OFMCDM/IRF: A Phishing Website Detection Model based on Optimized Fuzzy Multi-Criteria Decision-Making and Improved Random Forest

Md Abdullah Al Ahasan
Department of Computer Science
University of Regina
Regina, Canada
mai245@uregina.ca

Mengjun Hu
Department of Computer Science
University of Regina
Regina, Canada
mengjun.hu@uregina.ca

Nashid Shahriar
Department of Computer Science
University of Regina
Regina, Canada
nashid.shahriar@uregina.ca

***Abstract*—With increasing social and financial activities on the web, phishing has become one of the most critical threats in cybersecurity. Many methods have been proposed to identify phishing websites, such as fuzzy logic, neural networks, data mining, heuristic-based phishing detection, and machine learning. On the other hand, phishers develop more sophisticated techniques, decreasing the efficacy of the existing methods. This paper proposes a phishing detection model based on Optimized Fuzzy Multi-Criteria Decision-Making (OFMCDM) and Improved Random Forest (IRF). The model utilizes Uniform Resource Locator (URL) and Hypertext Markup Language (HTML) features to prevent sharing users' sensitive information such as username, password, social security, or credit card number. Our experiments show competitive results from our models compared to existing models, including Naive Bayes (NB), Logistic Regression (LR), K-Nearest Neighbor (KNN), and Decision Tree.**

***Index Terms*—Parallel Computing; URL Detection; Phishing Websites; Feature Extraction; Improved Random Forest**

I. Introduction

Phishing is one of the most common cybercrimes. APWG (Anti-Phishing Working Group) reported 1,270,883 yearly phishing assaults in 2022, with 1,097,877 happening from April to June [1]. Both numbers are record-breaking, showing the most damaging period for phishing history recorded by APWG. The finance industry, including banks, was the most common target in Q3 of 2022, accounting for 23.2% of phishing scams. The retail/e-commerce and social media industries also saw phishing attacks at 15%, and attacks against the Software As A Service (SAAS)/Webmail account for 17% of attacks. The cryptocurrency market saw a decrease in phishing attacks from 4.5% in Q2 to 2.0% in Q3 [2].

Phishing is a cat-and-mouse game between attackers, who employ a variety of techniques such as fake websites and victim investigation, and anti-phishing measures, which aim at detecting and preventing phishing attacks. Despite various technical and non-technical efforts to counter phishing, it remains a challenging problem due to the exploitation of human vulnerabilities and the constantly changing tactics of attackers. Additionally, the low cost and high reward of phishing make it an attractive tactic for scammers.

Machine Learning (ML) methods have been proven to be effective in detecting phishing websites. Specifically, ML algorithms can detect complex correlations between items of a similar nature. ML algorithms commonly contain two phases of training and testing. In the training phase, the algorithms learn from labelled examples. In the subsequent testing phase, researchers evaluate the performance and generalization of the algorithms. Random Forest (RF) shows the most promising results among the many ML models used in detecting phishing websites. Among them, Uddin et al. [3] compared ML-based website phishing detection methods based on URL information. They compared five ML models of Decision Tree, RF, KNN, Gaussian Naive Bayes, and XGBoost. They demonstrated that the RF algorithm outperforms other techniques with an accuracy of 97.0%. Algabri et al. [4] adopted a number of methods to select the highest associated characteristics in detecting phishing websites. The RF produced the most accurate classification results for two distinct experimental datasets. Using the best features, their RF model earned a maximum accuracy of 92.83%, followed by Convolutional Neural Network (CNN) with an accuracy of 90.46%. Following the effectiveness of RF models demonstrated in many existing works, we adopt RF to classify URLs in the proposed approach. Our approach can also be easily adapted to other ML methods.

Zhu et al. [5] proposed a model based on the updated Multi-Objective Evolution (MOE) optimization method and RF. The MOE/RF approach reduced the likelihood of falsely identifying phishing sites using accuracy as the detection target. Two new approaches were suggested to enhance the MOE performance: symmetric uncertainty-based population initialization and population state-based adaptive environmental selection. These two approaches were found to outperform existing ones. According to experimental results evaluating five different phishing datasets, this approach has a 99.4% accuracy rate.

Mandadi et al. [6] suggested several ML methods for categorizing URLs, including Support Vector Machine (SVM), Neural Networks, RF, Decision Tree, and XG boost. They used

10,000 URLs in total, 5000 of which were legitimate websites and 5000 of which were phishing websites. The dataset was shuffled and had a total of 17 features, which were used to train the model. The suggested methodology covered the RF and Decision Tree classifiers, which successfully distinguished legitimate and phishing URLs with an accuracy of 87.0% for RF classifiers and 82.4% for Decision Tree classifiers.

One challenge in detecting phishing websites is accurately identifying both phishing and legitimate websites. Phishers attract traffic by using similar URLs and designs to pretend to be legitimate websites. Most existing works focus on detecting phishing websites, but problems arise when legitimate websites are misclassified. The proposed OFMCDM classification approach addresses this issue. Using our OFMCDM/IRF model, we achieved good accuracy with a notable True Positive Rate (TPR), minimizing the misclassification of legitimate websites. Another challenge is to detect real-time phishing attacks. This is mainly because modern phishers are changing their tools and technology regularly to create new approaches to phishing. Our proposed approach adopts fuzzy logic and natural language processing techniques to analyze a URL and detect suspicious features. Accordingly, our proposed method can detect both previously-seen phishing as we have done the experiment on these datasets and brand-new zero-day attacks.

