Decision Support Systems 107 (2018) 88–102

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

Detection of online phishing email using dynamic evolving neural network based on reinforcement learning

Sami Smadi, Nauman Aslam*, Li Zhang

Department of Computer and Information Science, Northumbria University, UK

ARTICLE INFO

Article history:
Received 12 April 2017
Received in revised form 18 October 2017
Accepted 5 January 2018
Available online xxxx

Keywords:
Online phishing email detection
Reinforcement learning
Neural network

ABSTRACT

Despite state-of-the-art solutions to detect phishing attacks, there is still a lack of accuracy for the detection systems in the online mode which is leading to loopholes in web-based transactions. In this research, a novel framework is proposed which combines a neural network with reinforcement learning to detect phishing attacks in the online mode for the first time. The proposed model has the ability to adapt itself to produce a new phishing email detection system that reflects changes in newly explored behaviours, which is accomplished by adopting the idea of reinforcement learning to enhance the system dynamically over time. The proposed model solves the problem of limited dataset by automatically adding more emails to the offline dataset in the online mode. A novel algorithm is proposed to explore any new phishing behaviours in the new dataset. Through rigorous testing using the well-known data sets, we demonstrate that the proposed technique can handle zero-day phishing attacks with high performance levels achieving high accuracy, TPR, and TNR at 98.63%, 99.07%, and 98.19% respectively. In addition, it shows low FPR and FNR, at 1.81% and 0.93% respectively. Comparison with other similar techniques on the same dataset shows that the proposed model outperforms the existing methods.

1. Introduction

Stealing a person's identity is one of the most popular cybercrime activities. According to the Federal Trade Commission in 2015 [1], identity theft was ranked second with 16% of all customer complaints. The most common way to steal an online consumer's personal identity is called phishing, which can be defined as "a fraudulent attempt, usually made through email, to steal your personal information" [2]. Phishing started to appear in the mid-1990s, and it has become an important issue in online transactions since that time.

The total number of unique phishing attacks reports collected by anti-phishing working group (APWG) from January to September 2015 was 1,033,698, which is twice the number of reports for the same period in 2014 [3]. From the increase in registered phishing attacks we can conclude that the used solutions to handle phishing attacks failed to handle attacks lunched from hosts that are not blacklisted or using techniques that evade known approaches in phishing detection. These kind of attacks are known as zero-day phishing attacks [4,5].

Phishers use many different techniques to initiate phishing attacks, the main methods used nowadays are email [6], SMS (smishing) [7], Social network/media [8], VoIP (voice phishing) [9], instant messaging [10], search engines and malicious websites [5,11]. Phishers always modify their methods to use any communication method available to reach their victims. Previous studies have shown that 65% of phishing attacks use email as the main communication channel to lure online customers [12,13]. In this paper, the detection method is built to detect phishing attacks at the email level, since this is the most popular method used to spread these attacks. Nevertheless, the same text used in phishing emails are also sent via other communication channels to reach online customers. Moreover, the proposed model can be used to detect phishing attacks at any other communication channel as long as the format used is text or HTML. Thus, the proposed model is a significant research contribution to discover phishing attacks.

A new kind of phishing attack was launched in the late 2007 which is called spear phishing [14]. This is a highly targeted phishing attack. Rather than sending phishing emails to anyone in the usual way, the phisher sends spoofed emails to consumers that appear to originate from somebody they know. For example, phishers can send emails pretending that an organization's manager is asking employees to

* Corresponding author.
E-mail addresses: sami.smadi@northumbria.ac.uk (S. Smadi), nauman.aslam@northumbria.ac.uk (N. Aslam), li.zhang@northumbria.ac.uk (L. Zhang).

https://doi.org/10.1016/j.dss.2018.01.001
0167-9236/

update their data (like login information) or pretending that the email has come from a friend on social networks. The extent of spear phishing increased dramatically between 2009 to 2011, and it has been responsible for some high-profile corporate data breaches [15]. This kind of attack is successful because the phisher can collect information about consumers easily from the Internet by the basic mining of company websites.

Phishing problems affect the electronic commerce because online customers trust the Internet environment less [16,17]. Phishers use techniques which evolve to lure online customers, creating new phishing websites and spreading emails that try to convince Internet users to follow fraudulent links to access their websites. Phishing emails employ sophisticated techniques which direct the online customers to open a new web page, which has not yet been added to the black-list. A phishing attack that uses these new types of techniques is called a zero-day attack [4].

