International Journal of Innovative Computing, Information and Control
Volume **9**, Number **3**, March **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 1065–1086

AN ENHANCED ONLINE PHISHING E-MAIL DETECTION FRAMEWORK BASED ON "EVOLVING CONNECTIONIST SYSTEM"

Ammar ALmomani[1], Tat-Chee Wan[1,2], Ahmad Manasrah[3], Altyeb Altaher[1]
Mahmoud Baklizi[1] and Sureswaran Ramadass[1]

[1]National Advanced IPv6 Centre
[2]School of Computer Sciences
Universiti Sains Malaysia
11800 USM, Penang, Malaysia
{ Ammarali; altyeb; baklizi; sures }@nav6.usm.my; tcwan@cs.usm.my

[3]Faculty of Information Technology and Computer Sciences
Yarmouk University
21163, Irbid, Jordan
ahmad.a@yu.edu.jo

Received January 2012; revised May 2012

Abstract. *One of the main problems in Internet security is unknown "zero-day" phishing e-mail attack, a new phishing e-mail that has not been trained on the old data sample or blacklisted. Existing phishing e-mail prevention mechanisms do not perform well against zero-day phishing e-mail attacks. This paper introduces a novel framework, called phishing dynamic evolving neural fuzzy framework (PDENFF), which adapts the "evolving connectionist system" based on a hybrid (supervised/unsupervised) learning approach. PDENFF adaptive online is enhanced by offline learning to detect dynamically the unknown zero-day phishing e-mails. Our analyses have shown that this framework is designed for high-speed "life-long" learning with low memory footprint and minimizes the complexity of the rule base and configuration. This framework achieves high performance, including high level of accuracy, true positive, and true negative results that reached up to 99%, 97%, and 98%, respectively, and an improvement between 3% and 13% comparison of the generated existing solutions to zero-day phishing e-mail exploits.*

Keywords: Phishing e-mail, Dynamic evolving neural fuzzy inference system (DENFIS), Evolving connectionist system (ECOS), Unknown zero-day, Online adaptive learning, Evolving clustering method

1. Introduction.

Phishing is an e-mail scam that employs two techniques. The first technique is related to social engineering schemes, which depend on forged e-mails; it claims that it originated from a legitimate company or bank. Then through an embedded link within the e-mail, the phisher tries to redirect users to fake Web sites. These fake Web sites are designed to obtain financial data from their victim fraudulently, including usernames, passwords, and credit card numbers.

The second technique involves technical deception schemes which depend on malicious software program that remains on a user's computer and is designed to obtain directly the victim's online account information. Occasionally, the phisher tries to misdirect the user to a fake website or to a legitimate one monitored by proxies [1].

The problem of phishing e-mail is becoming worse. A survey by Gartner in 2007 on phishing attack showed that about 3.6 million users in the USA lose money because of phishing [2]. The total losses amount is estimated as US$3.2 billion dollars. The number of victims increased from 2.3 million in 2006 to 3.6 million in 2007, which represents an increase rate of 56.5%. Another recent study reported by the Symantec Company showed that roughly more than 95.1 billion phishing messages were sent in 2010. This figure equates to approximately 260 million phishing e-mail sent daily by phishers to the customers of trusted companies [3].

Many problems arise due to phishing e-mails, most of which affect financial companies and their clients. Phishing e-mails range from very simple to very complicated messages and can deceive even the cleverest of Internet users. Phishing is capable of damaging electronic commerce because it causes users to lose their trust on the Internet [4].

Today, one of the main problems in e-mails is the so-called unknown "zero-day" phishing e-mails. Zero-day attacks are defined as attacks that phishers mount using hosts that are not blacklisted or using techniques that evade known approaches in phishing detection [5,6]. Phishing e-mail is so complex that it cannot be detected by current techniques because the phishers can be using new vulnerabilities never known before. On the other hand, some artificial intelligence (AI) techniques, such as support vector machine (SVM), neural network Multi Layer Perceptron (NNet-MLP) [8], and k-means algorithms [9], are able to detect phishing e-mails based on fixed features and rules [1,7]. The level of errors in the classification process will increase over time, especially when dealing with unknown zero-day phishing e-mails.

Phishing e-mail detection has been a major area of focus in a number of studies. In this paper, a new framework based on the *ECOS* [10,11] called Phishing Dynamic Evolving Neural Fuzzy Framework (*PDENFF*), is proposed. This framework is capable of determining whether e-mail is phishing or ham (legitemate) dynamically. The implementation of the proposed framework depends on *ECOS*, which adapts the evolving clustering method (ECM) as a part of the dynamic evolving neural fuzzy inference system (DENFIS) in an online mode along the dynamic neural fuzzy inference system (DyNFIS) to enhance the rule creation in an offline mode [12-14].

The proposed framework can detect unknown zero-day phishing e-mails. It has a high level of performance, including accuracy with low level of false positives (FPs) and false negatives (FNs), and other measurements. It consumes a short time for e-mail classification, consequently improving the level of system performance. The motivation behind the current work is to prove that the use of ECOS will allow the use of a new clustering technique that provides a higher level of accuracy within a short time for zero-day phishing e-mail detection.

The recent paper is going to be organized as follows. Section 2 presents the related works. Section 3 discusses the proposed framework. Section 4 shows the implementation and test results, and finally, Section 5 presents the conclusions and future work.

2. **Related Works.**

A number of studies have examined phishing or Web phishing in general, but relatively few have discussed phishing e-mail detection. Particularly, there is a lack of research on unknown zero-day phishing e-mails detection in an online mode. In the current study, a number of methods pertaining phishing filtering are discussed.

Most of the literature surveyed on phishing detection yields techniques implemented through Toolbars; for example, SpoofGuard [15], AntiPhish [16], and NetCraft anti-phishing toolbar [17], are some of the most distinguished examples of this approach, which commonly depend on Bayesian (Naïve-base) algorithms [8]. The toolbar approach remains problematic, however. Zhang et al. [18] empirically analyzed 10 anti-phishing toolbars and reported that the toolbars were only able to consistently detect less than 90% of phishing URLs, as well as incorrectly detected 42% of legitimate URLs as phish.

Some approaches use supervised or unsupervised machine learning techniques for phishing e-mail detection and prediction [8,19], while others use a hybrid systems (supervised/unsupervised) learning technique [9,20]. The main aim of the machine learning technique depends on classifiers, which attempt to make a map of the input to the desired output depending on a specific function. When dealing with classification problems, the main rule involved is learning several types of input or features to predict a desirable output.

Fette et al. (2007) [19] proposed a method called PILFER with a machine learning technique dependent on features extraction to detect phishing e-mails, and they used 10 features denoted to phishing e-mails for training data by using supervised learning algorithms by the random forest [21] and the SVM [22] as classifiers. This method has 96% accuracy, with 0.1% FP and 4% FN. However, this supervised learning method can detect the type of e-mails with the same set of features but cannot detect unknown zero-day phishing attack because this attack might poss new features not included in the learning process yet [19].

Abu-Nimeh et al. (2006) [8], compared six classifiers related to the machine learning technique for phishing prediction. Most of the adopted classifiers used supervised learning, and the results indicated that there is no standard classifier for phishing prediction. For example, if some classifiers have low levels of FP, they will have a high level of FN [8]. For example, a neural network (NNet) classifier has FP rate up to 5.58% and 21.72% FNs, Linear regression classifiers (LR) whose FP 04.89%, but obtained a large number of FN at 17.04% [8], which means that they are not accurate enough to detect the various types of phishing attacks.

On the other hand, Islam et al. (2009) [20], built a system consisting of three-tier classifications to detect phishing e-mails. The method depends on feature extraction, followed by sequential classification and sending of the output for the decision process. Therefore, if an e-mail is misclassified by any tier, the final decision will be determined in the final tier. The average accuracy, reaches 97%, but this technique suffers from long processing and complexity of analysis because it requires many stages before arriving at the final decision; on the other hand 3% of its dataset suffers from misclassification which directly affects the detection accuracy [20].

Yearwood et al. (2010) [39], proposed a new method that depends on profiling phishing e-mails. They concentrated on embedded hyperlink information by extracting 12 features related with embedded URL only that represents phishing e-mails. These features are divided into two classes (phishing e-mail and ham e-mail). Extracted features are represented by a binary value. An e-mail is classified as "1" if it has any of the 12 features, and "0" otherwise. Classifier algorithms such as *SVM* [19] and *ada boost* [20] classifiers were used for detecting phishing e-mails. However, this technique depended on the phishing e-mail with embedded hyperlinks only [39], which will make the FP and FN happen from time to time; therefore, this technique still manifests weakness in its classification role, and thus, uncertainty in the overall detection accuracy.

Clustering is an unsupervised learning approach, used rarely for the classification of phishing e-mails. To our knowledge, one of studying uses a clustering method for classifying phishing e-mails conducted by Dazeley et al. (2010) [9], who combined unsupervised clustering algorithms with supervised classification algorithms. First, a variety of independent clustering algorithms were used such as k-means [9] algorithm, Global k-means algorithm (GKM) [9] to randomize data. Second, consensus clustering was done combined with independent clustering. Third, the data was trained using consensus clustering used Nearest Neighbor clustering and neural network (NNet), and then the entire data set was classified. This technique increases the speed of the classification and enhances the accuracy compared with the k-means algorithm. However, the level of accuracy is still less than 94% [9].

Basnet et al. (2008), proposed a technique that works based on 16 features. This technique adapted many machine learning algorithms like neural network-multi layer perceptron (*NNet-MLP*) feed forward and k-means clustering algorithm to classify e-mail into phishing or legitimate e-mail [23]; however, k-means clustering algorithm is working in offline mode and the level of accuracy was 90.8% which is still low; also NNet-MLP are not suitable for adaptive online learning [12], for these algorithms cannot solve online and unknown zero-day of phishing e-mail attack.

