradient boosting model has the greatest final score. With the exception of logic regression, all models have good scores when ranked according to the F1- score.

Training Algorithm	Precision	Recall	F1-Score	Accuracy
Logic regression	90%	91%	90%	90%
Random Forest	90%	91%	96%	91%
Gradient Boosting	90%	90%	90%	97%

Table 4. 2 Training Algorithm

The finding is that, among the trained models, the Gradient Boosting model had the highest accuracy. Given that logistic regression is a linear model and is extremely sensitive to the distributions of predictor variables, the following arguments support the study's findings. It is possible that there exist non-linear relationships between the predictors and the objective. (Kenneth, 2021). Random forests are ensemble methods that have demonstrated incredibly resilient performance in a wide range of classification issues. Hyperparameters were used to boost performance, which also increased accuracy.

Test and validate the model for predicting pharming malware

To facilitate the validation and evaluation of the predictive model built. The study utilized the performance of recommended methodology confusion matrix. This is a table that gives the performance of the classifier on the basis of some parameters on test data containing 1781 URLs. It indicates how the classification model gets confused as their make predictions. The step to follow

International Journal of Computer Applications Technology and Research
Volume 12–Issue 02, 32-41, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1202.1005

when using a confusion matrix is first the dataset validation or test with probable outcome results. Next Predict each row in the test dataset in this study the URLs dataset.

The expected outcomes and expected predictions provide the number of precise predictions for each class and the number of imprecise predictions for each class, requested by the class that was predicted. The values are structured in matrix comprising of Predicted class and Actual class, as shown below assigned to following terms.

True Positive (TP) = 303

False Negative (FN) = 14

False Positive (FN) = 0

True Negative (TN) = 38

The following confusion matrix derived by Gradient boost model shows values of both predicted and actual class.

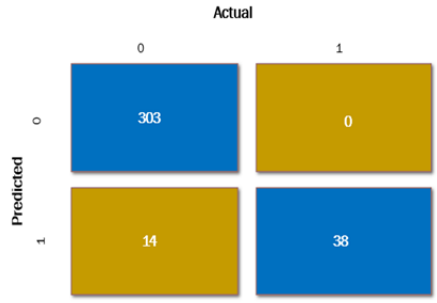


Figure 4. 9 Confusion Matrix

Performance Evaluation

Below are defined main lingo and metrics associated with a confusion matrix according to (Tharwat, 2018).

$$Recall = \frac{TruePositive}{TruePositive + FalsePositive} = 95\%$$

✓ Precision: This reveals the proportion of observations that are actually positive that we projected to be positive.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} = 100\%$$

✓ Specificity: This shows what percentage of our predicted positive observations are actually positive.

$$Specificity = \frac{TrueNegative}{TrueNegative + FalsePositive} = 100\%$$

✓ Accuracy: This shows how many observations, whether they are favourable or negative, that we accurately predicted.

$$Accuracy = \frac{\text{TruePositive} + \text{TrueNegative}}{\text{TruePositive} + \text{TrueNegative} + \text{FalsePositive} + \text{FalseNegative}} = 96\%$$

✓ F1-Score is a measures the balance amongst recall and precision of a model simultaneously. It is a significant measure as it can be challenging to compare models with high recall and low precision to models with low recall and high precision. The study utilized F1 Score to evaluate the models trained.

$$F1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} = 96\%$$

On the test set, the study's highest F1-score was equivalent to 0.96. As a result, the ensemble is effective and trustworthy and can anticipate Pharming attempts.

Variables are essential in predicting pharming malware, the study utilized three ensemble models. In all three model it was observed several variables that were considered essential. Thus, the variable was well-thought-out to be vital by each model, the REMOTE_IPS. This is consistent with the existing literature which show that the IP address check is a vital pointer of to check the legitimacy of a visited website. Hence, it's essential to note the ultimate ensemble learning model using the Gradient Boost technique had the best performance highlighting the necessity to adopt the method in pharming malware prediction on the client side.