In summary, phishers continue to improve their tactics using new techniques, targeting specific groups, and using alternative channels to spread their attacks. The present research focuses on detecting phishing attacks at email level, since most phishers use this channel to initiate their attacks.

The novel contribution of this paper is the ability of the proposed Phishing Email Detection System (PEDS) to adapt itself to reflect changes in the environment. The novelty claim stems from the fact that we introduced a new approach that used Reinforcement Learning. To the best of our knowledge, this is the first work that has applied RL based methods towards this application. A new algorithm called Feature Evaluation and Reduction (FEaR) is developed to explore the new behaviour as well as to rank a selected list of features. In the field of online phishing email detection, the number of important feature is always changing. The proposed algorithm is dynamically changing the number of important features and extract them from next email. A neural network (NN) is used as the core of the classification model, and a novel algorithm called Dynamic Evolving Neural Network using Reinforcement learning (DENNuRL) is developed to allow the NN to evolve dynamically and build the best NN able to solve the desired problem. This paper is organised as follows. Relevant work on protection techniques is discussed and their advantages and disadvantages are presented in Section 2. The proposed approach is shown in Section 3.1, inducing details of preprocessing, experimental design, the system model, DENNuRL, and RL-Agent. The results and discussion are shown in Section 4. Finally, the conclusions and suggestion for future work are presented in Section 5.

2. Relevant work

To date, many techniques have been developed to fight phishing attacks, including legislation and technology developed to protect online customers. The solutions developed can be categorised as technical or non-technical. Non-technical anti-phishing solutions include awareness and training programmes to teach online consumers how to recognise phishing emails and websites [18–20]. [19], the authors focus on phishing susceptibility of individuals and cognitive-information processing. [20] focus on how individuals behave or deal with the e-mails, a questionnaire has been designed and they collect behavioural data through logs. The non-technical solutions are very important, but they need to keep teaching Internet users about the new kinds of attack, and users need to read a lot of information. Therefore, a solution is needed that gives protection to online consumers without their requiring intervention or action. The non-technical solutions are outside of the scope of this study. The second type of solution includes technical anti-phishing such as developing a security system that protects online customers without the need for their intervention.

Technical phishing email detection has been investigated in many studies, a few of which have built systems to handle zero-day phishing attacks. Black- and white-list techniques are used to prevent phishing attacks by maintaining a database of both trusted and phishing websites. White-lists are the lists of trusted websites that an Internet user visits regularly. This technique allows the user to navigate only to a website which has been considered previously to be a legitimate internet site, which makes this method very efficient in handling zero-day phishing attacks. It also produces no false positives. The main disadvantage of using white-lists is that it is hard to manage previously used websites that users will navigate to in the future. When a user chooses to open a legitimate internet site, and this website is not listed in the white-list, the system will consider it to be a phishing website, which increases the false negative rate. This means that white-lists is not very popular. Li et al. [21] proposed a phishing prevention tool called the IEPlug which is based on a white-list. IEPlug inspects a website visited by the user and if a phishing website is detected the certificate authority dialog is shown and a warning is given to the user. The main disadvantage of using this technique is that the white-list need to be maintained manually and the visual effects may disturb users. In a black-list phishing prevention approach, the requested URL is compared with a predefined phishing black-list. Black-List phishing prevention is a very well-known technique to handle phishing attacks. Most famous web browsers such as Google Chrome and Internet Explorer use black-lists for phishing prevention. If the Internet user tries to visit a fraudulent website which is already saved in the black-list, the web browser denies access or warns the user about that website. Furthermore, black-lists have very low false-positive rates, which leads many users to prefer than heuristic methods. The main reasons for the extensive use of the black-lists is their low false positive rate and also they are very simple to design and implement [22]. Sheng et al. [23] conducted a study to evaluate the effectiveness of black-lists in detecting phishing attacks. Some fresh phishes were used in the evaluation process, where the phish life was less than 30 min. This study showed that tools based on the black-lists are ineffective in the prevention of zero-day phishing attacks, more than 80% of which were not detected. The main disadvantage of using a black-list is that it will not contain all phishing websites at any given moment, because the lifetime of these websites is so small. In the time between initiating an attack and its detection the phisher may deceive many customers and then create another fraud website [24].

