MSc. Computer Engineering, Cybersecurity and Artificial Intelligence

Phishing URL classification

Prepared by

Mohamed NJAH

Table of contents

I. Introduction

Phishing attacks have emerged as one of the most significant cybersecurity threats in recent years. These attacks rely on deceptive URLs designed to mimic legitimate websites, tricking users into divulging sensitive information. Given the increasing sophistication and volume of phishing attempts, there is a growing need for automated detection methods that can analyze and classify suspicious URLs with high accuracy.

The exponential growth of URL phishing attacks has become increasingly evident in recent years. We collected the last fifteen years' phishing URLs and email data from the Anti-Phishing Working Group (APWG) (Anon, 2023c) and plotted a graph in Fig. 1. The graph shows that the outbreak of the COVID-19 pandemic has led to an immense increase in phishing URL attacks. Cybercriminals exploit the fear, heightened emotions, uncertainty, medical urgency, increased online activity, and information-seeking behavior of individuals during times of crisis. Fig. 1 shows that the growth of phishing URL attacks is at an alarming rate after the outbreak of the COVID-19 pandemic.

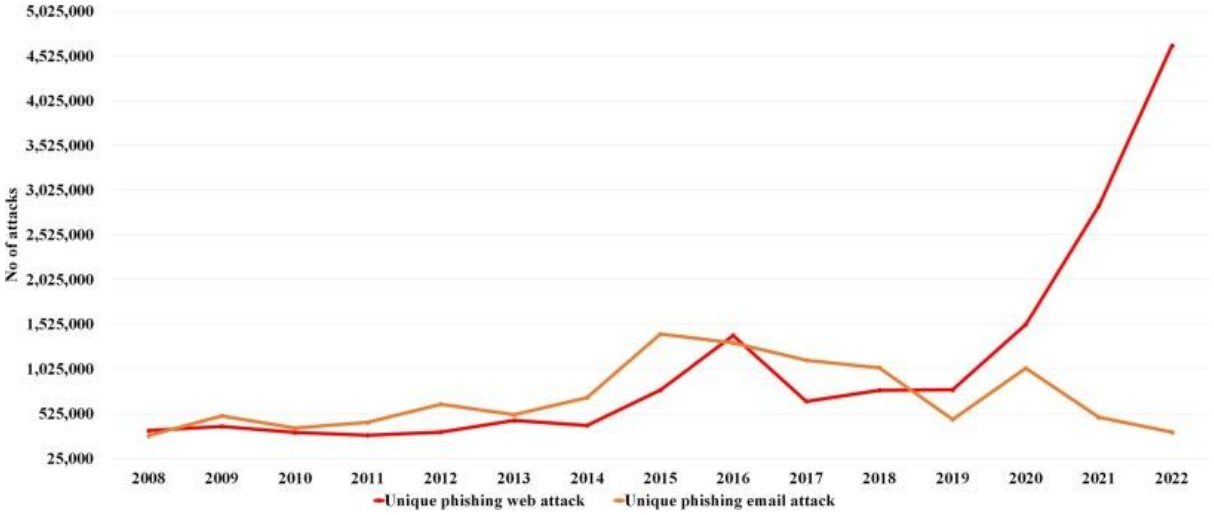

Figure 1 Increase in phishing URL attacks post pandemic.[1]

Self-Organizing Maps (SOM) offer a promising unsupervised approach for detecting anomalies in high-dimensional datasets. SOMs create low-dimensional (usually 2D) representations of complex input data while preserving the topological structure. This feature makes them ideal for clustering and visualizing patterns in datasets such as phishing URLs. In this report, we explore the application of SOMs to the UCI Phishing URL Dataset, providing insights into the data through visualization and comprehensive evaluation metrics.

II. Motivation and state of the art

[1] PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning - Arvind Prasad, Shalini Chandra

The motivation behind this study stems from the rapid increase in phishing attacks and the corresponding need for robust, scalable detection techniques. Traditional signature-based and rule-based methods have struggled to keep pace with evolving phishing strategies. Machine learning offers an adaptive alternative, with numerous approaches being explored in recent literature.