As a result, one of the main challenges in detecting phishing e-mails is how to detect unknown zero-day phishing e-mails in online mode. Therefore, ECOS is a promising platform for phishing detection [24]. ECOS proved its adaptability in terms of classifying e-mails into phish or ham e-mails in online mode, as well as in terms of speed and use of a one-pass algorithm [24], which accesses the data only once from the memory to create the rule.

ECOS is a connectionist architecture that simplifies the evolution processes using knowledge discovery. It can be a neural network or a set of networks that run continuously and change their structure and functionality through a continuous relationship with the environment and with other systems. This system, like traditional expert systems, works with unfixed number of rules used to develop the AI [25]. It is flexible with respect to the dynamic rule, works on either online or offline mode, and interacts dynamically with the changing environment. Such a system can solve the complexity and changeability of many real world problems. It grows throughout the process and adopts many techniques [10,11].

DENFIS proposed by Kasabov and Song [12,13], is a first-order Takagi-Sugeno inference engine [26]. DENFIS is one of the new types of fuzzy inference systems used for evolving a connectionist system implementation. It is used for online learning with dynamic time series prediction. The strength of the fuzzy inference system increases through its hybrid (supervised/unsupervised) learning, which has the capability of adding new input and new features with new classes and still evolves. DyNFIS is a new extension of the original offline version of DENFIS [14].

3. Proposed Framework – PDENFF.

Our proposed framework, called PDENFF, is shown in Figure 1. In the proposed framework, ECOS is adapted based on the level of similarity among the four groups of phishing e-mail features. The proposed methodology is divided into four stages. The first stage is called pre-processing, used to extract 21 binary features from e-mails, which we called "*long vector*". The second stage is the e-mail object similarities used to decrease the size of feature vectors from 21 to four feature groups, which we called "*short vector*". The third stage includes the ECM and its offline extension (ECMc) to generate the basis of rules [27]. Finally, DENFIS is utilized in online mode as a fuzzy inferences system to create, update, or delete a fuzzy rule while the system is running. DyNFIS is also used in offline mode to enhance the rules in offline mode, enhance the level of classification accuracy, and decrease the error rate in the prediction process based on Gaussian membership function. However, the profile management framework is suggested to put to order the relationship between DENFIS and DyNFIS and employ the best rules in our framework. Full details for each part are explained next.

Before discussing the stages of our framework, we should know that PDENFF collects and filters e-mail separately and sequentially. The first PDENFF stage implemented by pre-processing is explained as follows:

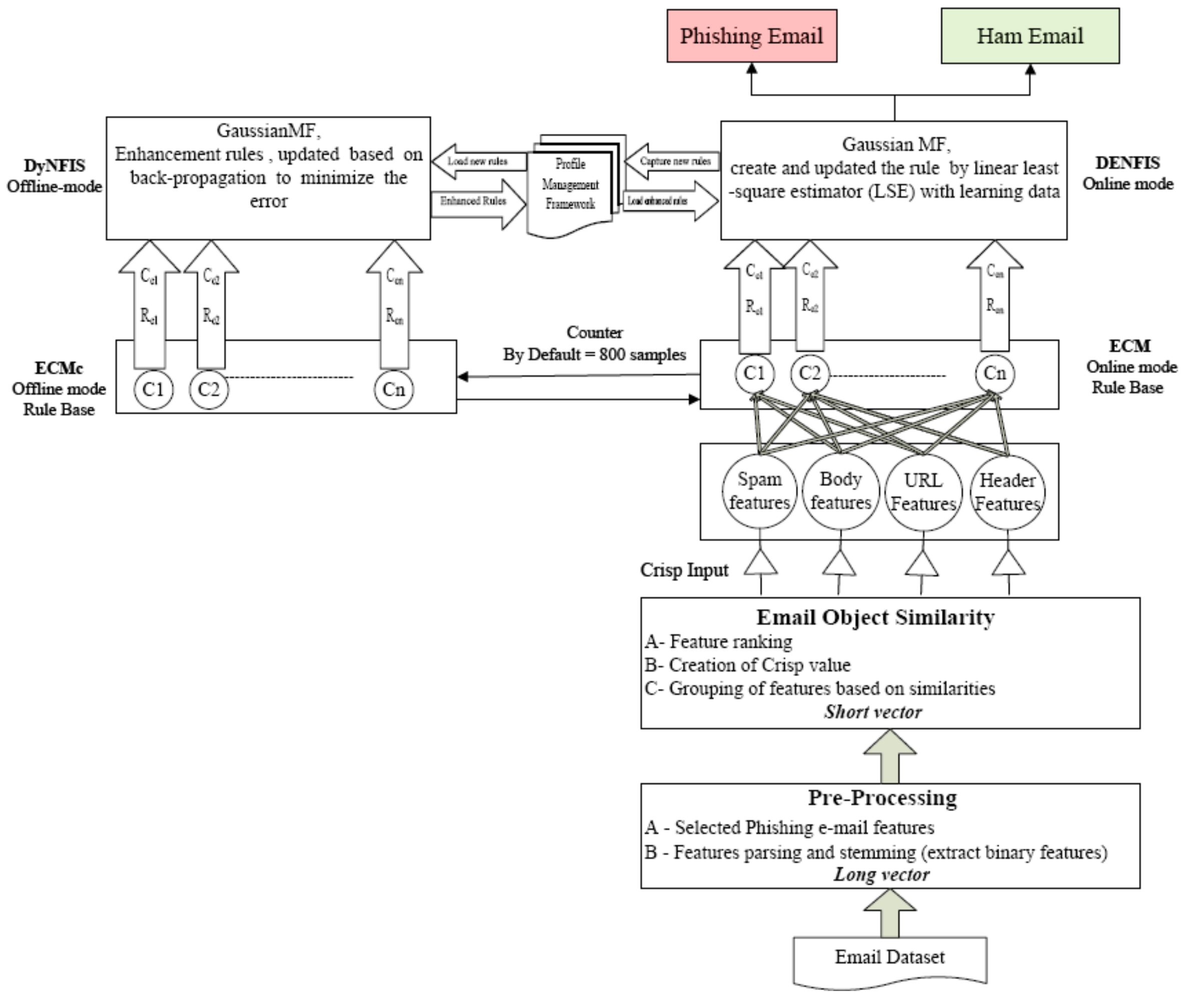

Phishing Email
Ham Email
GaussianMF, Enhancement rules , updated based on back-propagation to minimize the error
DyNFIS Offline-mode
Load new rules
Enhanced Rules
Profile Management Framework
Capture new rules
Load enhanced rules
Gaussian MF, create and updated the rule by linear least -square estimator (LSE) with learning data
DENFIS Online mode
C_{c1} R_{c1}
C_{c2} R_{c2}
C_{cn} R_{cn}
ECMc Offline mode Rule Base
C1
C2
Cn
Counter
By Default = 800 samples
ECM Online mode Rule Base
Spam features
Body features
URL Features
Header Features
Crisp Input
Email Object Similarity
A- Feature ranking
B- Creation of Crisp value
C- Grouping of features based on similarities
Short vector
Pre-Processing
A - Selected Phishing e-mail features
B - Features parsing and stemming (extract binary features)
Long vector
Email Dataset

FIGURE 1. PDENFF

3.1. **Pre-processing.** Pre-processing involves two steps. The first step involves the selection of phishing e-mail features, and the second one involves the parsing and stemming of e-mails.

To represent the most effective characteristics of phishing e-mails, 21 features were selected to obtain better accuracy in terms of the classification process [4,28,29]. The vector generated in this stage is designated as *long vector* because it decreases the size of this vector to be transformed to *short vector*, which will be discussed in the next phase. The features selected in our framework are discussed next.

3.1.1. *Selected phishing e-mail features.* PDENFF uses the features in the first learning process as seed to detect phishing e-mails. However, the system should be able to detect zero-day phishing e-mails without any prior knowledge of the phishing e-mail itself. The proposed method adapts unknown values of feature vector from the evolving clustering nature of e-mails to build the evolving rules. Hence, the feature behavior in our system is not a fixed factor, which is unlike the other approaches. Evolving the rule can include a new value of feature vector related to a new phishing e-mail attack. Therefore, the system can work with this new attack without prior knowledge of the feature vector value itself. The most effective 21 extracted features generated by many authors were adopted and some of the sub-features were merged into one feature, such as Feature numbers 14 and 20. We also suggested a new feature, such as Feature number 10. We suggested dividing all features into four groups because they represent all parts of the message. All features are binary in nature. The groups of features are discussed below.

Group 1: External features

These features are extracted from the open source SpamAssassin tool [32], which includes the following:

1-Spam features (spamfeatures)

This is a binary feature. Take a value of 1 if the message is classified as spam and 0 otherwise. Phishing e-mails comprise a subset of spam but not vice versa. In the current work, the open source SpamAssassin version 3.2.3.5 is used with the default rule, and using the default threshold 0.5 [19,30].

Group 2: Body-based features

All the body-based features occur in the body of an e-mail, which are explained as follows:

2-HTML e-mail (body_html)

This is a binary feature that returns 1 if HTML code is embedded in body of message, and 0 otherwise. At present, creating a phishing e-mail is difficult without using an HTML code because an HTML code enables an embedded link to connect directly to other Web sites [19].

3-Body of Multipart (body_multipart)

This is a binary feature that returns 1 if the message has a multipart MIME type, and 0 otherwise, our scanning search on "*Content-Type: multipart/alternative*" to detect this features [19].

4-Verify your account phrase (body_Verifyphrase)

This is a binary feature that returns 1 if the message has the phrase "Verify your account" and 0 otherwise [31].

5-"OnClick" JavaScript event (body_JSonclick)

This is a binary feature that returns 1 if the message has an "onClick" JavaScript event and 0 otherwise [31].

6-Code of JavaScript to Change the status bar (body_JSchangebar)

This is a binary feature that returns 1 if the message has JavaScript code to modify the status bar, and 0 otherwise [30].

7-Code of Java script (body_javascript)

This is a binary feature that returns 1 if the message has JavaScript code, and 0 otherwise. by match for "javascript" string [19].