A multi-tier classification method was proposed for phishing email filtering [25]. They also proposed an innovative method for extracting the features of phishing emails based on a weighting of message content and message header. A multiple classification algorithm is used including SVM, AdaBoost, and Naïve Bayes. They have divided the email classification on three tiers, and they use 21 different features. Each email is classified in the first tier and reclassified in the next tier using another classification algorithm, so that if the email is correctly classified it will be passed on to the analyser and subsequently sent to the appropriate folder. Otherwise, it will be reclassified using the classification algorithm in tier three, and then sent to the right mailboxes folder. The proposed model achieved 97% accuracy, and the main weakness is that the authors did not take into consideration the computing overhead and performance issues of using multiple classifiers in multiple tiers. Furthermore, their model is fixed and cannot be automatically adapted to handle new phishing attack behaviour. Finally, the overall metrics also needed to be enhanced.

Almomani et al. [4] proposed a novel framework that classifies email in the online mode into phishing and ham email. The proposed model is called the phishing dynamic evolving neural fuzzy framework (PDENFF). Here, a total of 21 features were extracted from each email which are grouped into four groups: spam, body, URL and header features. The generated set of features is called the short vector. In the next step, the framework generates basic rules, and then a dynamic evolving neural fuzzy inference system (DENFIS) is used to produce

the fuzzy rules, which the system is able to add, delete or update in the online mode. Two datasets were used for training and assessing the proposed model. In the first dataset, 8000 phishing and ham emails were collected from Monkey.org [26] and the SpamAssasin [27] corpora. In the second dataset, 2300 emails were collected from a mail server at the NAV6 centre at the University of Sains, Malaysia, which contain 300 of phishing, and 2000 ham emails. An experiment was conducted to show the merits of the proposed model, and the first two experiments demonstrated the benefits of using the short and long vectors and compared the output of the PDENFF with other classification models for the offline dataset. The third experiment was applied to zero-day phishing attacks and the model tested with a third dataset achieved a performance level of 98%. The main weaknesses of the proposed model were, firstly, that the dataset used for assessing the model in the online mode was very small, consisting of 300 emails. Secondly, the authors did not show how the system evolved in the online mode.

A robust server-side model to handle phishing email detection has been proposed by Ramanathan and Wechsler [28], which is called phishGILLNET. Natural language processing and machine learning techniques were used in multi-layered method to build the proposed model. PhishGILLNET was tested with 400,000 emails, 10% of which were phishing, 10% were ham and the rest were spam. The experiments were conducted using 10-fold cross-validation. The proposed model achieved 97.7% accuracy for phishGILLNET2 in the classification of emails as phishing, spam, and ham. The main drawback of the study is that URL processing was not taken into consideration to determine if a hyperlink is for a phishing or legitimate website. Furthermore, the detection mechanism was based on a list of topics which can be avoided easily by new phishing attacks, which means that this solution will fail to handle zero-day attacks. In addition, the phishing email dataset used was not a proven and publicly available dataset, and instead the author selected a subset of spam emails and considered them to be phishing emails.

The main conclusion drawn from previous studies is that only a limited number can handle zero-day phishing attacks. Moreover, the approaches published so far suffer from high false positive rates and low accuracy. Furthermore, some studies did not use proven datasets or only analysed a small number of instances which may not fully represent all kinds of phishing attack. Finally, handling zero-day phishing attacks requires new behaviour that phishers may use to lure the online customer to be explored, and none of the previous studies provide clear ideas of how this can be accomplished.

3. Online phishing email detection system

3.1. Problem statement

Through investigating the previous studies, a very limited number of studies have been built to handle zero-day phishing attacks. Any model that supposed to detect zero-day phishing attack need to have the ability to dynamically adapt the detection model to reflect changes in new phishing emails. In addition, it should have the ability to explore new behaviours in newly received email in the online mode. To the best of our knowledge, none of the previous studies give a clear idea of how to explore these new behaviours in zero-day phishing emails. The proposed model called PEDS will have the ability to explore new behaviours using a novel algorithm called FEaR in any new dataset. The PEDS will incrementally enhance itself to handle new attacks depending on the idea of reinforcement learning. The proposed model will be the first work in this field that used reinforcement learning to detect zero-day phishing attack. The NN is used as the core of the detection model and a novel algorithm called DENNuRL is proposed to derive the best NN that can be used to solve the desired problem. Moreover, the detection model automatically changed to reflect changes in zero-day phishing attack.