Our algorithm improves the existing malicious URL detection algorithms and detects phishing webpages in real time by analyzing the URLs with different machine learning algorithms. A summary of our main contributions is given below:

- Three data processing techniques based on URL and HTML feature extraction are defined: 1) word tokenization that divides incoming URLs into separate words and removes digits, 2) random word detection that identifies the length and number of URLs, and 3) maliciousness analysis that determines if the words are used fraudulently.
- The proposed model enhances the multi-class RF algorithm for classification, which can extract important features to provide improved accuracy, robustness, and reliability. Furthermore, it also enables real-time phishing detection.
- We combine ML methods and Optimized Fuzzy Multi-Criteria Decision-Making to analyze the characteristics of both phishing and legitimate websites and identify effective user-side features for phishing detection.

The rest of this paper is arranged as follows. Section II proposes the architecture of the OFMCDM/IRF model. Section III presents the experimental results. Section IV concludes the paper and outlines future work.

II. The Proposed Architecture of OFMCDM/IRF

The proposed OFMCDM/IRF approach to detecting phishing websites includes three main components as depicted in Fig. 1:

A. URL parsing;

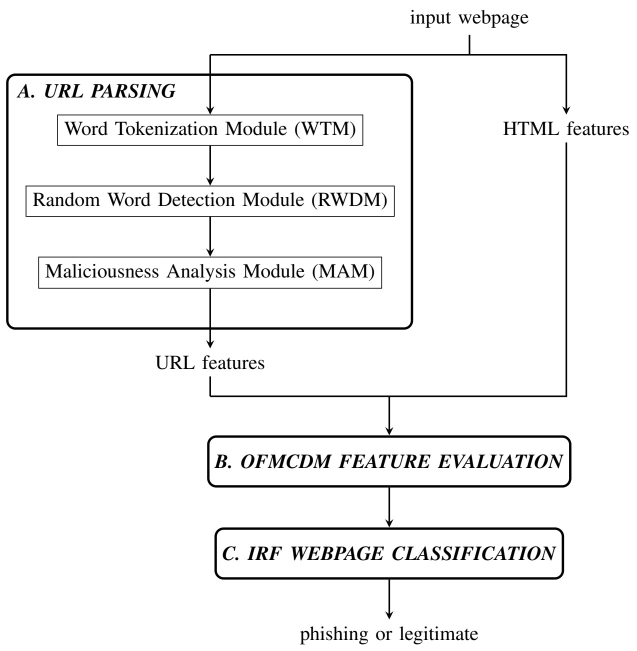
input webpage
A. URL PARSING
Word Tokenization Module (WTM)
Random Word Detection Module (RWDM)
Maliciousness Analysis Module (MAM)
HTML features
URL features
B. OFMCDM FEATURE EVALUATION
C. IRF WEBPAGE CLASSIFICATION
phishing or legitimate

Fig. 1: An architecture of the OFMCDM/IRF model

B. OFMCDM feature evaluation;

C. IRF webpage classification.

As shown in Fig. 1, two groups of HTML and URL features are obtained from the input webpage. HTML features are computed based on the HTML file of the webpage, such as the numbers of external and internal links, login form, the length of HTML content, and alarm window. URL features are computed based on the URL of the webpage by the first main component A of URL parsing. URL features include suspicious symbols, sensitive vocabulary, protocols, and the number of dots. An URL is parsed by three modules respectively for word tokenization, detecting randomly generated words, and analyzing malicious words. Both HTML and URL features are then analyzed by the second main component B of feature extraction using OFMCDM. A set of effective features are generated and input into the ML model IRF used in the third main component C of classifying the webpage as phishing or benign. The remaining part of this section discusses the three components in detail.

A. URL Parsing

The objective of the first main component of URL parsing is to obtain a set of features based on a URL. The URL is analyzed by three modules: 1) Word Tokenization Module (WTM), 2) Random Word Detection Module (RWDM), and 3) Maliciousness Analysis Module (MAM).

1) Word Tokenization Module (WTM): This module recursively splits the URL into a list of distinct words or tokens. Firstly, the URL is broken down into substrings at the special characters that are added by the phisher to make the URL more complicated. Secondly, the substrings are checked against a dictionary or vocabulary of words or tokens. If a substring is in the dictionary, we add it to the list of output words. Otherwise, the substring is recursively divided into smaller strings.

Algorithm 1 URL Parsing

```
1: Input URL
2: ▷ Word Tokenization Module (WTM)
3: A method handle to be proceeded.
4: Parse URL by special characters.
5: if Dictionary-Check then
6:     Add to Wordlist.
7:     Remove all subcodes with a character length less than three.
8:     Sort words according to their length.
9: end if
10: if Dictionary-Check-for-all-substrings then
11:     Add to Wordlist.
12: end if
13: ▷ Random Word Detection Module (RWDM)
14: for all Brand-Name and Keyword-Check do
15:     CountB=Brand-Name.
16:     CountK=KeyWord.
17:     for all Extract-Feature do
18:         if Random-Word-Detection then
19:             CountW=Random-Word.
20:         end if
21:         if length >7 then
22:             Secure login.
23:             Word tokenizer.
24:             Add to Wordlist.
25:             Analysis securely.
26:             Maliciousness analysis.
27:         else
28:             Add to Wordlist.
29:             Remove False Positives using threshold adjustment.
30:         end if
31:     end for
32: end for
33: ▷ Maliciousness Analysis Module (MAM)
34: Input Wordlist.
35: if Brand-Name and Keyword-Check then
36:     Add to Found-Wordlist.
37:     Calculate word similarity.
38: end if
39: if Edit-Distance <2 then
40:     Add to Similar-Wordlist.
41: end if
```