8-Code of Java script to open popup windows (body_JSpopup)

This is a binary feature that returns 1 if the message has JavaScript code to open popup windows, and 0 otherwise [31].

Group 3: URL based Features

These features are extracted from the URL links of the e-mail, which included as follows:

9-html-links (url_htmllink)

This binary feature takes a value of 1 if there are embedded links in HTML part and 0 otherwise [32].

10-Number of dots in a link (url_nodots)

Generally, a legitimate company will have no more than three dots on its domain name [19], *We suggest.* It takes a value of 1 if there are more than three dots in the domain and 0 otherwise.

11-Non matching between target and text of urls (url_TarDiflink)

Phishers often use HTML e-mails, in which it is possible to show a link that says paypal.com but actually links to badsite.com. like <a href="badsite.com"> paypal.com</a >. This is binary features, if they have different host a value "1" and "0" otherwise [19].

12-Url IP address (url_ip)

A binary feature that returns 1 if the message has IP addresses rather than a qualified domain name, and 0 otherwise [19].

13-Image links (url_imagelink)

Some attackers use an image as a link to hide fraudulent URLs [31]. This is binary feature, return 1 if e-mail has images links and 0 otherwise.

14-URL bag of word links (url_bagword)

A binary feature that returns 1 if the message has one of human-readable link text, which included one or more of the following words: click; here; login; or update in the text portion of their links in order to hide a suspicious domain name [31].

15-URL has two domains (url_twodomain)

This is a binary feature that returns 1 if the URL founded has two domain names and 0 otherwise. For example, http://paypal.com.phish.com, has *two* different domain *paybal.com* and *phish.com*; generally, this does not happen in ham e-mail [33].

16-Non-standard port in the URL (url_nonstport)

A server accesses Web pages using ports, and a few phishers use non-standard ports to hide their identity and location. This is a binary feature that takes a value of 1 if the e-mail message uses a port other than 80 or 443 [29], and 0 otherwise.

17-URL containing hexadecimal characters or @ symbol (url_hexorat)

Some attackers use hexadecimal character codes to hide embedded URLs. Attackers can write an IP address using the "%" symbol to build a hexadecimal number. Sometimes, they use the "@" symbol to confuse users. This binary feature takes a value of 1 if the message URL contains either the "%"or @ symbol, and 0 otherwise [19].

Group 4: Features Header

This is a group of features, which represent the features of header message between sender and receiver; we selected some of those features as follows.

18-Subject replay word (sub_replay)

This is a binary feature that returns 1 if the "Re:" word is appearing in the subject field of a given message, and 0 otherwise [31].

19-Difference between the sender domain from the domain of the embedded links (diffsenlindom).

When the link embedded in the HTML does not equal the sender's domain, it is most likely a phishing e-mail. This is a binary feature, which returns 1 if the domain name in the "from" field does not equal the domain name in the URL, and 0 otherwise [19].

20-Subject (bank, verify, debit) word (sub_words)

This is a binary feature that returns 1 if the message's subject line has the word *bank*, *verify*, *debit*, and 0 otherwise [31].

21-Sender e-mail address uses different replay address (sendiffreplyto)

This is binary feature that returns 1 if sender e-mail address uses different replay address, and 0 otherwise [31].

3.1.2. *Feature parsing and stemming to extract binary features – Long vector.* Parsing is a process used to extract e-mail features. Stemming is a process that cleans the text data included in the e-mail features. The data are then converted to a binary value $(1, 0)$, which takes a value of one if the e-mail is likely to be a phishing e-mail and zero otherwise. The system takes the processed e-mail dataset to the next stage. A set of methods is used to extract all 21 probable useful features from each e-mail message by writing a series of short JAVA scripts and Python programming language to extract the features programmatically as follows:

Let $M = \{m_1, m_2, \ldots, m_{|M|}\}$ represent all e-mail messages and $F = \{f_1, f_2, \ldots, f_{|F|}\}$ be the vector of the features space, where $|M|$ and $|F|$ are the number of messages and the number of feature vectors, respectively. Let v_{ij} be the value of the Jth feature of the Ith message. As a result, each message appears as $V_i = (v_{i1}, v_{i2}, \ldots, v_{i|F|})$, and each message is $V = \{v_{ij}\}$, where $i = 1, 2, \ldots, |M|$ and $j = 1, 2, \ldots, |F|$.

After the parsing and stemming features in PDENFF are discussed, the generation of the *short vector* of features is discussed next.

3.2. **E-mail object similarity.** In this stage, we convert the feature vector from *long vector* to *short vector* for three reasons. The first one is to decrease the complexity of rule generation in our framework. The second reason is to decrease the number of rule generation, and the third is to increase the speed of classification process in our system. Two processes have to be run to accomplish these processes after the binary values are obtained from the pre-processing stage, which represent each e-mail in the dataset. The processes are as follows:

3.2.1. *Feature ranking.* To decrease the size of the feature vector and save the effect of all feature vectors, *we suggest* using a new technique called feature ranking and value calculation. Feature ranking is used to determine the most effective feature from the extracted features. One of the most effective algorithms used in feature ranking is the information gain ratio (IGR) [31,34].

The system uses two main measures to develop the value of the feature vector. The first one is *entropy*, which measures the disorder in a system, and the second one is *information gain*, which measures the decrease in entropy achieved in the classification based on a particular feature.

Entropy. Assuming that a collection of data samples D is available, we can calculate the entropy of D – $E(D)$ – in the system as follows:

$$E(D) = \sum_{i=1}^{N} -p_i \log_2 p_i \tag{1}$$

where N is the number of classes in the dataset and p_i is the possibility that a particular instance belongs to class i. In our dataset, two classes exist: ham and phishing e-mails, respectively. Equation (2) gives the formula for the two classes of problem.

$$E(D) = -P_h \log_2 p_h - P_p \log_2 p_p \tag{2}$$

where P_h represents the possibility that a sample is a ham e-mail and p_p denotes the probability that it is a phishing e-mail [33].

Based on these entropy values, we can proceed to the examination of the effects of each feature on the information gain algorithm as follows:

Information gain. The most effective attribute for the classification process is the reduction entropy based on the largest amount. The measurement for this *information gain* [31] to calculate the *information gain* of attribute F over dataset D is shown by Equation (3).

$$G(D, F) = E(D) - \sum_{v \in values(F)} \frac{D_v}{D} E(D_v) \tag{3}$$

where $E(D)$ is the entropy of the all datasets calculated based on Equation (2), F is the attribute in which the information gain is measured, and D_v is the number of attributes in D. F has the value v, and $E(D_v)$ is the entropy of this subset of the dataset.

Table 1 provides a complete ranking of all features selected for the proposed framework. The training set was composed of 8,000 samples of phishing and ham e-mails collected from a well-known dataset used from most researchers in this field, which includes phishing e-mail from the Monkey website [35] and ham e-mail from the SpamAssassin project [36]. The more the information gains are, the more helpful will a feature be. From the investigation, "spamfeatures" was found to have the best quality, whereas "body_JSpopup" was the least helpful and possibly caused noise in the classifier. This ranking was used to build the value of each feature. However, our system depends on the evolving rule; thus, if this ratio will be changed in the future, the rule can adapt to this change by evolving with the rule itself, which will be explained in the next section. However, up to this stage, we will use the standard dataset to calculate this ratio as a seed for our database in the future.

3.2.2. *Creation of crisp value.* This step will help group the features in the next step based on similarity. In addition, fuzzy logic is part of our framework. To work with crisp value, therefore, *we suggest* to convert the binary values $(0, 1)$ of all e-mail datasets into crisp values for the development of crisp value by dividing all features on a 1,000 score based on Algorithm (4) shown in [24].

$$X_i = (1000 * IGR_i/SUM(IGR_i)) \tag{4}$$

where X is a crisp value, i is the feature number, and IGR is the information gain ratio. Then, for each feature in the dataset, every binary value is multiplied with its crisp value, and the product will be used in our framework, as shown in Table 1. The value is entered into the system after grouping to construct the vector of features.

Table 1. Phishing e-mail features with IGR

No	Features (X_i)	IGR Ranking	Crisp value
1	spamfeatures	0.80254	176
2	body_html	0.69261	152
3	url_htmllink	0.64324	141
4	url_nodots	0.62678	138
5	url_TarDiflink	0.36063	79
6	sub_replay	0.29624	65
7	diffsenlindom	0.26995	59
8	body_multipart	0.18317	40
9	url_ip	0.13054	29
10	url_imagelink	0.11872	26
11	url_bagword	0.11294	25
12	sub_words	0.09481	21
13	url_twodomain	0.05421	12
14	sendiffreplyto	0.04322	9
15	body_JSonclick	0.03441	8
16	url_nonstport	0.02966	7
17	body_Verifyphrase	0.02615	6
18	body_JSchangebar	0.01512	3
19	body_javascript	0.00557	2
20	url_hexorat	0.00522	1
21	body_JSpopup	0.00441	1
sum		**4.55014**	**1000**

TABLE 2. Phishing e-mail groups and features values

Group Features	Features	Crisp value	summation range
External Features	spamfeatures	176	0-176
Body based Features	body_html	152	0-212
	body_multipart	40	
	body_JSonclick	8	
	body_Verifyphrase	6	
	body_JSchangebar	3	
	body_javascript	2	
	body_JSpopup	1	
URL based Features	*url_htmllink*	141	0-458
	url_nodots	138	
	url_ TarDiflink	79	
	url_ip	29	
	url_imagelink	26	
	url_bagword	25	
	url_twodomain	12	
	url_nonstport	7	
	url_hexorat	1	
Header based Features	sub_replay	65	0-154
	diffsenlindom	59	
	sub_words	21	
	sendiffreplyto	9	
Sum			0-1000

3.2.3. *Grouping of features based on similarities.* The grouping of features based on similarities is a process that makes the data undergo classification processes easily and rapidly. Four groups are constructed. Table 2 shows the phishing e-mail groups and the features values.