3.2. Proposed framework

The proposed framework is called PEDS which is an online phishing email detection based on supervised and unsupervised machine learning techniques, as shown in Fig. 1. The supervised machine learning technique used a training dataset to build the detection model, while the unsupervised machine learning tries to adapt the detection model using a new delivered email to the system. The proposed model combines a neural network, reinforcement learning, data mining associative classification techniques and a set of algorithms to detect phishing attacks. The proposed framework will have the ability to explore new phishing behaviours, and consists of the following stages: pre-processing, FEaR, DENNuRL and RL-Agent. These stages will be discussed in more detail in the following sections.

3.3. Phishing email detection system

The generation of the PEDS, as shown in Fig. 1, starts by applying the pre-processing phase which will extract fifty features from the offline dataset, the pre-processing phase is described in Section 3.4. The result of the pre-processing is used as input to FEaR algorithm which will detect the important list of features based on the training dataset, the FEaR algorithm is described in Section 3.5.1. In the next step, DENNuRL, as shown in Section 3.7, will be used to generate the PEDS which will be utilized in the online mode to classify emails. Later the RL-Agent will be used continuously to adapt the PEDS to reflect newly explored behaviour in the online mode. The generated PEDS is a trained NN used to classify emails in the online mode to phishing and ham emails, the first NN is built using the DENNuRL based on the offline dataset. The PEDS is continuously adapted later using the RL-Agent to reflect changes in the online dataset. The full description of how the system evolve in the online mode will be discussed in Section 3.8.

3.4. Pre-processing

The pre-processing phase contains two steps. The first selects feature to be extracted from each email text and header; these features describe the different properties of each email. The second step includes selecting the most effective features from the set extracted in the first step to speed-up the building and adaptation of the classification model.

3.4.1. Feature selection and extraction

The features are selected from two sources: email headers and email content. These contain fifty features, eight of them are newly proposed and the rest of them drawn from previous studies. During the development of the detection system, the set of important features could change dynamically to reflect zero-day phishing attacks. The proposed features extracted from different parts of the email, and grouped into four groups depending on their nature: email headers, URLs, HTML, and text as shown in Table 1. To accomplish the pre-processing process, the email is divided into header and content. From the content, the URLs, HTML, and text are extracted depending on the email content type. The email is divided into these parts to reduce the pre-processing time, and at the end every feature has been extracted from the appropriate parts.

3.5. Description of features

The full description for the features listed in Table 1 is as follows:

- Features extracted from email header: 1-CompareMsg-SenderDomain: This compares the message ID domain and sender domain and is a binary feature that determines if the