5. CONCLUSIONS AND FUTURE RESEARCH

There have been rigorous efforts to predict and hence eradicate the pharming malware attacks, a critical contributor to social engineering in the cyber security threats globally. As an effort to contribute to the cyber security threats, an objective determination of URL features to before building a predictive model for pharming attack. These significant features were utilized as variables to the base model logic regression random forest and gradient boost for the modelling using the Boosting algorithm. However, because typical network usage is so unpredictable, more reliable models will always be essential for keeping track of different networks. A dataset with a distribution of harmful and benign traffic was used by the model. Previous studies have utilized single classifier in predicting the pharming attacks that various users can utilize in malware detection process.

The comparison and combination was undertaken for logic regression, Random Forest and Gradient boost model features. The contribution to computing is in the combination the several models utilizing the URLS features as variables. Thus, the results generated by models providing an insightful way of assembling more details that may not have been captured by initial modelling. This has been helpful in improving the model performance even further.

Future testing of the model will be required on actual network traffic, where the proportion of malicious and benign traffic is far from balanced. The imbalance class may prove to be difficult for conventional ensemble learning methods, requiring the development of unique loss functions. By applying sample strategies to address the class imbalance and more complicated models, the study's findings can be further improved. The limitation of the study was identifying URLs that facilitate pharming attacks and any other malware attack thus a need for further work. Further work can be extended to focus on differentiating pharming attacks from other phishing attacks.

[1] I. Mukhopadhyay, "Cyber threats landscape overview under the new normal," in *ICT Analysis and Applications*, Springer, 2022, pp. 729–736.

[2] R. M. Gupta, P. Mathur, and R. Nanda, "Cyber Security Threat, it's Risks and Susceptibilities, in the Global Perspective," in *Rising Threats in Expert Applications and Solutions*, Springer, 2022, pp. 607–613.

[3] B. B. Gupta, N. A. G. Arachchilage, and K. E. Psannis, "Defending against phishing attacks: taxonomy of methods, current issues and future directions," *Telecommun. Syst.*, vol. 67, no. 2, pp. 247–267, 2018.

[4] L. Barlow, G. Bendiab, S. Shiaeles, and N. Savage, "A novel approach to detect phishing attacks using binary visualisation and machine learning," in *2020 IEEE World Congress on Services (SERVICES)*, 2020, pp. 177–182.

[5] S. Manhas, S. Taterh, and D. Singh, "Detection Of Pharming Attack On Websites Using Svm Classifier," *Int. J. Sci. Technol. Res.*, vol. 8, p. 11, 2019.

[6] N. A. Azeez, S. S. Oladele, and O. Ologe, "Identification of pharming in communication networks using ensemble learning," *Niger. J. Technol. Dev.*, vol. 19, no. 2, pp. 172–180, 2022.

[7] K. Gajera, M. Jangid, P. Mehta, and J. Mittal, "A novel approach to detect phishing attack using artificial neural networks combined with pharming detection," in *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2019, pp. 196–200.

[8] I. S. Ibrahim S. Alfayoumi, "Client – Side Pharming Attacks Detection using Authoritative Domain Name Servers," *Int. J. Comput. Appl.*, vol. Volume 113, no. 10, p. (0975 – 8887), 2015.

[9] R. Soleymanzadeh, M. Aljasim, M. W. Qadeer, and R. Kashef, "Cyberattack and Fraud Detection Using Ensemble Stacking," *AI*, vol. 3, no. 1, pp. 22–36, 2022.

[10] C. Do Xuan, H. D. Nguyen, and V. N. Tisenko, "Malicious URL detection based on machine learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 1, 2020.

[11] A. Cohen, N. Nissim, and Y. Elovici, "MalJPEG: Machine learning based solution for the detection of malicious JPEG images," *IEEE Access*, vol. 8, pp. 19997–20011, 2020.