In Table 2, the first group is classified as *External Features* group consisting of spam features. The second group is called the *Body-based features* consisting of the summation of the values of seven features. The third group is the *URL-based Features* consisting of the summation of the values of nine features, and the fourth group is the header-based features consisting of the summation of the values of four features. The vector now consists of four values. *Short vector* represents 21 features of the *long vector*. This grouping aims to decrease the size of the feature vector by saving the effect of each feature, thus helping the system have more control in the classification process. In next stage, the process of how our framework will use the crisp input value *short vector* to work with ECOS for the classification of the input samples in the two classes (ham and phishing e-mails) is discussed. Figure 1 shows the process of adaptive DENFIS and DyNFIS for the classification of crisp input samples based on Gaussian function.

3.3. **Adaptive ECM and its extension, ECMc.**

ECM online, along with its extension, constrained optimization called ECMc-offline [27] , is adopted for the clustering process because ECM is fast. The ECM algorithm works in a very straightforward manner. One sample is needed at a time and performed by one-pass dynamic clustering algorithm. Partitioning of an input space is also done where no prior knowledge about the optimum number of clusters is required [12]. The one-pass algorithm sequentially reads the input exactly once, without unbounded buffering. ECM constructs a powerful method that is able to classify and partition e-mail data input. Each cluster is described by a cluster center C_{cn} and cluster radius R_{cn}. The cluster centers (called prototypes) are described by the maximum distance $d_{\max}$ between the input samples and the nearest cluster center; the distance cannot be longer than some distance threshold value D_{thr}.

ECM is used to allow PDENFF to deal with noisy datasets. ECM with ECMc works in DENFIS and DyNFIS, respectively, to build the base of evolving rules. The cluster center does not essentially fall in the center of gravity of the "real center" in ECM online. On the other hand, the offline version of ECM is proposed to deal with such problem of cluster centers whenever they are not positioned at the center of gravity. Thus, *we suggest* a dynamic system between ECM and ECMc in our framework. ECMc optimizes the final result in offline mode by capturing stream of e-mails while the system is working in the online mode. The idea came from inspiration of the combination of ECM and ECMc using the enhanced evolving clustering method [37].

We suggest to enhance the clustering of every 8,000 samples captured as a default parameter of the system because the time required to receive a new e-mail is not known in advance; thus, we build our enhancement based on the number of e-mail samples. Figure 2 shows the flowchart of ECM with its extension ECMc as it works in our framework.

Below we explain the algorithm steps of combination between ECM and ECMc. Note that steps from one to five related to the offline mode of *ECMc* and the steps from six to eleven related to the online mode of *ECM*.

Step 1: if it is the first time of inputs go to step 6 to create new cluster. Else create the cluster center C_{cj}, $j = 1, 2, 3, \ldots, n$ that already created before.

Step 2: determine the membership matrix U where the element U_{ij} is 1 if the ith data point Z_i belongs to C_j, and 0 otherwise. Once the cluster centers C_{cj} are defined, the values U_{ij} are derived as

$$\begin{aligned}&\text{IF } ||Z_i - C_{cj}|| \leq ||Z_i - C_{ck}||, \text{ For } k = 1, 2, \ldots, n,\ j \neq k; \\ &\text{Then } U_{ij} = 1, \text{ Else } U_{ij} = 0.\end{aligned} \tag{5}$$

Step 3: use the condition of minimization method to update the cluster centers, as per the following equation.

$$||Z_k - C_{cj}|| \leq D_{thr} \tag{6}$$

Step 4: to optimize the cluster. Calculate the objective function J_j according to the following equation.

$$J_j = \sum\nolimits_{j=1}^{n} \left(\sum_{k, Xk \in Cj} || \, Z_k - C_{cj} \, || \right) \tag{7}$$

Within cluster C_j for each $j = 1, 2, 3, \ldots, n$.

Step 5: if the result is less than certain tolerance value, or the result after compared with the earlier iteration is less than threshold value, or the iteration number for the optimization is greater than certain value, go to step 7, else go to step 2.

Step 6: create a new cluster C_1 by suggesting the place of the first input from the input samples as a first cluster center Cc_1, put the cluster radius $R_{U1} = 0$.

Step 7: if all input data from the data stream, have been implemented, then STOP, else the input sample Z_i which is in progress is taken with normalized Euclidean distance D_{ij}, between the input data sample with all n clusters center C_{cj} that created previously.

$$D_{ij} = ||Z_i - C_{cj}||, \text{ where } j = 1, 2, \ldots, n, \text{ is calculated} \tag{8}$$

Step 8: if we have cluster C_m with its center C_{cm} and radius R_{um}, while the distance value D_{im} represent the minimum distance between the cluster center C_{cm} and input Z_i

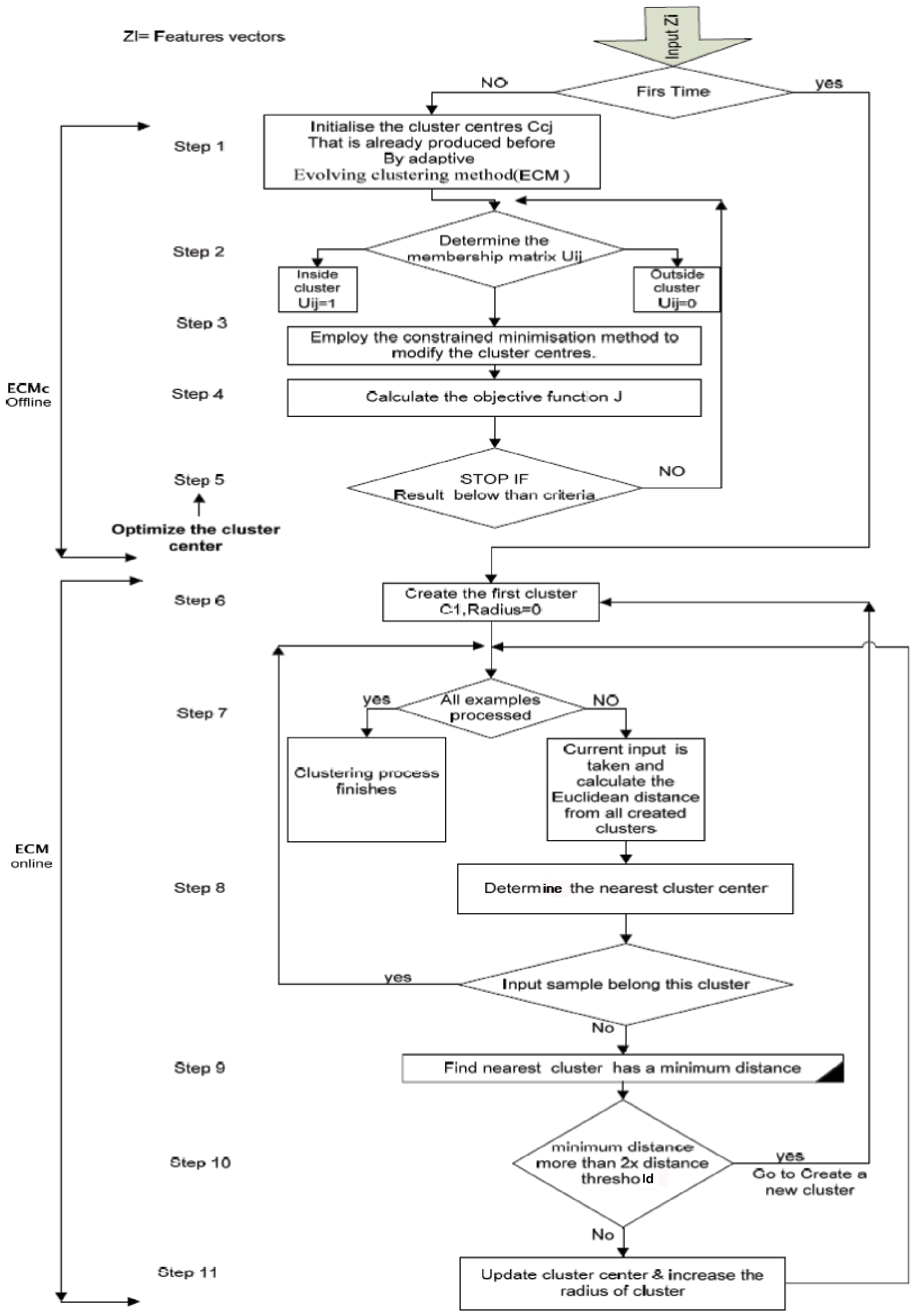

ZI= Features vectors
Input Zi
Firs Time
NO
yes
Step 1
Initialise the cluster centres Ccj That is already produced before By adaptive Evolving clustering method(ECM)
Step 2
Determine the membership matrix Uij
Inside cluster Uij=1
Outside cluster Uij=0
Step 3
Employ the constrained minimisation method to modify the cluster centres.
ECMc Offline
Step 4
Calculate the objective function J
Step 5
STOP IF Result below than criteria
NO
Optimize the cluster center
Step 6
Create the first cluster C1,Radius=0
Step 7
yes
All examples processed
NO
Clustering process finishes
Current input is taken and calculate the Euclidean distance from all created clusters
ECM online
Step 8
Determine the nearest cluster center
yes
Input sample belong this cluster
No
Step 9
Find nearest cluster has a minimum distance
Step 10
minimum distance more than 2x distance threshold
yes
Go to Create a new cluster
No
Step 11
Update cluster center & increase the radius of cluster

Figure 2. ECM and its extension ECMc

which is defined by this equation

$$D_{im} = ||Z_i - C_{cm}|| = \min\{D_{ij}\}, \text{ for } j = 1, 2, 3, \ldots, n \tag{9}$$

$$\text{And } D_{im} \le R_{um} \tag{10}$$

When the current input sample Z_i belongs to this cluster, then go back to step 7.
Step 9: based on the calculation of the distance values, discover a cluster C_a (have center C_{ca}, cluster radius R_{ua}, with distance value D_{ia} which has the shortest distance value S_{ia}).

$$S_{ia} = D_{ia} + R_{ua} = \min, \quad j = 1, 2, 3, \ldots, n \tag{11}$$

Step 10: if the shortest distance value $S_{ia} > 2D_{thr}$ (while D_{thr} represent the distance threshold parameter) then sample Z_i will not belong to any existing cluster. Then go back to step 6 to create a new cluster.
Step 11: if the shortest distance value $S_{ia} \le 2D_{thr}$, the cluster C_a is updated by moving its center, C_{ca}, with increasing the value of cluster radius R_{ua}. While $(R_{ua}^{new} = S_{ia}/2)$ and the new cluster center C_{ca}^{new} is placed on the line connecting input vector Z_i with the old cluster center C_{ca}. From the last calculation the distance between the new cluster centers C_{ca}^{new} to the input sample Z_i is equal to R_{ua}^{new}.

$$C_{ca}^{new} = Z_i - \left((C_{ca} - Z_i) \times \frac{\left(\frac{S_{ia}}{a}\right)}{D_{ia}} \right) \tag{12}$$

Then go back to step 7.