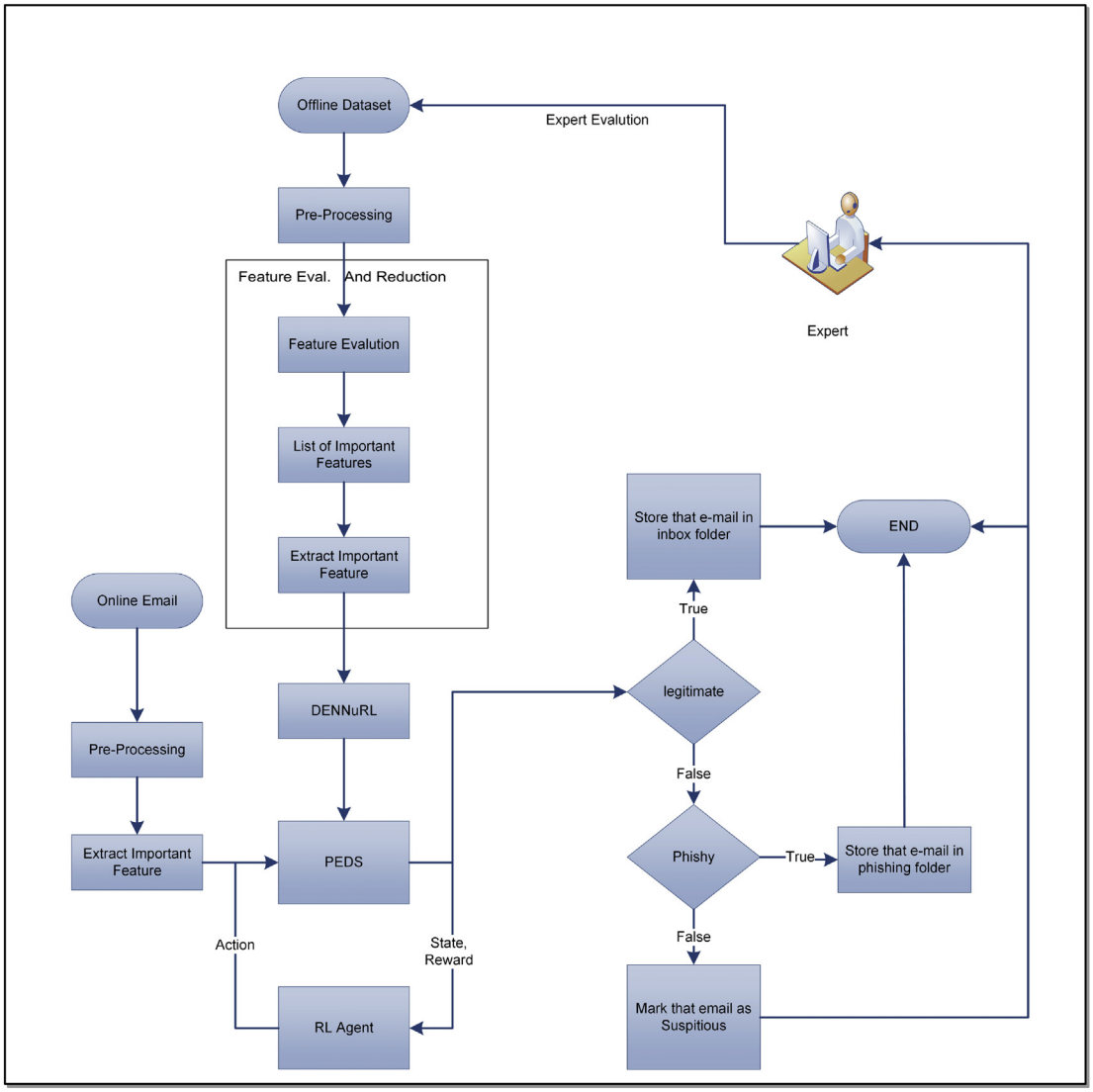

Fig. 1. Phishing email detection system.

domain names taken from the email sender are equal to those taken from the message-ID [29]. 2-HTMLmail: Binary feature that determines if the email content type is TEXT/HTML [29]. 3-Textmail: This validates if the email content type is text/plain in the email header [30]. 4-MultiPartMail: A binary feature that validates if the email content type is multi-part [30]. 5-NumberOfReceivers: Counts how many receivers are included in the (from) attribute in the email header. 6-NumberOfAttachments: Counts how many attachments there are. 7- SubjectBankWord: Checks if the email subject contains the word "bank" and returns (true) if it exists and (false) otherwise [31]. 8-SubjectDebitWord: Checks if the email subject contains the word "debit" and returns (true) if it exists and (false) otherwise [31]. 9-SubjectFwdWord: Checks if the email subject contains the word "Fwd:" and returns (true) if it exists and (false) otherwise [29]. 10-SubjectReplyWord: Checks if the email subject contains the word "Re:" and returns (true) if it exists and (false) otherwise [29]. 11-SubjectVerifyWord: Checks if the email subject contains the word "verify" and returns (true) if it exists and (false) otherwise [31]. 12-SubjectNumChars: Counts the number of characters the subject field contains and returns that number [29]. 13-SubjectNumWords: Counts the number of words the subject field contains and returns that number [29]. 14-SubjectRichness: Calculates the division between the number of words over the number of characters found in the subject field [32]. 15-SendNumWords: Counts the number of words the sender field contains and returns that number [29]. 16-SendDiffReplayto: Checks if the email sender is not the same as the "Replay to" field, and returns (true) if they are equal and (false) otherwise [29]. 17-NumberOfRecipients: Counts how many receivers are included in the (To:) attribute in the email header. 18-NumberOfCcRecipients: Counts how many receivers are included in the (Cc:) attribute in the email header. 19-NumberOfBccRecipients: Counts how many receivers are included in the (Bcc:) attribute in the email header.