The process is described in the first part of Algorithm 1. We check whether the extraction process is set up before starting the next step. If it is, we add the extracted words to the list of words. We use a module called "tokenize" to separate words in the list. We check if each word is a valid dictionary word using the "enchanting" package [7]. If a word is longer than two characters and not in the dictionary, we break it into smaller sub-strings and check if they are in the dictionary. However, we need to be careful when creating the word list. The module may generate false positives because it has not been trained for the detection of choosing a meaningful word. Note that the next RWDM module will handle words longer than seven characters.

2) Random Word Detection Module (RWDM): Phishing URLs may contain randomly generated phrases where the probabilities of two consecutive letters do not follow the distribution in ordinary texts written in a certain language. In our approach, the Random Word Detection Module (RWDM) identifies the randomly generated phrases based on the Markov chain model [8], [9].

The process is described in the second part of Algorithm 1. We focus on letters only and do not count other characters, such as spaces or special characters. We check the consecutive letters of a given word in a random pattern and determine whether it is a meaningful term or a random word based on its significance. This is done by setting a threshold adjustment value for the fitness score. If the fitness score is high, it is considered a valid term. Otherwise, it is treated as a random word.

3) Maliciousness Analysis Module (MAM): This module determines if the words generated from the previous two modules are malicious. Particularly, MAM detects "typosquatting" attacks where phishers trick users into visiting fake websites by slightly altering the legitimate URLs, such as using "Goggle.com" instead of "Google.com". Typosquatting is a well-known threat that is difficult to eliminate completely. Nevertheless, there are strategies to reduce the risk of typosquatting, such as implementing security measures to detect and block malicious domains, educating users on the dangers of typosquatting, and encouraging the use of trusted sources for domain registration.

The process of MAM is described in the third part of Algorithm 1. We check if the words produced from the two previous modules match any brand names or keywords in a list. The Levenshtein algorithm [10], [11] is used to measure how similar a word is to the brand names or keywords. It calculates the distance between two words based on the number of changes (such as adding, removing, or changing letters) needed to convert one word into another. A similarity measure can be defined reversely to the distance. Such a similarity measure represents the graded degrees of non-maliciousness rather than crisp Boolean decisions. Accordingly, fuzzy sets and fuzzy logic [12] are appropriate rather than traditional crisp sets and Boolean logic. A fuzzy set uses a membership function to represent the degrees to which an object belongs to a set. Fuzzy logic generalizes the two truth values in Boolean logic into continuous degrees of truth, and accordingly, the logic operations are also redefined. With respect to the MAM module, the found words or similar words in the Algorithm 1 wordlist are represented as a fuzzy set. The similarity measure defines its membership function, representing the membership degrees of the words to the found word or similar word. Using fuzzy sets and fuzzy logic, the MAM module tackles the challenge of detecting both phishing and legitimate cases by introducing a graded transition between the two cases rather than a sharp shift.

B. OFMCDM Feature Evaluation

This step evaluates the URL features produced by the previous step as well as the HTML features. We present an OFMCDM method to evaluate the features. Fuzzy logic, fuzzy multi-criteria decision-making (FMCDM), and natural language processing (NLP) are commonly used together in phishing website detection to address the complexity and ambiguity of the problem. Fuzzy logic is used to handle imprecise and uncertain data, allowing for a more nuanced assessment of the likelihood of a phishing website. FMCDM combines fuzzy logic with multiple criteria decision-making (MCDM). MCDM performs the ranking of decision alternatives based on multi-criteria. In the context of phishing website detection, FMCDM is adopted to investigate the detection decisions, represented by fuzzy sets, based on multi-features as criteria. NLP is used to analyze the content of a website and identify text or language patterns commonly used in phishing attacks. Combining these techniques allows a more comprehensive and effective approach to phishing website detection to be achieved [13].

Our detection process first searches for the presence of hexadecimal values (represented by % in the URL), which can appear on both legitimate and phishing websites. Legitimate websites use them to replace special characters such as question marks, commas, spaces, or colons. Phishing websites use them to conceal the actual URL by replacing letters and numbers with hexadecimal values. To distinguish between them, we convert the hexadecimal values to their numerical representation in the ASCII table and check if the resulting number represents a valid letter or number.

Furthermore, we use parallel computing to improve the efficiency of our algorithm. As shown in Fig. 2, it consists of a master process for data processing and a slave process for URL checking. If the slave process finds the input URL in the trained dataset, there is no need to enter the master process. However, if the URL is not in the trained dataset, it goes through the master process.

The master process creates the Database after the Algorithm 2 is performed. The PhishingDetection procedure relies on the URLs variable defined outside the procedure. The algorithm uses a nested for loop to iterate over the phishing features and URLs and calculates a score for each URL based on the number of Key Phishing Characters (KPC) it contains. The variable keeps track of the number of phishing features found in the current URL and is used to calculate the phishing score. If the phishing score is greater than or equal to a threshold, the URL is declared as phishing. Otherwise, it is declared as benign.