Recent research in phishing detection has leveraged both supervised and unsupervised learning. Supervised methods—including Support Vector Machines (SVM), decision trees, and neural networks—require extensive labeled datasets and may struggle with novel, previously unseen attack patterns.

Furthermore, several studies have emphasized the importance of feature extraction and data preprocessing in achieving high classification accuracy. In our approach, we preprocess the UCI dataset by removing non-numeric fields, normalizing the data, and addressing feature redundancies.

III. Dataset

The UCI Phishing URL Dataset[2] is a comprehensive collection designed for studying the characteristics of phishing websites. The dataset includes a variety of features extracted from URLs—such as URL length, special character counts, domain information, and obfuscation metrics—that are indicative of phishing behavior.

[2] Dataset Link

FILENAME	URL	URLLength	Domain	DomainLength	IsDomainIP	TLD	URLSimilarityIndex	CharContinuationRate	TLDLegitimateProb	URLCharProb	TLDLength	NoOfSubDomain	HasObfus
521848.txt	https://ww	31	www.southbankmosaics.com	24	0	com	100	1	0.5229071	0.061933179	3	1	0
31372.txt	https://ww	23	www.uni-mainz.de	16	0	de	100	0.666666667	0.0326503	0.050207214	2	1	0
597387.txt	https://ww	29	www.voicefmradio.co.uk	22	0	uk	100	0.866666667	0.028555	0.06412872	2	2	0
554095.txt	https://ww	26	www.sfnmjournal.com	19	0	com	100	1	0.5229071	0.057605756	3	1	0
151578.txt	https://ww	33	www.rewildingargentina.org	26	0	org	100	1	0.0799628	0.059441389	3	1	0
23107.txt	https://ww	30	www.globalreporting.org	23	0	org	100	1	0.0799628	0.060614474	3	1	0
23034.txt	https://ww	25	www.saffronart.com	18	0	com	100	1	0.5229071	0.063549404	3	1	0
696732.txt	https://ww	25	www.nerdscandy.com	18	0	com	100	1	0.5229071	0.0604856	3	1	0
739255.txt	https://ww	29	www.hyderabadonline.in	22	0	in	100	1	0.0050842	0.056980442	2	1	0
14486.txt	https://ww	18	www.aap.org	11	0	org	100	1	0.0799628	0.070497453	3	1	0
167350.txt	https://ww	33	www.religionenlibertad.com	26	0	com	100	1	0.5229071	0.063946492	3	1	0
mw42508.txt	http://wwv	22	www.teramill.com	16	0	com	82.6446281	1	0.5229071	0.06741797	3	1	0
515489.txt	https://ww	27	www.socialpolicy.org	20	0	org	100	1	0.0799628	0.064491134	3	1	0
858208.txt	https://ww	20	www.aoh61.com	13	0	com	100	1	0.5229071	0.055131169	3	1	0
712305.txt	https://ww	26	www.bulgariaski.com	19	0	com	100	1	0.5229071	0.055714659	3	1	0
252332.txt	https://ww	24	www.brightika.com	17	0	com	100	1	0.5229071	0.053754992	3	1	0

Figure 2 UCI Phishing URL Dataset

Key characteristics of the dataset include:

- **Feature Diversity**: The dataset comprises multiple numerical features, including URL similarity indices, character continuation rates, and various statistical measures.
- **Labeling**: Each record in the dataset is annotated with a label that identifies whether the URL is phishing or legitimate.
- **Balance**: the dataset has 100945 phishing URLs and 134850 legitimate URLs which means that it is balanced.

The dataset's rich feature set allows for thorough analysis and facilitates the training of a SOM model that can capture the subtle differences between phishing and non-phishing URLs.