- Feature extracted from URLs available in email content: 20-NumOfLink: Computes the number of hyperlinks which appear in the email body [33]. 21-NumberOfDiffDomain: Counts the number of different domains that are used in the email content as hyperlinks [33]. 22-NumDiffLinkText: Computes the number of hyperlinks that have hyperlink text which does not contain the domain name of the hyperlink target [33]. 23-NumDomainNLSender: Computes how many hyperlinks use a domain which is not equal to the sender domain [33]. 24-NumOfDotInDomain: Computes the

Table 1
List of features.

ID	Feature	ID	Feature
Features extracted from email header			
1	CompareMsgSenderDomain	36	SubjectVerifyWord
2	HTMLmail	37	SubjectNumChars
3	Textmail (New)	38	SubjectNumWords
4	MultiPartMail (New)	39	SubjectRichness
22	NumberOfReceivers (New)	40	SendNumWords
23	NumberOfAttachments (New)	41	SendDiffReplayto
32	SubjectBankWord	47	NumberOfRecipients (New)
33	SubjectDebitWord	48	NumberOfCcRecipients (New)
34	SubjectFwdWord	49	NumberOfBccRecipients (New)
35	SubjectReplyWord		
Feature extracted from URLs available in email content			
6	NumOfLink	12	NumberOfLinkContainIP
7	NumberOfDiffDomain	13	NumberOfLinkContainEsc
8	NumDiffLinkText	14	NumberOfLinkContainNSPort
9	NumDomainNLSender	42	urlBagLink
10	NumOfDotInDomain	43	UrlNumPort
11	NumberLinkContain@	50	BlackListURL (New)
Features extracted from email HTML content			
5	HTMLform	19	NumLinkNonASCII
15	ContainScript	21	NumOfDNSrDNS
16	CountSSLLink	44	ScriptOnclick
17	NumOfLinksUsingImage	45	ScriptPopup
18	NumMapLink	46	ScriptStatusChange
Feature extracted from the email main text appear to the email reader			
20	SizeOfDocument	28	BodyRichness
24	BodyDearWord	29	BodyNumFunctionWords
25	BodyNumChars	30	BodySuspensionWord
26	BodyNumWords	31	BodyVerifyYourAccountPhrase
27	BodyNumUniqueWords		

number of dots used in each hyperlink and returns the maximum number [33]. 25-NumberLinkContain@: Counts the number of links in the email body which contain the @ character [34]. 26-NumberOfLinkContainIP: Counts the number of URLs in the email which contain an IP address [33]. 27-NumberOfLinkContainEsc: Counts the number of URLs in the email body which contain hexadecimal numbers or URL-escaped characters [35]. 28-NumberOfLinkContainNSPort: Counts the number of URLs in the email body which contain a non-standard port (other than 80 or 443) [35]. 29-urlBagLink: A binary feature that returns (true) if any of the following words is found in the URL; these words are click, here, login, and update [36]. 30-UrlNumPort: Counts the number of URLs that contain a port in the authority section of that URL and returns that number [29]. 31-BlackListURL: A binary feature that returns (true) if there is any of the URL which exist in the email body exist in the black-list of URLs. These blacklist URLs are collected from PhishTank, which is a free community site where anyone can submit, verify, track and share phishing data. This feature is updated every 60 min to ensure that it contains the most recently registered phishing websites.