The group centers are implemented based on evolving the features vector of e-mails in the input stream, which depends on the distance between four groups of feature vectors. With the level of similarity between the values of group features, the final result will determine to which group the e-mail belongs based on the fuzzy rule created by *DENFIS*. The main output that will be taken from the *ECM* algorithm is the centers of the clusters and the cluster radius for dynamic rule creation.

In next phase we will explain how ECM and ECMc will work with fuzzy inferences system based on DENFIS and DyNFIS respectively to build full evolving dynamic system able to distinguish between phishing e-mail and ham e-mail in life-long learning system. However, this is the first time the DENFIS and DyNFIS will be used to solve phishing e-mail problem so we have a novelty in adaptive like this algorithms see Figure 1.

3.4. **Rule creation based on fuzzy inference system.** In this phase, an evolving rule is generated for classification process. We suggest three parts to build unlimited "life-long" training system that will enhance the rules while the system is working in the online mode. The first one depends on the DENFIS-online mode. The second one depends on the DyNFIS-offline mode to enhance the rules. Finally, profile management framework *was suggested* to arrange the relationship between the first and second parts. The rule for each part in our framework is explained below.

3.4.1. *Rule creation based on DENFIS – online mode.* The proposed framework uses *DENFIS* in online mode. *DENFIS* is a dynamic inference system which is capable of creating or updating a fuzzy rule while the system is running. The output depends on the most active fuzzy rules at any given time. *DENFIS* depends on the fuzzy rule set that is chosen automatically. One of the most powerful features related to *DENFIS* is the capability to generate new rules before or during the learning process. *DENFIS* can also extract rules during or after the learning process.

ECM is the most important parts of the *DENFIS* algorithm, as the antecedent of the fuzzy rules based on the cluster centers. When once the clusters are fully optimized, a fuzzy inference system is then developed.

DENFIS uses the Takagi-Sugeno fuzzy inference engine with triangular membership functions (*MF*) composed of m fuzzy rules. However, the new version of the DENFIS-online mode works only based on the Gaussian membership function [12,38].

We suggest to adapt the new version of the DENFIS-online mode based on Gaussian function because an extension can be made to the DyNFIS-offline mode also because the Gaussian *MF* coverage of the problem space expands a long way as the degree of membership decreases steadily. The bell curve from the peak at different rate depends on the parameter of the function and does not reach zero, as shown in Figure 3.

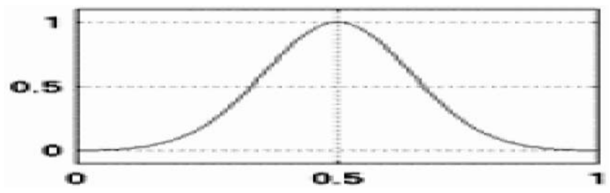


FIGURE 3. Gaussian membership function

In DENFIS, the rules are defined by the Gaussian-type membership function, and it has two parameters, as expressed in Equation (13).

The fuzzy rule sets are defined by the following Gaussian-type membership function (*MF*):

$$\text{Gaussian MF} = \alpha \exp\left(\frac{-(z-m)^2}{2\sigma^2}\right) \tag{13}$$

where z is the input vector, m is the center of the Gaussian function, and σ is the width (cluster radius) of the Gaussian function, listed in the order of vector $[\sigma, m]$ [14].

When the model is given as an input-output pair (z_i, y_i), DENFIS is used to distinguish a phishing from a ham e-mail.

However, to enhance the rules generated by DENFIS, *we suggest* the need for the DyNFIS offline mode while the system is working in the online mode to enhance the rules based on ECMc to make the generated rules more fit and accurate for the classification input samples without stopping the system. This process should be done because DENFIS and DyNFIS have the same format or rules based on the Gaussian membership function rules.

3.4.2. *Enhancement rules based on DyNFIS-offline mode.* The proposed framework uses the DyNFIS-based on ECMc to enhance the rule in offline mode while the system is working in online mode to build life-long learning framework. The data derived from the offline part of ECM *is fed to* DyNFIS. However, DyNFIS works based on the Gaussian membership function in offline mode because it is more accurate and more suitable for real-world applications. DyNFIS will allow the antecedents and consequences to be optimized using back-propagation and minimize the error in the active rules based on minimizing the objective function [14].

Our proposed framework takes advantage of the DENFIS and DyNFIS-Gaussian MF to minimize the error in the active rules for building the framework. It is most suited to higher level of noise data and has the ability to detect and predict zero-day phishing e-mail rapidly with high accuracy in the classification and prediction processes.

However, to arrange the relationship between the two algorithms (DENFIS and DyNFIS), *we suggest* the profile management framework.

3.4.3. *Profile management framework.* Profile management frameworks play two main rules in third and fourth phase respectively in PDENFF. The first contribution of Profile management framework occours in the third level which is implemented by capturing input samples form ECM-Online then enhancing the place of input vector by enhancing the postion of clusters centers is based on ECMc-offline. This process to optimize the clusters is based on decreasing the objective function. This condition happens by default for every 800 samples of e-mails because the length of time the mail server that will receive a new message is not known. Therefore, this process is controlled based on the number of e-mails and not based on time.

The second contribution of profile management framework occours in the fourth level which is implement by 2 steps

1. capture the rule profiles created in DyNFIS-offline mode
2. insert capturing rule profile to DENFIS-online mode while the system is working.

This process appears as parallel system to enhance the repository of rules based on ECMc. The profile management framework will load the enhanced rule to DENFIS, which can be automatically adaptive the rules without duplication and has ability to select the best rule in classification process automatically because it has the same format of rules, which depends on the same type of Gaussian membership function.

For any new input data, this process will make PDENFF work with unlimited life-long training system, using footprint memory, because the system will use the enhanced rules in the online mode. Afterward, in our framework, if a new rule is created, a new feature of phishing e-mail will occur. Therefore, the system can work with the noise data in a high level of accuracy in the classification process, as proven in the implementation process and test results. See Figure 4.

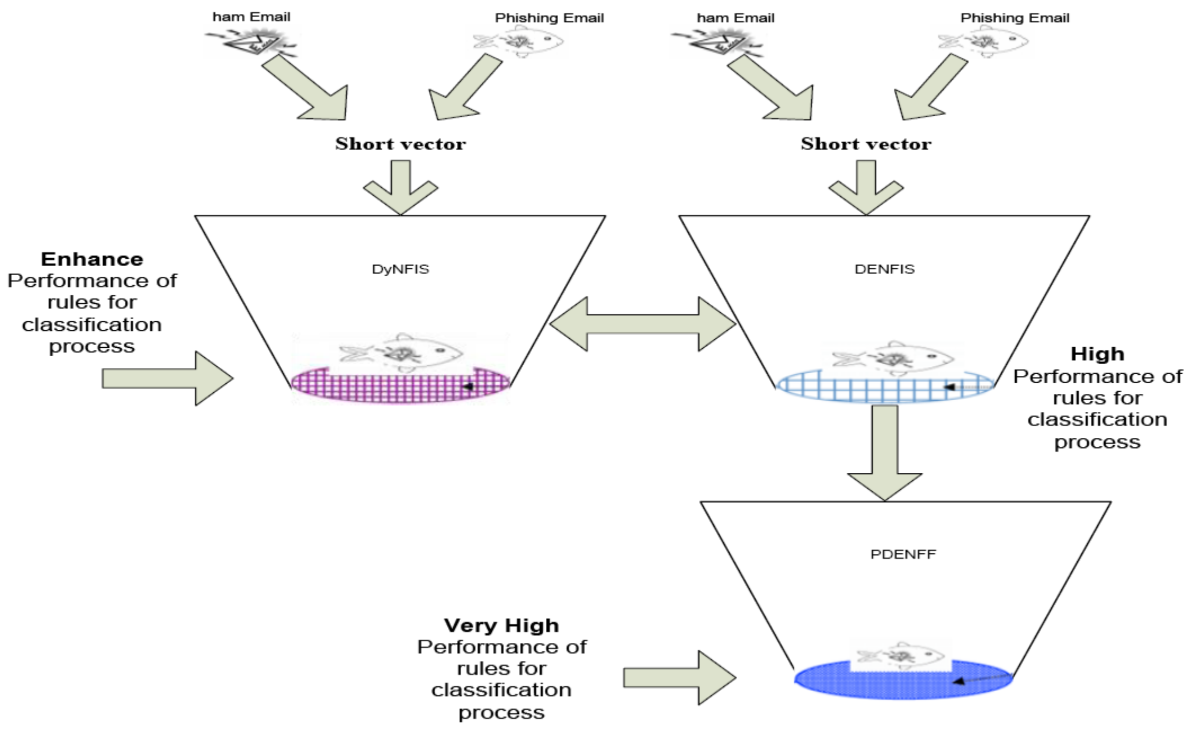

ham Email
Phishing Email
ham Email
Phishing Email
Short vector
Short vector
Enhance Performance of rules for classification process
DyNFIS
DENFIS
High Performance of rules for classification process
PDENFF
Very High Performance of rules for classification process

Figure 4. Methodology for detection phishing attack based on PDENFF

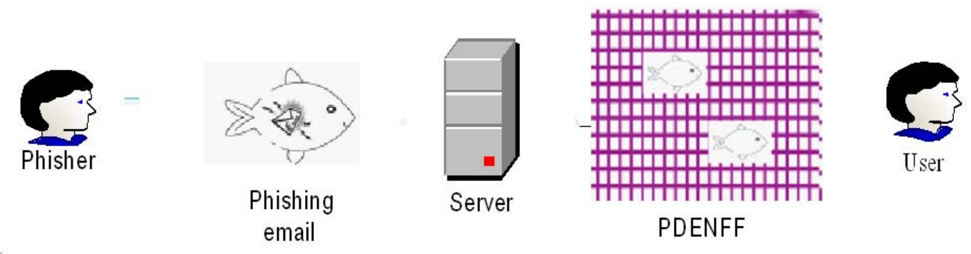


FIGURE 5. Place of PDENFF in computer network

Figure 4 shows how the effective of rule by DENFIS for classification process will be increased by taking the advantage of rule generated by DyNFIS to make very high level of performance in classification process inplemented by PDENFF.