We have a 10-layer system to detect phishing websites, where each layer has two rules checking for phishing characteristics shown in Table I. If the website URL and HTML features match these rules, it gets a score of 0.1. The final score of the website ranges from 0 to 1, with 0 being a low-risk website and 1 being a high-risk phishing website. For simplicity, we treat each rule equally important in identifying potential web security threats and adopt equal weights accordingly. One may easily generalize it into unequal weights if they could be given by a meaningful context.

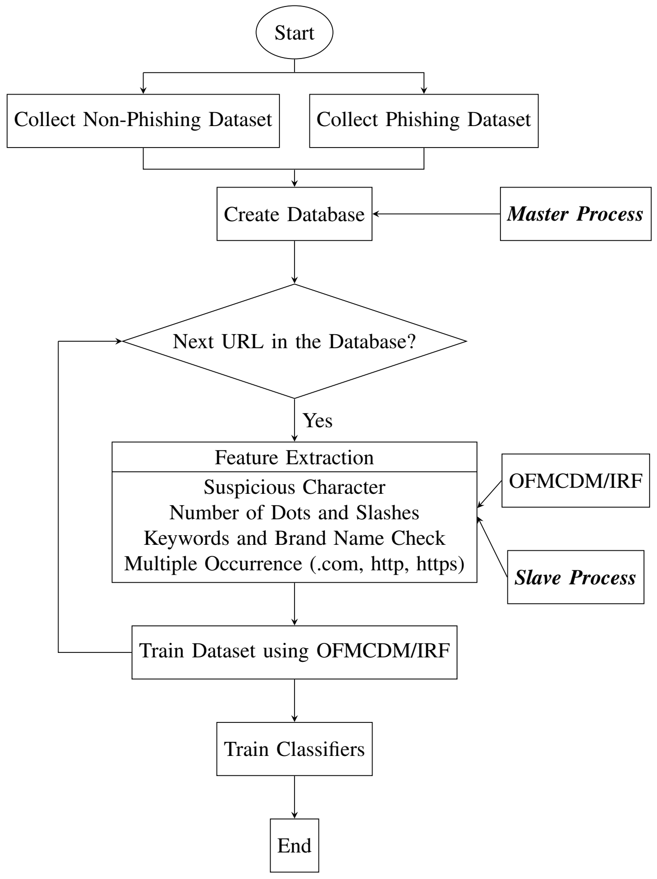
Start
Collect Non-Phishing Dataset
Collect Phishing Dataset
Create Database
Master Process
Next URL in the Database?
Yes
Feature Extraction
Suspicious Character
Number of Dots and Slashes
Keywords and Brand Name Check
Multiple Occurrence (.com, http, https)
OFMCDM/IRF
Slave Process
Train Dataset using OFMCDM/IRF
Train Classifiers
End

Fig. 2: The flow chart of OFMCDM with parallel computing

C. Webpage Classification Based on Improved Random Forest

The RF method binds a collection or a forest of decision trees. It can be utilized for both classification and regression with high robustness and accuracy. We present an IRF method as our classification for detecting phishing websites. As shown in Fig. 3, the proposed IRF first generates a random set of parameters for creating trees in the forest for each class. Each tree will be built according to the information gained from attributes in the training dataset. Secondly, the set of parameters is split to minimize the information gain. Each object that passes from the nodes of trees accumulates the decision weight for each class. Finally, the decision trees are created based on the chosen threshold of different levels of leaves. Each leaf contains the calculated weight for a particular class with size and its reference point. Each part of the text is sampled from the URL and ends in a leaf of a tree. The following equation is used to calculate the probability of the

TABLE I: The rule base in the ten layers of assessing webpages.

Layer Number	Rule base	Layer weight
1	Lengthy URL used as a connection. Using an IP address rather than a DNS name.	0.1
2	Many dots in the IP Address. Utilizing a changed port number.	0.1
3	Untrustworthy SSL certificate. Below a half year is the domain's duration.	0.1
4	Redirected pages and insecure websites. Requiring additional time to access accounts.	0.1
5	Accessing accounts requires extra time. Employing Java Scripts to conceal information.	0.1
6	Picture synchronization with other websites. Under Google's blacklist for that site.	0.1
7	Rule using forms with a submit button. Popping up windows.	0.1
8	Higher emphasis on responsiveness and safety. Honger access times to individuals.	0.1
9	Server form handler (SFH). Using mouse-over to make the link invisible.	0.1
10	In online address bars, prefixes and suffixes are added. Using @ sign and hexadecimal characters.	0.1

Algorithm 2 FMCDM Feature Evaluation

```
1: Input URLs
2: procedure PhishingDetection
3:     for each URL in URLs do
4:         $score \leftarrow 0$
5:         $KPC \leftarrow 0$
6:         for each phishing feature do
7:             if URL contains phishing feature then
8:                 $score \leftarrow score + feature\_weight$
9:                 $KPC \leftarrow KPC + 1$
10:            end if
11:        end for
12:        if $KPC > 0$ then
13:            $phishing\_score \leftarrow score/KPC$
14:            if $phishing\_score \geq threshold$ then
15:                Return URL_weight
16:            end if
17:        else
18:            Return URL_weight
19:        end if
20:    end for
21: end procedure
```

patch of text falling under any class:

$$pr(h(c,x,z) \mid \text{Leaf } t(y)) = \frac{pr(\text{dist}(x,y,z) \mid c, \text{Leaf } t(y))}{pr(c \mid \text{Leaf } t(y))}. \tag{1}$$

The notation $h(c,x,z)$ represents a latent variable that captures the relationship between the class label c, the input text x, and the context information z. In the context of a machine learning model, this latent variable represents the internal state of the model that helps to predict the class label for a given input text. To improve performance, one can use more informative features or representations of the input text and context, model the relationships between variables in a more expressive way, and use more advanced inference techniques to estimate the posterior distribution of the latent variable.