IV. Methodology

Our methodology comprises several stages, from data preprocessing to model training and evaluation. The following steps outline our approach:

1. Data Preprocessing

- **Data Cleaning:** We begin by importing the dataset and removing non-numeric columns (such as URL, domain, and title) that do not directly contribute to numerical analysis.

```
% Specify the columns that are non-numeric (string features) and remove them
stringColumns = {'FILENAME', 'URL', 'Domain', 'TLD', 'Title'};
data(:, stringColumns) = [];
```

Figure 3 Removal of non-numeric columns

- **Feature Reduction:** Redundant features are eliminated. For instance, DomainTitleMatchScore is removed due to its high correlation with URLMatchScore.

```
% Remove DomtainTitleMatch Score because it highly correlates to URLMatchScore
features.DomainTitleMatchScore = [];
```

Figure 4 removal of DomainTitleMatch

- **Splitting:** The dataset is split into training (80%) and testing (20%) sets to prevent data leakage.

```
% ----------------------
% 1. Split the Data
% ----------------------
cv = cvpartition(size(numericFeatures,1), 'HoldOut', 0.2);
trainIdx = training(cv);
testIdx = test(cv);

trainFeatures = numericFeatures(trainIdx, :);
testFeatures  = numericFeatures(testIdx, :);
trainLabels   = numericLabels(trainIdx, :);
testLabels    = numericLabels(testIdx, :);
```

Figure 5 Dataset split

- **Normalization:** Training features are normalized using z-score normalization, the same parameters are applied to the test set.

```
% ----------------------
% 2. Normalize the Data
% ----------------------
[trainFeaturesNorm, mu, sigma] = zscore(trainFeatures);
testFeaturesNorm = (testFeatures - mu) ./ sigma;
```

Figure 6 Data normalization

2. SOM Training

- **Self-Organizing Maps:** A self-organizing map (SOM) or self-organizing feature map (SOFM) is an unsupervised machine learning technique used to produce a low-dimensional (typically two-dimensional) representation of a higher-dimensional data set while preserving the topological structure of the data.

  The following figure shows the learning steps:

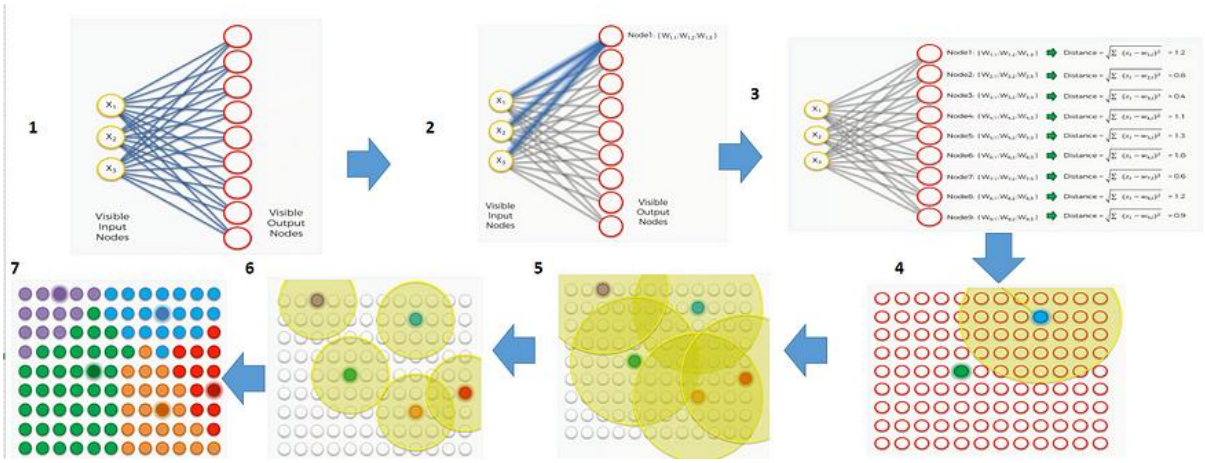

Figure 7 SOM Steps[3]

- **SOM Configuration:** A 3×3 SOM grid is selected, providing 9 neurons that map the high-dimensional data into a two-dimensional space, the number of epochs for the learning phase is 200.