A schematic representation PDENFF is shown in Figure 5.

Figure 5 shows that PDENFF stays between the server and user to stop the phisher attack before it gets to the user, which makes the process work in high level of security and fully controlled by PDENFF.

4. Implementation and Test Results.

Our framework was tested on an Intel Duo Core E4500 system with 2 GB RAM and a Windows operating system. A flexible pre-processing and feature extraction system was implemented using Python and Java programming language for the purpose of this investigation. MATLAB version 7.10 was used for the connectionist framework of the system engine and for computation and analysis. Three experiments were constructed to prove our objectives, which are explained in detail after the discussion of the dataset used in our framework.

4.1. Dataset.

Two datasets were used for the assessment of the proposed framework in our experiments. The first one was the well-known dataset from most authors in this area, consisting of 4,000 phishing messages received from November 2004 to August 2007 provided by the Monkey website [35] and 4,000 ham e-mails from the SpamAssassin project [36]. The second one was a collection of sample sets of 300 new phishing and 2,000 ham e-mails from the mail server in our NAV6 center in the period from July 2011 to April 2012. We used the first dataset in the three experiments, whereas the second dataset was used in the third experiment only. Our experiments worked based on 10-fold cross validation using random method with min-max (linear normalization) for all input to confine the input in the range $[0, 1]$ and to preserve exactly all relationships of the data values. For each fold of the experiments, 80% of the dataset was trained, whereas 20% of the dataset was tested.

We used many measurements to make our experiments clear. This measurements included true positive (TP), the number of phishing e-mails correctly classified as phishing, true negative (TN), the number of ham e-mails correctly classified as ham, FP, the number of ham e-mails correctly classified as ham, and FN, the number of phishing e-mails wrongly classified as ham. The other measurements are presented in Table 3, and the full details of each experiment are shown below.

TABLE 3. Equations used to calculate recall/sensitivity, precision, F-measure, and overall accuracy

Recall/Sensitivity	precision	F-Measure	Over all Accuracy
$= \frac{\lvert TP \rvert}{\lvert TP \rvert + \lvert FN \rvert}$	$= \frac{\lvert TP \rvert}{\lvert TP \rvert + \lvert FP \rvert}$	$= \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$	$= \frac{\lvert TP \rvert + \lvert TN \rvert}{\lvert TP \rvert + \lvert TN \rvert + \lvert FP \rvert + \lvert FN \rvert}$

4.2. **First experiment.** This experiment used the first dataset, which consists of 8,000 e-mails (ham = 4,000 and phish = 4,000). The main objective of this experiment is to show the effect of the *short vector* based on the crisp input value compared with the *long vector* based on binary input value in our framework. PDENFF depends on the distance threshold (D_{thr}) in ECM; thus, we designed the experiments based on 10-fold cross validation for each D_{thr}. We selected five D_{thr} in the range $[0.1, 0.5]$ because they showed the best result in our framework. The testing results based on the *short vector* are shown in Table 4, and the average of the testing results for each D_{thr} is shown in Figure 6.

Table 4. Performance measurement results based on distance threshold (D_{thr}) – *short vector*

D_{thr}	FP	FN	Accuracy	precision	Recall/ Sensitivity	Time consuming (s)	Number of fuzzy rules
0.1	0.03 ± 0.02	0.04 ± 0.2	0.98 ± 0.01	0.97 ± 0.02	0.97 ± 0.01	4 ± 2	48 ± 2
0.2	0.04 ± 0.01	0.03 ± 0.2	0.97 ± 0.01	0.95 ± 0.02	0.96 ± 0.02	4 ± 1	21 ± 3
0.3	**0.02 ± 0.01**	**0.03 ± 0.01**	**0.99 ± 0.01**	**0.98 ± 0.01**	**0.97 ± 0.02**	**3 ± 1**	**9 ± 3**
0.4	0.04 ± 0.02	0.04 ± 0.02	0.96 ± 0.02	0.90 ± 0.03	0.95 ± 0.03	4 ± 1	14 ± 1
0.5	0.02 ± 0.01	0.05 ± 0.01	0.94 ± 0.01	0.95 ± 0.03	0.95 ± 0.01	4 ± 1	6 ± 1

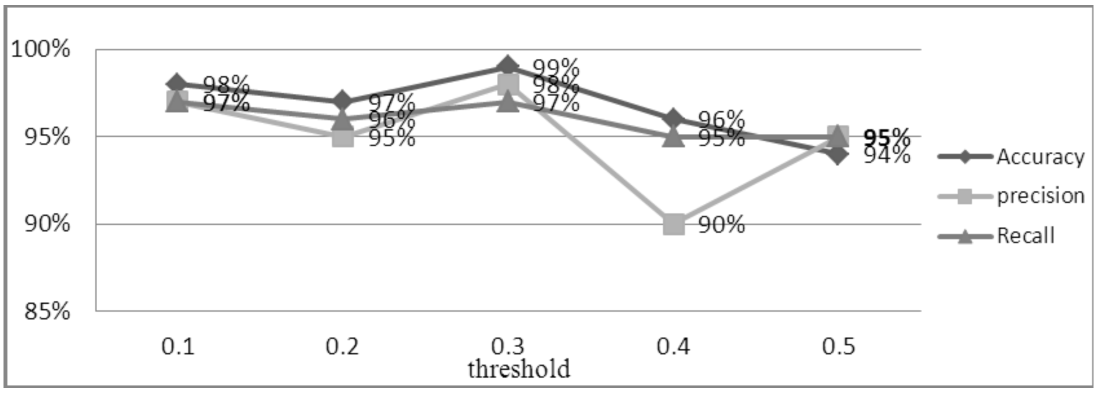


Figure 6. PDENFF performance average based on distance threshold – *short vector*

Table 4 and Figure 6 show that the performance of PDENFF based on *short vector* had very good results. However, the optimal threshold was $D_{thr} = 0.3$, which showed the best result compared with all other thresholds. The overall accuracy reached up to 99%. Thus, this threshold was used in experiments 2 and 3 to prove the other objectives in our framework. To show the difference in working with the *short vector* from the *long vector*, the same experiments were constructed using the *long vector* features. The results are shown in Table 5 and Figure 7.

Table 5 and Figure 7 show that the performance of PDENFF based on *long vector* had very good result but is less than that using the *short vector*. In addition, the optimal threshold here is $D_{thr} = 0.3$. However, the main difference between using *long vector* and *short vector* in our framework lies in the number of rules generated, the time consumed in the classification process after extracting the e-mails (shown in Figures 8 and 9, respectively), and the complexity of the rule sets.

Figures 8 and 9 respectively show that the average time consumed and the fuzzy rule generated by the *long vector* were about 10 times more than those of the *short vector*, proving that the general performance of the *short vector* is better than that of the *long*

TABLE 5. Performance measurement results based on distance threshold (D_{thr}) – *long vector*

D_{thr}	FP	FN	Accuracy	precision	Recall/ Sensitivity	Time consuming (s)	Number of fuzzy rules
0.1	0.02 ± 0.01	0.04 ± 0.01	0.96 ± 0.01	0.97 ± 0.01	0.96 ± 0.01	40 ± 10	637 ± 50
0.2	0.04 ± 0.01	0.02 ± 0.01	0.97 ± 0.01	0.96 ± 0.02	0.95 ± 0.01	21 ± 5	280 ± 30
0.3	**0.02 ± 0.01**	**0.03 ± 0.02**	**0.97 ± 0.01**	**0.98 ± 0.01**	**0.97 ± 0.01**	**15 ± 3**	**87 ± 20**
0.4	0.05 ± 0.01	0.03 ± 0.01	0.96 ± 0.01	0.96 ± 0.02	0.95 ± 0.03	13 ± 2	25 ± 5
0.5	0.04 ± 0.01	0.04 ± 0.01	0.96 ± 0.02	0.98 ± 0.01	0.95 ± 0.01	8 ± 2	20 ± 2