The left-hand side of Equation (1) represents the conditional probability $pr(h(c,x,z) \mid \text{Leaf } t(y))$. This is the probability of the latent variable h taking on the value $h(c,x,z)$ given that we have observed a leaf node Leaf $t(y)$ with label y. Specifically, we are interested in the probability of h taking on a particular value $h(c,x,z)$ given that we have observed a leaf node Leaf $t(y)$. The right-hand side of Equation (1) is about the probability of a hypothesis (in this case, the value of h) in light of new evidence (in this case, the observed leaf node Leaf $t(y)$ with label y). The numerator $pr(\text{dist}(x,y,z) \mid c, \text{Leaf } t(y))$ represents the likelihood of the observed data $\text{dist}(x,y,z)$ (i.e., the distance between x and y and z) given the value of c and the observed leaf node Leaf $t(y)$ with label y. The denominator $pr(c \mid \text{Leaf } t(y))$ represents the prior probability of c given the observed leaf node Leaf $t(y)$ with label y.

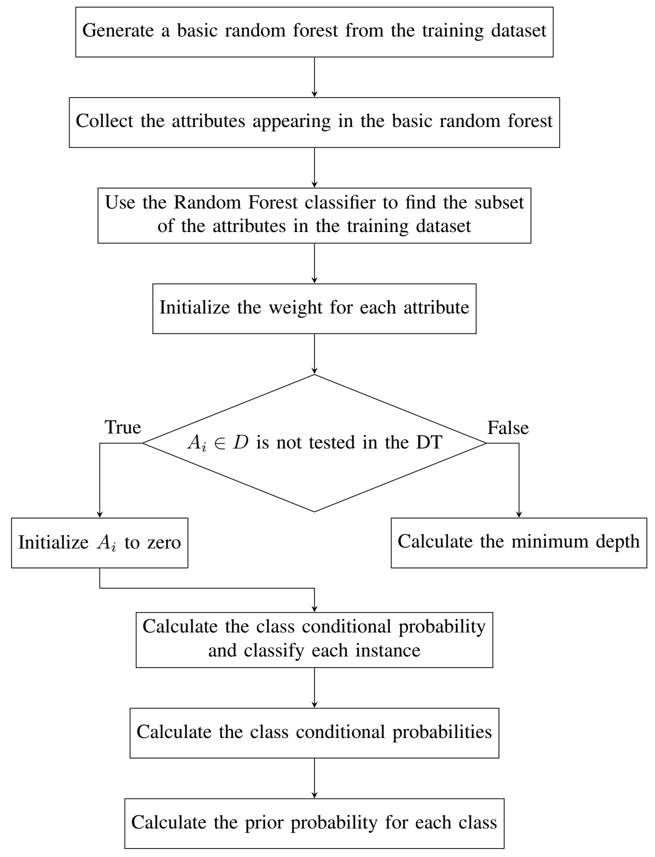
Generate a basic random forest from the training dataset
Collect the attributes appearing in the basic random forest
Use the Random Forest classifier to find the subset of the attributes in the training dataset
Initialize the weight for each attribute
$A_i \in D$ is not tested in the DT
True
False
Initialize A_i to zero
Calculate the minimum depth
Calculate the class conditional probability and classify each instance
Calculate the class conditional probabilities
Calculate the prior probability for each class

Fig. 3: The flow chart of IRF

III. EXPERIMENTAL ANALYSIS

We demonstrate the effectiveness of the proposed OFM-CDM/IRF approach to detecting phishing in the early life of an attack. The experiments assert that by combining and generating techniques in multiple areas, such as MCDM, fuzzy logic, NLP, and ML, our OFMCDM/IRF model can provide an effective and efficient way to detect phishing websites.

A. Dataset Preparation for Experiments

We collected our data from various reliable sources [14], [15]. PhishTank is a shared platform for data and Internet knowledge [16]. A project called OldPhishTank was run by OpenDNS, which was brought up by Cisco in 2015. The current database, known as NewPhishTank, is owned and run by the cybersecurity company Cofense. Security researchers, businesses, and individuals use these databases to detect and restrict access to known harmful websites in order to defend against phishing attempts. After registration, engineers and researchers can download a verified URL list of various formats. We downloaded the dataset in the CSV file format and compiled the first set of 10,000 URLs. It should be noted that the criminal tactics used to steal sensitive information are changing over time. We have selected 10,000 URLs for sensitive identity theft to track these new URL features and closely mimic the actual situation. Data were also collected from two public directories Alexa and DMOZ for non-phishing URLs, phishing URLs, and unknown URLs. Alexa and DMOZ servers were used to select a total of 20,000 URLs randomly. We randomly selected 10,000 URLs as phishing and 10,000 URLs as non-phishing. Among them 20% were Unknown URLs, and for labeling these unknown URLs we follow the approach described below.