[3] A Beginner's Guide to Self Organizing Map (SOM) - Link

- **Training Process:** The SOM is trained using the normalized training data. During training, each input vector is mapped to its Best Matching Unit (BMU), and weight updates are performed based on a neighborhood function.

```
%% ----------------------
% 3. Train the SOM
% ----------------------
gridSize = [3 3];   % Define a 3x3 SOM grid (9 neurons)
net = selforgmap(gridSize);
net = train(net, trainFeaturesNorm');
```

Figure 8 SOM training script

3. Neuron Labeling and Testing

- **Label Assignment:** For each neuron, a label is assigned by majority voting from the training samples that are mapped to that neuron.

```
%% ----------------------
% 4. Assign Neuron Labels (Training Phase)
% ----------------------
% Get the BMU indices for each training sample
trainOutput = net(trainFeaturesNorm');
[~, trainBMU] = max(trainOutput, [], 1);
trainBMU = trainBMU';

% Determine the number of neurons (should be gridSize(1)*gridSize(2))
numNeurons = prod(gridSize);
neuronLabels = -ones(numNeurons, 1);  % initialize with -1 for neurons with no samples

% For each neuron, assign the label based on majority vote of training samples mapping to it
for i = 1:numNeurons
    idx = find(trainBMU == i);
    if ~isempty(idx)
        neuronLabels(i) = mode(trainLabels(idx));
    end
end
```

Figure 9 Neuron labels assignment

- **Testing Phase:** Test samples are then fed through the trained SOM. The BMU for each test sample is determined, and the neuron's label is used as the prediction.

```
% ------------------------
% 5. Testing Phase: Predict Labels for Test Samples
% ------------------------
testOutput = net(testFeaturesNorm');
[~, testBMU] = max(testOutput, [], 1);
testBMU = testBMU';

% Assign predicted labels based on the neuron's assigned label
predictedTestLabels = zeros(size(testBMU));
for i = 1:length(testBMU)
    predictedTestLabels(i) = neuronLabels(testBMU(i));
end
```

Figure 10 Label prediction for test samples

4. Evaluation Metrics

- **Accuracy:** The overall classification accuracy is calculated.

```
% Compute classification accuracy
accuracy = sum(predictedTestLabels == testLabels) / length(testLabels);
fprintf('Test Accuracy: %.2f%%\n', accuracy * 100);
```

Figure 11 Computing accuracy

- **Confusion Matrix:** A confusion matrix is generated to illustrate the model's performance.

```
% Create and display a confusion matrix
figure;
confMat = confusionmat(testLabels, predictedTestLabels);
confusionchart(confMat);
title('Confusion Matrix for SOM Classification');
```

Figure 12 Creating the confusion matrix

- **Additional Metrics:** Per-class precision, recall, F1 score, and specificity are computed, along with macro-averaged metrics. These metrics provide a detailed understanding of the model's performance across different classes.

The methodology leverages the strengths of SOM for both visualization and clustering, while comprehensive evaluation ensures that the model's classification ability is critically examined.

V. Results

The results of our study are presented in several parts:

SOM Visualization

- **Hits Map:** A hits map (plotsomhits) shows the number of training samples mapped to each neuron, indicating data density and cluster activity.