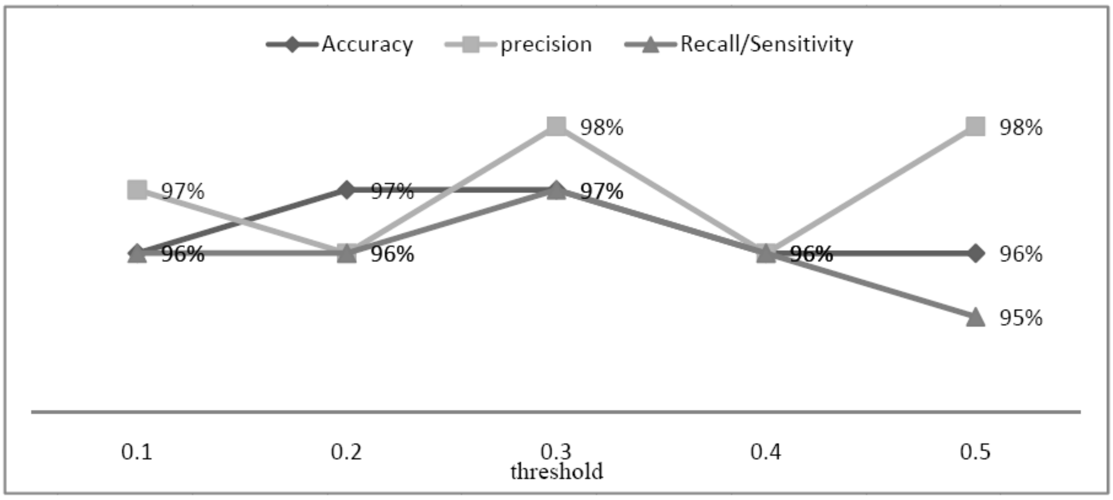
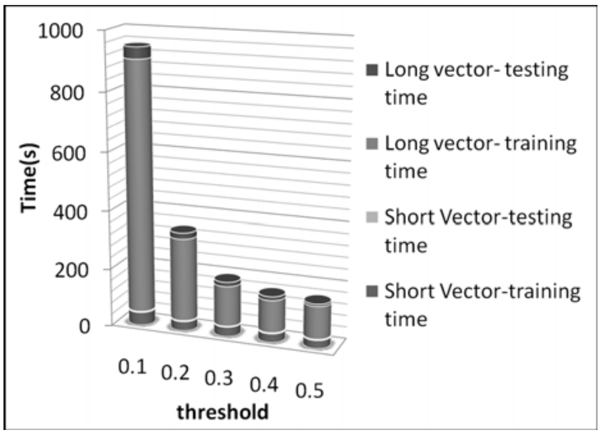


FIGURE 7. PDENFF – performance average based on distance threshold – *long vector*



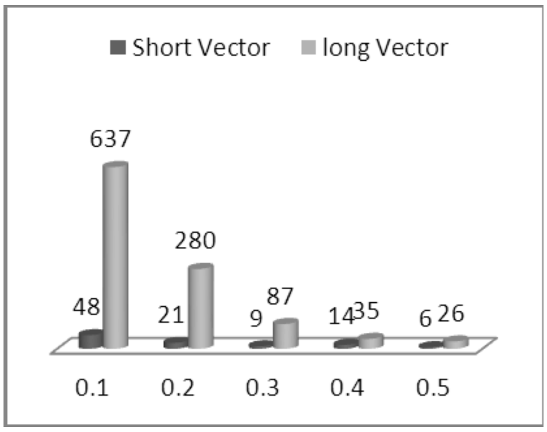


FIGURE 8. Average time consumed in the testing and training phase

FIGURE 9. Average number of rules – training phase

vector. The complexity of rules generated in the *long vector* is more than that of the *short vector* because the number of Gaussian membership function in each rule in the *short vector* is four, whereas the number of Gaussian membership function in the *long vector* is 21, for each rule. This result increases the complexity of the rule generation with time; thus, experiments 2 and 3 were completed using PDENFF based on *short vector*.

4.3. **Second experiment – comparison of the PDENFF performance with other classifier algorithms.** This experiment used the same dataset employed in the first experiment. The objective of this experiment is to compare PDENFF with other classification algorithms currently used for phishing e-mail detection. Therefore, we used the *short vector* with six classifiers in our framework. We used the random 10-fold cross validation to evaluate the performance of these algorithms. The performance measurement results for all folds are shown in Table 6, whereas Figure 10 shows the performance average of the classification algorithms.

Table 6 and Figure 10 show that PDENFF based on DENFIS and DyNFIS has the best performance compared with other classifier algorithms, at an accuracy of approximately 99% and approximately 96% for DENFIS. SVM and NNet (MLP), usually use to detect phishing e-mail, had an overall accuracy of approximately 93%, and the k-means had the worst level of performance because it depended on unsupervised learning only. The performance of the Bayesian (Naïve base) was approximately 90%, usually use in the toolbar algorithms. The performance of the random forest algorithms was approximately 91% using the PILFER model. These results encouraged us to present the performance

Table 6. Performance measurement result of the classification algorithms

classifiers	TP	TN	Accuracy	precision	Recall	F-Measure	Online/ Offline
PDENFF (DENFIS & DyNFIS)$_{(Dthr=3)}$	**0.97 ± 0.02**	**0.98 ± 0.01**	**0.99 ± 0.01**	**0.98 ± 0.01**	**0.97 ± 0.02**	**0.97 ± 0.02**	**Online enhanced by Offline**
DENFIS$_{(Dthr=3)}$	0.95 ± 0.02	0.96 ± 0.02	0.96 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	Online or offline
SVM	0.94 ± 0.02	0.93 ± 0.02	0.93 ± 0.01	0.94 ± 0.01	0.93 ± 0.01	0.93 ± 0.01	Online or offline
NNet (MLP)	0.93 ± 0.01	0.91 ± 0.02	0.92 ± 0.03	0.93 ± 0.01	0.92 ± 0.02	0.92 ± 0.02	Online or offline
k-means	0.89 ± 0.01	0.90 ± 0.02	0.90 ± 0.01	0.88 ± 0.02	0.90 ± 0.01	0.89 ± 0.02	Offline-only
Bayesian (Naïve-base)	0.91 ± 0.01	0.90 ± 0.01	0.91 ± 0.01	0.90 ± 0.01	0.92 ± 0.02	0.91 ± 0.03	Online-only
Random forest	0.92 ± 0.01	0.93 ± 0.02	0.91 ± 0.02	0.91 ± 0.01	0.90 ± 0.02	0.90 ± 0.02	Offline or offline

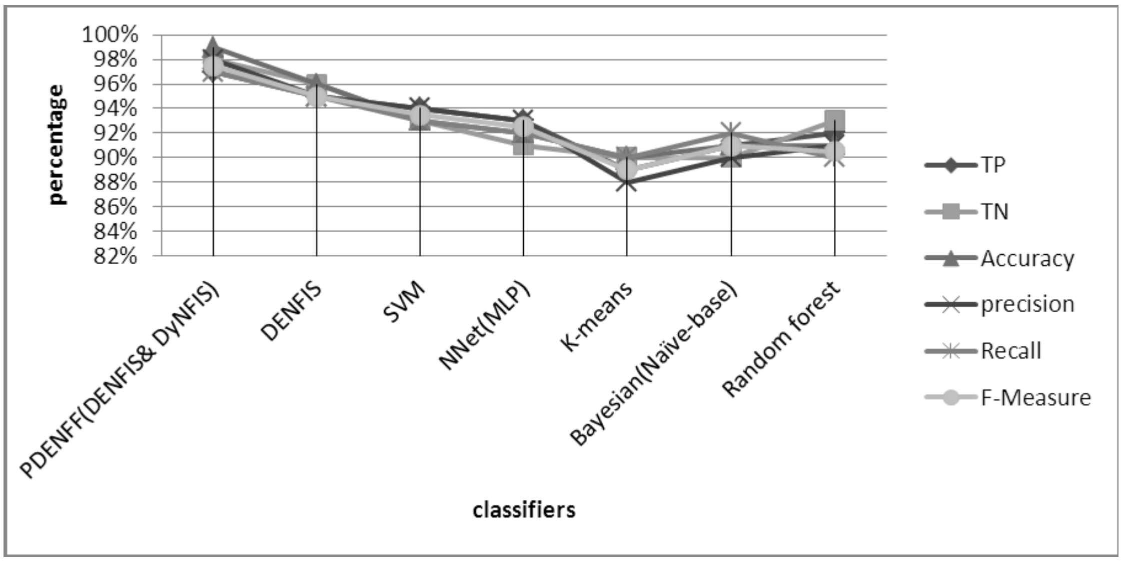


Figure 10. Performance average of the classification algorithms

of our framework to detect zero-day phishing e-mail attack compared with the other algorithms in experiment 3.

4.4. **Third experiment – detecting zero-day attack.** We present the process of how our framework was able to solve the phishing problem using the first dataset used in the first experiment during the training phase and using the second dataset collected from our center (NAV6) as a test dataset during the testing phase, based on 10-fold cross validation. We need to have more different between the learning and testing datasets to show the average performance of the classification algorithms of our framework based on DENFIS and DyNFIS compared with other classifier algorithms in these experiments (Figure 11).

Figure 11 shows that the average performance of 98% of our framework based on DENFIS and DyNFIS is still the best result compared with other classification algorithms, followed by the approximately 95% of DENFIS. The worst result from the NNet (MLP) and k-means algorithms was approximately 85%, indicating an improvement of PDENFF based on DENFIS and DyNFIS in detecting zero-day phishing e-mail attacks from 3% to 13%, capacity to work with noise data, and the ability of life-long learning using classification data in the online mode.