Our approach adopts and extends a few existing methods for unknown URLs, each of which is capable of giving us an answer on its own about whether a website is a phishing site. These methods have different levels of effectiveness and credibility. So we assign a weight to each method and combine their results to calculate the health of a website by the following formula of Website Health (WH):

$$WH = \frac{(W_1V_1 + W_2V_2 + ... + W_nV_n)}{n}, \tag{2}$$

where $W_1, W_2, ..., W_n$ are the weights for n methods, and $V_1, V_2, .., V_n$ are values representing the results from the methods. Specifically, V_i is +1 if method i classifies the website as non-suspicious and -1 if it classifies the website as suspicious. If WH is negative, the website is considered suspicious. Otherwise, it is legitimate. The value of WH also reflects the degree of suspicion. For negative values, the less the WH, the more suspicious the website.

In the experience from our research experiment, we are currently using four methods:

- The first method is URL-based. Its value is denoted as V_{url}. Its weight is 0.5 or 50%.
- The second method is whitelist checking. Its value is denoted as V_{white}. Its weight is 1 or 100%.
- The third method is blacklist checking. Its value is denoted as V_{black}. Its weight is 1 or 100%.
- The fourth method is content-based checking. Its value is denoted as $V_{content}$. Its weight is 1.5 or 150%.

We do not normalize the weights because normalization is not always necessary or appropriate in every ML model. In our case, the normalization of weights also introduces additional computational overhead, which can slow down the training process and increase the memory requirement of our model. With the above four methods, the formula for our current implementation becomes:

$$WH = \frac{(0.5 * V_{url} + 1 * V_{white} + 1 * V_{black} + 1.5 * V_{content})}{4}. \tag{3}$$

The splitting ratio between training and testing data can vary depending on the size of the dataset. In this paper, we use a splitting ratio to use 70% of the data for training and the remaining 30% for testing.

B. Performance Metric

For the performance metrics, we consider phishing as positive (P) and benign as negative (N). As shown by the confusion matrix in Table II, real phishing websites classified as phishing are true positive (TP); those classified as benign are false negative (FN). Real benign websites classified as phishing are false positives (FP); those classified as benign are true negatives (TN). Based on it, we evaluate the performance using precision, recall, F1-score, and accuracy (ACC). Precision is also called positive predictive value (PPV). Recall is also called sensitivity or true positive rate (TPR). F1-score is the harmonic mean of precision and recall. These measures are calculated as follows:

$$\begin{aligned} \mathrm{PPV} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \\ \mathrm{TPR} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \\ \mathrm{F1-score} &= \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}, \\ \mathrm{ACC} &= \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{TN}+\mathrm{FN}}. \end{aligned} \tag{4}$$

TABLE II: The confusion matrix for phishing and benign

Real \ Predicted	Phishing (P)	Benign (N)
Phishing (P)	TP	FN
Benign (N)	FP	TN

C. Experimental Results

The specifications of our experimental environment are as follows: Python 3.7 (64-bit), Windows 11 Pro (64-bit) operating system, processor Intel Core (TM) i3-6320 CPU @ 3.90GHz, and install memory (RAM) 8GB with hard disk drive 1TB. We perform the experiments on eight representable algorithms individually, including IRF, RF, Decision Tree, SVM, Naive Bayes, KNN, AdaBoost, and XGBoost. Table III describes the different detection results. SVM performance is the worst overall. RF and IRF both outperform other

TABLE III: Experimental results with different classifiers

Classifier	PPV	TPR	F1-score	ACC
Improved Random Forest	**0.990**	**0.991**	**0.990**	**99.55%**
Random Forest	0.970	0.990	0.980	97.98 %
Decision Tree	0.964	0.977	0.971	97.02 %
SVM	0.957	0.955	0.94	94.45 %
Random Forest	0.970	0.990	0.980	97.98 %
Naive Bayes	0.940	0.977	0.958	95.67 %
KNN	0.940	0.977	0.958	95.67 %
Adaboost	0.908	0.963	0.935	93.24%
XGB	0.96	0.985	0.955	96.64%

TABLE IV: Experimental results of ensemble learning

Classifier	PPV	TPR	F1-score	ACC
RF+SVM+ParallelComputing	**0.990**	**0.990**	**0.990**	**99.01 %**
RF+SVM	0.970	0.980	0.960	96.98 %
DecisionTree+SVM	0.974	0.966	0.963	98.05 %
SVM	0.957	0.955	0.94	94.45 %
NaiveBayes+SVM+DecisionTree	0.987	0.866	0.876	87.6 %
RF+SVM+DecisionTree	0.879	0.876	0.876	87.9%

algorithms by using multiple features. The best accuracy is 99.55%, which demonstrates the effectiveness of our method.

We also performed experiments using different combinations of algorithms for ensemble learning. The results are given in Table IV. The highest accuracy of 99.01% is achieved by the combination of RF, SVM and Parallel Computing. The combination of RF, SVM and Decision Tree performed the worst overall.

D. Comparison with other works

Table V shows the performance of our proposed approach and other related works. We obtained the results of most related works from their papers and reproduced some of them for fair comparison. We compare all approaches with respect to TPR, false positive rate (FPR), ACC, whether or not they are search-engine-independent solutions (SEI), third-party-service-independent solutions (3rdI), and language-independent solutions (LI). The measure FPR is calculated based on the confusion matrix in Table II as:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FN} + \mathrm{TN}}. \tag{5}$$

As shown in Table V, the proposed OFMCDM/IRF model provides the highest accuracy. The work of Tan et al. [8] and Chen et al. [31] achieved higher TPR than our approach. However, these two methods produce much higher FPR. Almost all previous methods used search engines in datasets [14], [32]. However, there are some problems associated to their approaches. First, no real new sites appear at the top of search results, and this main feature leads to a challenging situation. Second, search engines do not produce accurate results in non-English queries [8]. Another work of Zhang etal. [22] achieved the lowest FPR. However, their work is third-party-service-dependent and language-dependent. Our method performs better by helping search engines find non-existent websites and produce lower FPR.