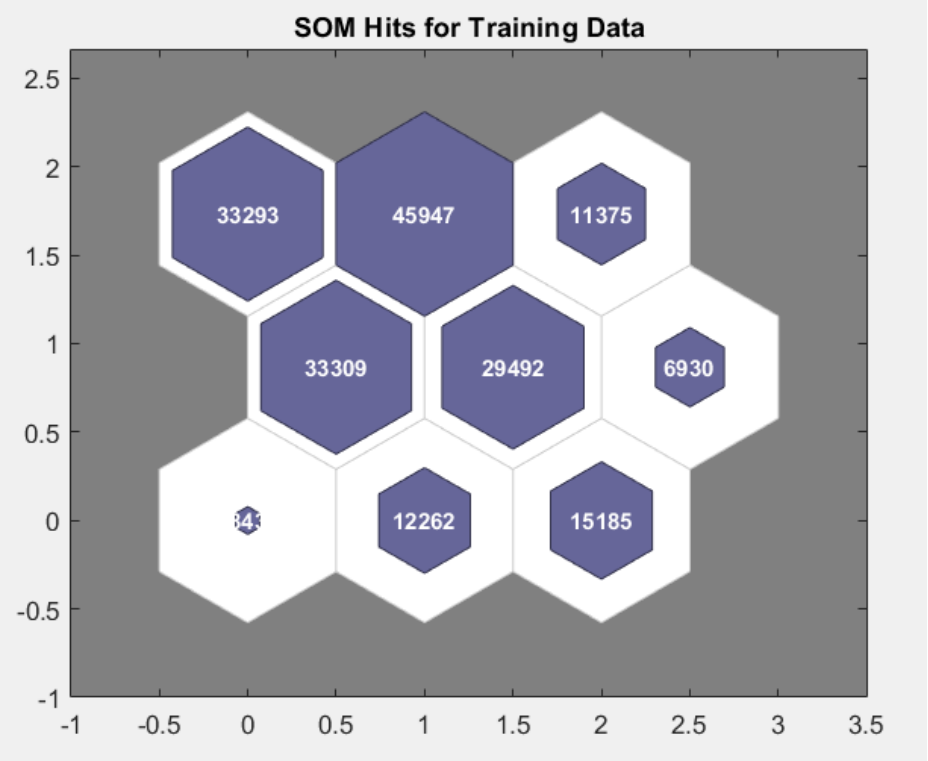

Figure 13 SOM hits map

- **Weight Planes:** The weight planes (plotsomplanes) provide insight into how individual features are distributed across the SOM grid.

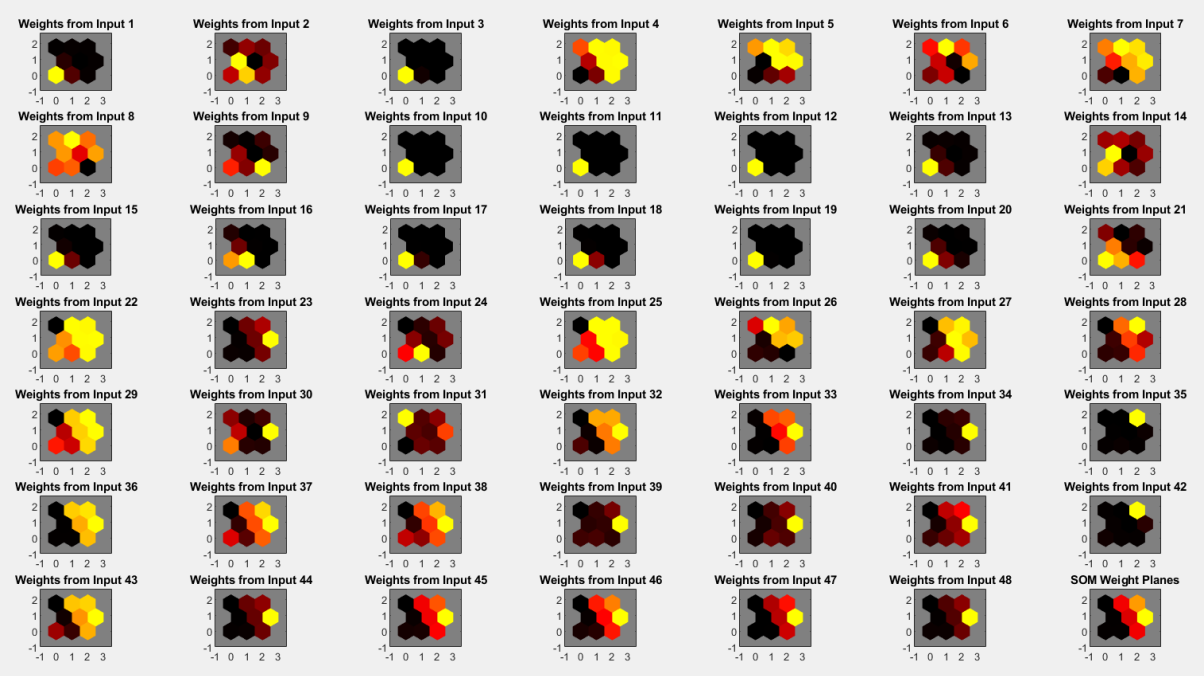

Figure 14 SOM Weights

Classification Performance

- **Overall Accuracy:** Our SOM classifier achieved an accuracy of approximately *99.11%* on the test set.
- **Confusion Matrix:** The confusion matrix reveals the distribution of correct and incorrect classifications, showing how well the model distinguishes between phishing and non-phishing URLs.