- Features extracted from email HTML content: 32-HTMLform: Checks if the email content contains an HTML form element [36]. 33-ContainScript: Checks if the email contains a JavaScript pop-up [36]. 34-CountSSLLink: Counts the number of URLs in the email body which point to a website that encrypts the connection with a self-signed certificate. 35-NumOfLinksUsingImage: Computes the number of pictures which are used as hyperlinks [34]. 36-NumMapLink: Computes the number of pictures with image maps that are used as hyperlinks [34]. 37-NumLinkNonASCII: Counts the number of URLs that contain non-standard ASCII characters [34]. 38-NumOfDNSrDNS: Checks if the domain names have a corresponding reverse DNS entry and return (true) if they are equal, otherwise returns (false) [37]. 39-ScriptOnclick: A binary feature that checks if an email contains onClick JavaScript event and returns (true) if it is available and return (false) if not [31]. 40-ScriptPopup: A binary feature that checks if an email contains a JavaScript pop-up windows in its content and return true if it is available and false otherwise [31]. 41-ScriptStatusChange: A binary feature that checks if an email contains a JavaScript that changes the text which appears in the status bar, and returns (true) if it is available otherwise returns (false) [34].
- Feature extracted from the email main text appear to the email reader: 42-SizeOfDocument: Returns the email message size in bytes [34]. 43-BodyDearWord: Binary feature which checks if the email body contains the word “Dear” and returns (true) if it exists, or otherwise returns (false) [31]. 44-BodyNumChars: Counts how many characters are in the email body [32]. 45-BodyNumWords: Counts how many words are in the email body [32]. 46-BodyNumUniqueWords: Counts the number of unique words in the email body [32]. 47-BodyRichness: Calculates the number of words divided by the number of characters found in the email body [32]. 48-BodyNumFunctionWords: Counts the number of function words discovered in the email body. Function words are account, access, bank, credit, click, identify, inconvenience, information, limited, log, minutes, password, recently, risk, social, security, service and suspended [32]. 49-BodySuspensionWord: Checks if the email body contain the word suspension and returns (true) if it exists and (false) otherwise [31]. 50-BodyVerifyYourAccountPhrase: Binary feature that checks if the email body contains the words (verify your account) and return (true) if it exists and (false) otherwise [31].

The newly proposed features with their IDs are: Textmail (3), MultiPartMail (4), NumberOfReceivers (22), NumberOfAttachments (23), NumberOfRecipients (47), NumberOfCcRecipients (48), NumberOfBccRecipients (49), and BlackListURL (50).

3.5.1. Feature evaluation and reduction (FEaR)

After applying the pre-processing phase and extracting the fifty features described in the previous section, the results are sent to the FEaR algorithm. This algorithm is used to determine the number of actual features that the classification process is applied to. The selected features will depend on the training dataset, which is used to determine the actual number of features that is necessary to classify the emails in the training dataset. The features used will change if the training dataset is changed. Depending on the dataset used to build the detection model, the algorithm will choose from the fifty features a set of features that is effective in determining email type. The rest of the features are not considered important and will be excluded from subsequent analysis, which will speed up the development process while preserving the same level of accuracy as if the detection model was developed with all fifty features. The steps of the FEaR algorithm are shown below, the first step will use the CART algorithm [38] to generate the regression tree (RT), then RT will be used in the later steps to reduce the number of features from fifty features to the list of important features that can classify emails in the training dataset with high accuracy.

Applying the FEaR algorithm involves the following benefits. Firstly, it will discover the phisher behaviour used in the offline dataset. Secondly, it will decrease the complexity of the generated model, by minimizing the number of input neurons and the number of connections between the input layer and the hidden layer. Thirdly, it will speed up the adaptation process to generate the best neural

network; and the speed of the adaptation will have a crucial effect on the online system. Finally, it will accelerate the classification process.

3.6. Reinforcement learning paradigm

The reinforcement learning approach has been used to learn the optimal behaviour. It is based on the idea of trial-and-error interaction with the environment. RL has the following main components as shown in Fig. 2:

(1) Agent: The controller of the system observes the system state S_t by reading observations X_t. In the proposed model the observation will be the NN architecture, the email record, and the NN output. The agent interacts with the system by performing actions Ut and it gives rewards $R_{(t+1)}$, which can be used to improve the policy. After trying these actions, a NN is generated for every action, the reward from these actions is computed and the Q-table is updated.
(2) Action: These influence the development of the environment, which apply changes in the environment. The actions Ut will be one or any number of the following. Firstly, the number of significant features can be modified, which will change the number of neurons in the input layer. Secondly, the up-to-date-dataset can be changed, which contains the offline dataset plus any new phishing and ham emails detected by the PEDS with high accuracy. These newly added emails may contain new behaviour explored by the FEaR algorithm. Lastly, the NN architecture that is used as the core of the classifications system can be changed.
(3) Policy (π): This is a mapping from the state of the environment to the action to be taken in this state. The policy is the core of the RL-Agent in the sense that it alone is sufficient to determine the behaviour of mapping from the state of the environment to the action to be taken by the RL-Agent.
(4) Reward/cost function: The reward function specifies the overall objective of the RL-Agent, and it depicts the immediate reward the agent receives for performing a certain action in a given system state.