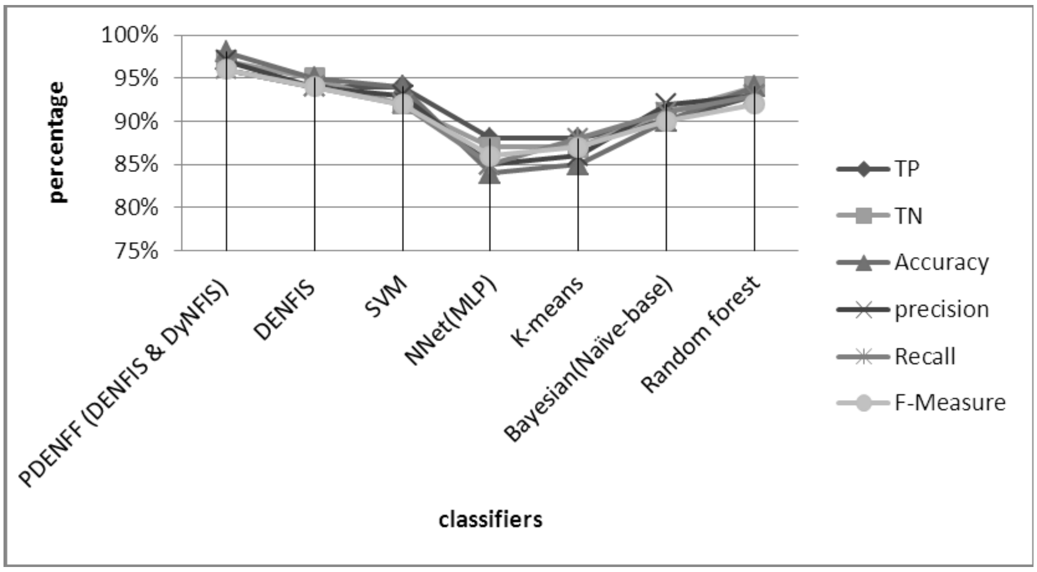


FIGURE 11. Average performance of the classification algorithms – "zero day" attack

5. **Conclusions and Future Work.** The proposed framework improved the level of performance in detecting and predicting unknown zero-day phishing e-mails between 3% and 13%. PDENFF distinguished phishing e-mails from ham e-mails in an online mode based on new rules, classes, or features to enhance learning using ECOS. Therefore, the present work is an important step in the use of ECOS for phishing e-mail detection.

A new technique was used for the extraction of features based on the assumption that all features have a binary value of either zero or one. It built *short vector* of values based on the IGR. The proposed approach used a new incremental clustering algorithm modified for this purpose, which depends on the maximum distance (*MaxDist*) between the input data and the cluster center for classification and for developing new rules in DENFIS *online mode* and enhancing the rules based on *DyNFIS offline mode while the system is* *working*. Our framework depended on the Takagi-Sugeno fuzzy model generation and Gaussian membership function.

The experiments proved that the proposed framework has better performance, including the enhancement of time consumption by 10 times and decreasing the number of generated rules, enhancing the overall accuracy of FP, FN, F-measure, recall, precision, and others measurement compared with other learning algorithms that used existing solution. Therefore, the proposed approach has a great potential for real-world applications. For future studies, we will try to build a system that takes advantage of DENFIS based on triangular function and of DyNFIS based on Gaussian function to enhance the fuzzy rule for building a system capable of high-speed work and has a high performance for real-world implementation.

Acknowledgment. This work was supported by the National Advanced IPv6 Centre of Excellence (NAV6), Universiti Sains Malaysia.

REFERENCES

[1] Anti-Phishing-Work-Group, *Phishing Activity Trends Report*, http://www.antiphishing.org/reports /apwg_trends_report_h1_2011.pdf, 2011.

[2] Gartner, *Gartner Survey Shows Phishing Attacks Escalated in 2007; More than $3 Billion, Lost to These Attacks*, http://www.gartner.com/it/page.jsp?id=565125, 2007.

[3] Symantec, *MessageLabs Intelligence: 2010 Annual Security Report*, www.messagelabs.com/mlire port/MessageLabsIntelligence_2010_Annual_Report_FINAL.pdf, 2010.

[4] C. E. Drake, J. J. Oliver and E. J. Koontz, Anatomy of a phishing email, *Proc. of the 1st Conference on Email and Anti-spam*, Mountain View, CA, USA, pp.2-3, 2004.

[5] D. L. Cook, V. K. Gurbani and M. Daniluk, Phishwish: A simple and stateless phishing filter, *Security and Communication Networks*, vol.2, no.1, pp.29-43, 2009.

[6] M. Blasi, *Techniques for Detecting Zero Day Phishing Websites*, Master Thesis, Iowa State University, 2009.

[7] M. Jakobsson and S. Myers, Microsoft's anti-phishing technologies and tactics, in *Phishing and Countermeasures*, 2007.

[8] S. Abu-Nimeh, D. Nappa, X. Wang and S. Nair, A comparison of machine learning techniques for phishing detection, *Proc. of the eCrime Researchers Summit*, Pittsburgh, PA, vol.2837, pp.60-69, 2007.

[9] R. Dazeley, J. Yearwood, B.-H. Kang and A. Kelarev, Consensus clustering and supervised classification for proling phishing emails in Internet commerce security, *Knowledge Management and Acquisition for Smart Systems and Services*, vol.6232, pp.235-246, 2010.

[10] N. Kasabov, Z. S. H. Chan, Q. Song and D. Greer, *Evolving Connectionist Systems with Evolutionary Self-Optimisation*, vol.173, no.4, 2005.

[11] P. Angelov, D. P. Filev and N. Kasabov, *Evolving Intelligent Systems: Methodology and Applications*, Wiley-IEEE Press, 2010.

[12] N. K. Kasabov and Q. Song, DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction, *IEEE Transactions on Fuzzy Systems*, vol.10, no.2, pp.144-154, 2002.

[13] Q. Song and N. Kasabov, Dynamic evolving neuro-fuzzy inference system (DENFIS): On-line learning and application for time-series prediction, *IEEE Transactions on Fuzzy Systems*, vol.10, pp.144-154, 2000.

[14] Y. Hwang and C. Song, Dynamic neural fuzzy inference system, *Advances in Neuro-Information Processing*, vol.5506, pp.1245-1250, 2009.

[15] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh and J. C. Mitchell, Client-side defense against web-based identity theft, *Proc. of the 11th Annual Network and Distributed System Security Symposium*, San Diego, CA, 2004.

[16] E. Kirda and C. Kruegel, Protecting users against phishing attacks with AntiPhish, *Proc. of the 29th Annual International Computer Software and Applications Conference*, pp.517-524, 2005.

[17] Netcraft, *Netcraft Toolbar*, http://toolbar.netcraft.com/, 2006.

[18] Y. Zhang, S. Egelman, L. Cranor and J. Hong, Phinding phish: Evaluating anti-phishing tools, *Proc. of the 14th Annual Network & Distributed System Security Symposium*, San Diego, CA, 2007.
[19] I. Fette, N. Sadeh and A. Tomasic, Learning to detect phishing emails, *Proc. of the 16th International World Wide Web Conference*, New York, NY, USA, pp.649-656, 2007.
[20] M. R. Islam, J. Abawajy and M. Warren, Multi-tier phishing email classification with an impact of classifier rescheduling, *International Symposium on Pervasive Systems, Algorithms, and Networks*, Kaohsiung, Taiwan, pp.789-793, 2009.
[21] A. Liaw and M. Wiener, Classification and regression by randomForest, *R News*, vol.2, no.3, pp.18-22, 2002.
[22] O. Chapelle, Training a support vector machine in the primal, *Neural Computation*, vol.19, no.5, pp.1155-1178, 2007.
[23] R. Basnet, S. Mukkamala and A. H. Sung, Detection of phishing attacks: A machine learning approach, *Studies in Fuzziness and Soft Computing*, vol.226, pp.373-383, 2008.
[24] A. Almomani, T. C. Wan and K. Al-Saed, An online model on evolving phishing e-mail detection and classification method, *Journal of Applied Science*, vol.11, pp.3301-3307, 2011.
[25] L. Benuskova and N. Kasabov, Evolving connectionist systems (ECOS), in *Computational Neurogenetic Modeling*, Springer, US, 2007.
[26] T. Takagi and M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, *IEEE Transactions on Systems Man and Cybernetics*, vol.15, pp.116-132, 1985.
[27] Q. Song and N. Kasabov, ECM – A novel on-line, evolving clustering method and its applications, *Proc. of the 5th Biannual Conference on Artificial Neural Networks and Expert Systems*, New Zealand, pp.87-92, 2001.
[28] I. Fette et al., Learning to detect phishing emails, *Proc. of the 16th International World Wide Web Conference*, Banff, Alberta, Canada, pp.649-656, 2007.
[29] W. N. Gansterer and D. Pölz, E-mail classification for phishing defense, *Proc. of the 31st European Conference on IR Research on Advances in Information Retrieval*, Toulouse, France, pp.449-460, 2009.
[30] *SpamAssassin for Win32*, http://sourceforge.net/projects/sawin32/, 2010.
[31] F. Toolan and J. Carthy, Feature selection for spam and phishing detection, *eCrime Researchers Summit (eCrime)*, Dallas, TX, pp.1-12, 2010.
[32] A. Bergholz, J. De Beer, S. Glahn, M. F. Moens, G. Paa*β* and S. Strobel, New filtering approaches for phishing email, *Journal of Computer Security*, vol.18, no.1, pp.7-35, 2010.
[33] M. Khonji, A. Jones and Y. Iraqi, A novel phishing classification based on URL features, *GCC Conference and Exhibition, IEEE*, pp.221-224, 2011.
[34] T. Mori, Information gain ratio as term weight: The case of summarization of IR results, *Proc. of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan, vol.1, pp.688-694, 2002.
[35] A. J. Nazario, *Phishingcorpus Homepage*, http://monkey.org/jose/wiki/doku.php?id=PhishingCorpus, 2006.
[36] *Apache Software Foundation*, Spamassassin public corpus, http://spamassassin.apache.org/publiccorpus/, 2006.
[37] M. F. Pasha, R. Budiarto, M. Syukur and M. Yamada, EFIS: Evolvable-neural-based fuzzy inference system and its application for adaptive network anomaly detection, *Advances in Machine Learning and Cybernetics*, vol.3930, pp.662-671, 2006.
[38] S. Dehuri, C. Mohapatra, A. Ghosh and R. Mall, A comparative study of clustering algorithms, *Information Technology Journal*, pp.551-559, 2006.
[39] J. Yearwood, M. Mammadov and A. Banerjee, Profiling phishing emails based on hyperlink information, *International Conference on Advances in Social Networks Analysis and Mining*, Odense, Denmark, pp.120-127, 2010.