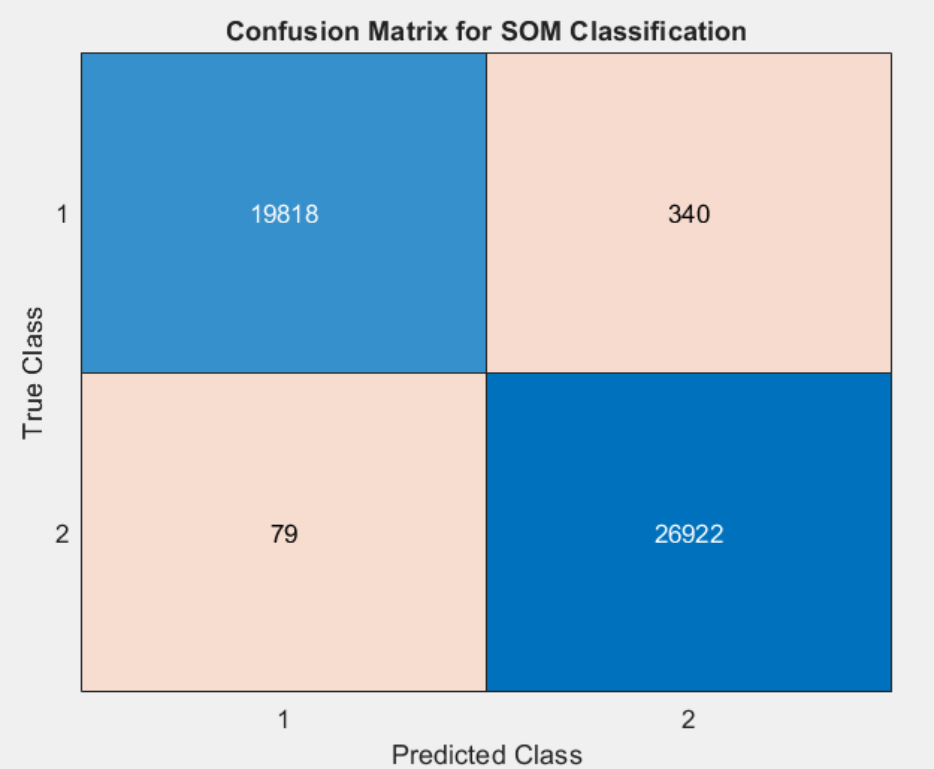

Figure 15 Confusion matrix

Evaluation Metrics

For each class, the following metrics were calculated:

- **Precision:** The ratio of true positives to the sum of true and false positives.
- **Recall:** The ratio of true positives to the sum of true positives and false negatives.
- **F1 Score:** The harmonic mean of precision and recall.
- **Specificity:** The proportion of true negatives correctly identified.

Macro-averaged metrics, calculated as the mean of the per-class metrics, provide an overall performance indicator.

```
Class 1: Precision=99.60%, Recall=98.31%, F1 Score=98.95%, Specificity=99.71%, Support=20158
Class 2: Precision=98.75%, Recall=99.71%, F1 Score=99.23%, Specificity=98.31%, Support=27001
Macro-Averaged Metrics: Precision=99.18%, Recall=99.01%, F1 Score=99.09%, Specificity=99.01%
```

Figure 16 Evaluation metrics

VI. Conclusion

In conclusion, this study demonstrates the applicability of Self-Organizing Maps in classifying phishing URLs. The unsupervised nature of SOM enables effective clustering and visualization of high-dimensional URL features, thereby highlighting underlying patterns in the data. Our methodology—from data cleaning and normalization to SOM training and comprehensive evaluation—offers a robust framework for phishing detection.

Key takeaways include:

- **Effectiveness of SOM:** The SOM successfully grouped similar URLs, providing both a visual and quantitative basis for phishing detection.
- **Evaluation Insights:** While the overall accuracy is promising, detailed evaluation metrics (precision, recall, F1 score, specificity) reveal areas where the model may require further refinement.
- **Future Directions:** Future work could explore advanced feature engineering, the integration of dimensionality reduction techniques (such as PCA), and the combination of SOM with other classification models to enhance performance.

Overall, this work contributes to the growing body of research on machine learning-based phishing detection, offering insights that could help develop more sophisticated, adaptive security systems.