Our approach is compared to a number of works based on different metrics in Table V. It should be noted that there

TABLE V: Comparison of OFMCDM/IRF and other works

Approach	TPR	FPR	ACC	SEI	3rdI	LI
Montazer et al. [16]	88	12	88%	Y	N	Y
Xiang et al. [17]	92	0.4	95.8%	N	N	N
Gowtham et al. [18]	98.24	1.71	98.25%	N	N	Y
Zhang et al. [19]	97	6	95%	N	N	N
Tan et al. [8]	99.68	7.48	96.10%	N	N	N
Chiew et al. [20]	99.8	13	93.4%	N	Y	Y
El-Alfy et al. [21]	97.24	3.88	96.74%	N	N	N
Zhang et al. [22]	98.64	0.53	99.04%	Y	N	N
OFMCDM/IRF	**99.55**	**2.5**	**99.02%**	**Y**	**Y**	**Y**

TABLE VI: Comparison of OFMCDM/IRF and other works based on the number of features

Approach	Number of features	Feature approach	Accuracy
Islam and Abawajy [23]	21	Hybrid	97%
Almomani et al. [24]	21	Hybrid	98%
Khonji et al. [25]	47	Hybrid	97%
Gansterer and Polz [26]	30	Hybrid	97%
Ramanathan et al. [27]	200 topics	Content	97.7%
Ma et al. [28]	7	Hybrid	99%
Toolan and Carthy [29]	22	Hybrid	97%
Hamid and Abawajy [30]	7	Hybrid	92%
OFMCDM/IRF	**42**	**Hybrid**	**99.55%**

is no perfect approach that outperforms all the others in all aspects. According to Table V, the proposed approach OFMCDM/IRF has high TPR and ACC values compared to the other approaches, although it has a lower FPR value than the works from Xiang et al. [17], Gowtham et al. [18], and Zhang et al. [22]. Nevertheless, OFMCDM/IRF has advantages over these works in terms of providing independent solutions as indicated by the columns of SEI, 3rdI, and LI. Overall, the choice of approach would depend on the specific requirements and constraints of the task at hand. In addition, it may not be meaningful to compare different tables as they are evaluated with different datasets.

E. Comparisons Based on the Number of Features

Table VI compares the proposed OFMCDM/IRF model with other approaches based on the number of features. It shows that our approach has obtained the highest accuracy among all the approaches. Each row represents a study or approach, and the columns show the number of features used, the approach used to extract features (e.g., hybrid, content-based), and the accuracy achieved in the study. The first four rows show studies that used a hybrid approach with varying numbers of features, ranging from 21 to 47. All these four studies achieved high accuracy, indicating that a hybrid approach using multiple features works effectively. The fifth row shows a study by Ramanathan et al. [27] that used 200 topics as features extracted with a content-based approach and achieved an accuracy of 97.7%. The sixth and seventh rows show studies using a hybrid approach extracting 7 and 22 features, respectively. They achieved high accuracy of 99% and 97%. The eighth row shows a study by Hamid and Abawajy [30] that used a hybrid feature extraction approach producing only 7 features and achieved a slightly lower accuracy of 92%. Finally, the last row shows our proposed approach that extracted 42 features and achieved a high accuracy of 99.55% because of using reasonable number of features.

IV. Conclusion and future work

The proposed OFMCDM/IRF phishing website detection model is a promising approach to detecting phishing websites. This model uses a combination of fuzzy logic and multi-criteria decision-making to optimize feature selection and an improved random forest algorithm for classification. The use of fuzzy logic and multi-criteria decision-making allows the inclusion of subjective and uncertain data in the decision-making process, making the model robust and reliable. The improved random forest algorithm enhances the accuracy and efficiency of classification. The experimental results show that the proposed OFMCDM/IRF model outperforms existing phishing website detection methods regarding a few standard performance measures. This suggests that the OFMCDM/IRF model can be a valuable tool for identifying and preventing phishing attacks.

One direction for future work is to optimize the feature selection process, using methods other than fuzzy logic and MCDM. A second direction is to verify the effectiveness of the OFMCDM/IRF model and extend its application with various data sources such as social media or email phishing attacks. A third direction is to explore the use of other machine learning algorithms in the OFMCDM/IRF model, such as deep learning or support vector machines, to improve the accuracy and efficiency of phishing website detection.

Acknowledgment

The authors thank the anonymous reviewers of the paper for their constructive and invaluable comments. This work was supported in part by an NSERC Discovery Grant and in part by a Mitacs Accelerate Grant.

References

[1] Group APW, et al. Phishing Activity Trends Report Q2/2022. http://www apwg org/reports/apwg_report_Q2_2022 pdf. 2022;.
[2] Group APW, et al. Phishing Activity Trends Report Q3/2022. http://www apwg org/reports/apwg_report_Q3_2022 pdf. 2022;.
[3] Uddin MM, Islam KA, Mamun M, Tiwari VK, Park J. A Comparative Analysis of Machine Learning-Based Website Phishing Detection Using URL Information. In: 2022 5th International Conference on Pattern Recognition and Artificial Intelligence (PRAI). IEEE; 2022. p. 220–224.
[4] Aljabri M, Mirza S. Phishing Attacks Detection using Machine Learning and Deep Learning Models. In: 2022 7th International Conference on Data Science and Machine Learning Applications (CDMA). IEEE; 2022. p. 175–180.
[5] Zhu E, Chen Z, Cui J, Zhong H. MOE/RF: A Novel Phishing Detection Model based on Revised Multi-Objective Evolution Optimization Algorithm and Random Forest. IEEE Transactions on Network and Service Management. 2022;.
[6] Mandadi A, Boppana S, Ravella V, Kavitha R. Phishing Website Detection Using Machine Learning. In: 2022 IEEE 7th International conference for Convergence in Technology (I2CT). IEEE; 2022. p. 1–4.
[7] Buber E, Diri B, Sahingoz OK. NLP based phishing attack detection from URLs. In: International Conference on Intelligent Systems Design and Applications. Springer; 2017. p. 608–618.
[8] Tan CL, Chiew KL, Wong K, et al. PhishWHO: Phishing webpage detection via identity keywords extraction and target domain name finder. Decision Support Systems. 2016;88:18–27.
[9] Marchal S, François J, Engel T, et al. Proactive discovery of phishing related domain names. In: International Workshop on Recent Advances in Intrusion Detection. Springer; 2012. p. 190–209.
[10] Beijering K, Gooskens C, Heeringa W. Predicting intelligibility and perceived linguistic distance by means of the Levenshtein algorithm. Linguistics in the Netherlands. 2008;25(1):13–24.
[11] Ene A, Ene A. An application of Levenshtein algorithm in vocabulary learning. In: 2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI). IEEE; 2017. p. 1–4.
[12] Zadeh LA. Fuzzy sets. Information and control. 1965;8(3):338–353.
[13] Sap MNM, Yusof R, Rahman MZA, Ramli AR, Rahim AA. A review on fuzzy logic, FMCDM, and NLP techniques for phishing website detection. In: 2018 7th International Conference on Cyber Crime, Security and Digital Forensics (Cyberforensics). IEEE; 2018. p. 1–6.
[14] Ma J, Saul LK, Savage S, Voelker GM. Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining; 2009. p. 1245–1254.
[15] Peng T, Harris I, Sawa Y. Detecting phishing attacks using natural language processing and machine learning. In: 2018 ieee 12th international conference on semantic computing (icsc). IEEE; 2018. p. 300–301.
[16] Montazer GA, ArabYarmohammadi S. Detection of phishing attacks in Iranian e-banking using a fuzzy–rough hybrid system. Applied Soft Computing. 2015;35:482–492.
[17] Xiang G, Hong J, Rose CP, Cranor L. Cantina+ a feature-rich machine learning framework for detecting phishing web sites. ACM Transactions on Information and System Security (TISSEC). 2011;14(2):1–28.
[18] Gowtham R, Krishnamurthi I. A comprehensive and efficacious architecture for detecting phishing webpages. Computers & Security. 2014;40:23–37.
[19] Zhang Y, Hong JI, Cranor LF. Cantina: a content-based approach to detecting phishing web sites. In: Proceedings of the 16th international conference on World Wide Web; 2007. p. 639–648.
[20] Chang EH, Chiew KL, Tiong WK, et al. Phishing detection via identification of website identity. In: 2013 international conference on IT convergence and security (ICITCS). IEEE; 2013. p. 1–4.
[21] El-Alfy ESM. Detection of phishing websites based on probabilistic neural networks and K-medoids clustering. The Computer Journal. 2017;60(12):1745–1759.
[22] Hutchinson S, Zhang Z, Liu Q. Detecting phishing websites with random forest. In: International conference on machine learning and intelligent communications. Springer; 2018. p. 470–479.
[23] Islam R, Abawajy J. A multi-tier phishing detection and filtering approach. Journal of Network and Computer Applications. 2013;36(1):324–335.
[24] Almomani A, Gupta BB, Atawneh S, Meulenberg A, Almomani E. A survey of phishing email filtering techniques. IEEE communications surveys & tutorials. 2013;15(4):2070–2090.
[25] Khonji M, Iraqi Y, Jones A. Enhancing phishing e-mail classifiers: A lexical url analysis approach. International Journal for Information Security Research (IJISR). 2012;2(1/2):40.
[26] Gansterer WN, Pölz D. E-mail classification for phishing defense. In: European conference on information retrieval. Springer; 2009. p. 449–460.
[27] Ramanathan V, Wechsler H. phishGILLNET—phishing detection methodology using probabilistic latent semantic analysis, AdaBoost, and co-training. EURASIP Journal on Information Security. 2012;2012(1):1–22.
[28] Ma L, Ofoghi B, Watters P, Brown S. Detecting phishing emails using hybrid features. In: 2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing. IEEE; 2009. p. 493–497.
[29] Toolan F, Carthy J. Feature selection for spam and phishing detection. In: 2010 eCrime Researchers Summit. IEEE; 2010. p. 1–12.
[30] A Hamid IR, Abawajy J. Hybrid feature selection for phishing email detection. In: International Conference on Algorithms and Architectures for Parallel Processing. Springer; 2011. p. 266–275.
[31] Chen TC, Stepan T, Dick S, Miller J. An anti-phishing system employing diffused information. ACM Transactions on Information and System Security (TISSEC). 2014;16(4):1–31.
[32] Gupta A, Joshi J, Thakker K, et al. Content based approach for detection of phishing sites. Published in International Research Journal of Engineering and Technology. 2015;.