3.7. Dynamic evolving neural network using reinforcement learning (DENNuRL)

The NN is one of the most powerful techniques that has been used extensively to build classification models. The capability of a NN model depends on characteristics such as the number of layers, the number of neurons in the hidden layers, the training function, the number of training epochs and the problem to be solved [39]. Usually, researchers determine the NN architecture manually and give a large number of training epoch, which means that the model

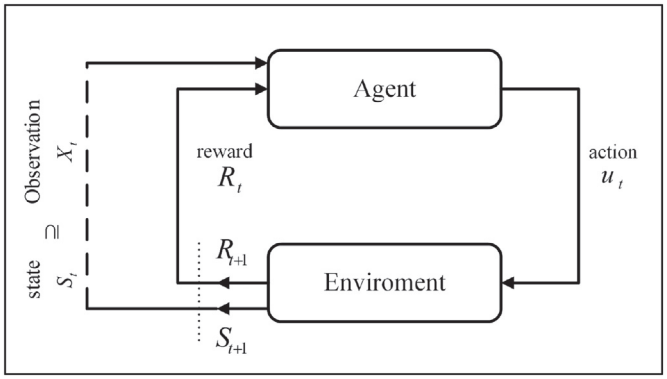

Fig. 2. Reinforcement learning framework.

generated will not be the optimal one to solve the desired problem. Furthermore, the model generated will not have the ability to adapt itself to reflect changes in the environment.

In the proposed algorithm, a NN with three layers was chosen; these are the input, hidden, and output layers. A three-layer NN was chosen because it can solve any linear or non-linear problem [40–42]. The number of neurons in the input layer depends on the number of features selected by the FEaR algorithm, which will change dynamically depending on the discovery of new behaviour. The number of neuron in the output layer for NN network built for a classification problem will depend on the problem being solved. In the proposed model the output layer will contain two neurons, as we are developing a binary classification problem. The main focus in the proposed model, as shown in Fig. 3, will be to determine the right number of neurons in the hidden layer. A large number of hidden neurons may overfit the training data, and a small number will underfit the data. Overfitting and underfitting will generate a NN with a bad level of generalization, these too issues must be taken into consideration when NN used as the core of a classification models.

During the development and enhancement of the NN, the proposed technique takes into consideration the overfitting and underfitting problems that affect NN efficiency. The overfitting problem is caused by a large number of neurons in the hidden layer. It is solved by using the merge operation. As an indication of the scale of overfitting problems, the early stopping technique [43] is used, which is accomplished by dividing the training dataset into training and validation datasets. At the time of training, the validation error is computed and if this error cannot be reduced six consecutive times, then the training process is stopped. If the validation error cannot be enhanced, this means that the architecture used has led to overfitting in the problem to be solved, even if the training error is reduced. The second issue is called underfitting, and if too few hidden neurons are chosen the NN generated will not have enough power to solve the problem, which is solved by adding neurons to the hidden layer. The steps of the DENNuRL algorithm are shown in Fig. 3, and Algorithm A2 shows the full detail. The merge operation as shown in step 5, Algorithm A2 which select which neuron to delete was proposed by Islam et al. [39] and works as follows Islam et al. [39]. To choose which neurons to merge, the following steps are applied. The correlation between hidden neurons is measured, which is calculated based on the output of the hidden neurons for the training dataset according to Eq. (A.4). The least significant hidden neuron is merged with the most correlated one, where the least significant neuron does not affect the classification process. To accomplish this action, two steps need to be applied. Firstly, the significance of every neuron is computed, which is evaluated by computing the standard deviation of the output of every neuron. A neuron's output that does not change for different inputs is considered to be less significant, as shown in Eq. (3). Secondly, the correlation is computed between the most and least significant neurons using Eqs. (1) and (2). The two neurons with the highest correlation is chosen for merging, by deleting the two correlated neurons and creating one new neuron. The input weight and bias for the new neuron is the average of the input and bias for the deleted neurons. The output weight is the sum of weight for the deleted neurons.

$$R(i,j) = \frac{C(i,j)}{\sqrt{C(i,i)C(j,j)}} \tag{1}$$

where i, j are the two neurons whose correlation $R(i,j)$ is measured, and $C = cov(x,y)$ is the covariance between all hidden neurons which is defined as described in Eq. (2):

$$con(A,B) = \frac{1}{N-1}\sum_{i=1}^{N}(A_i - \mu_A) * (B_i - \mu_B) \tag{2}$$

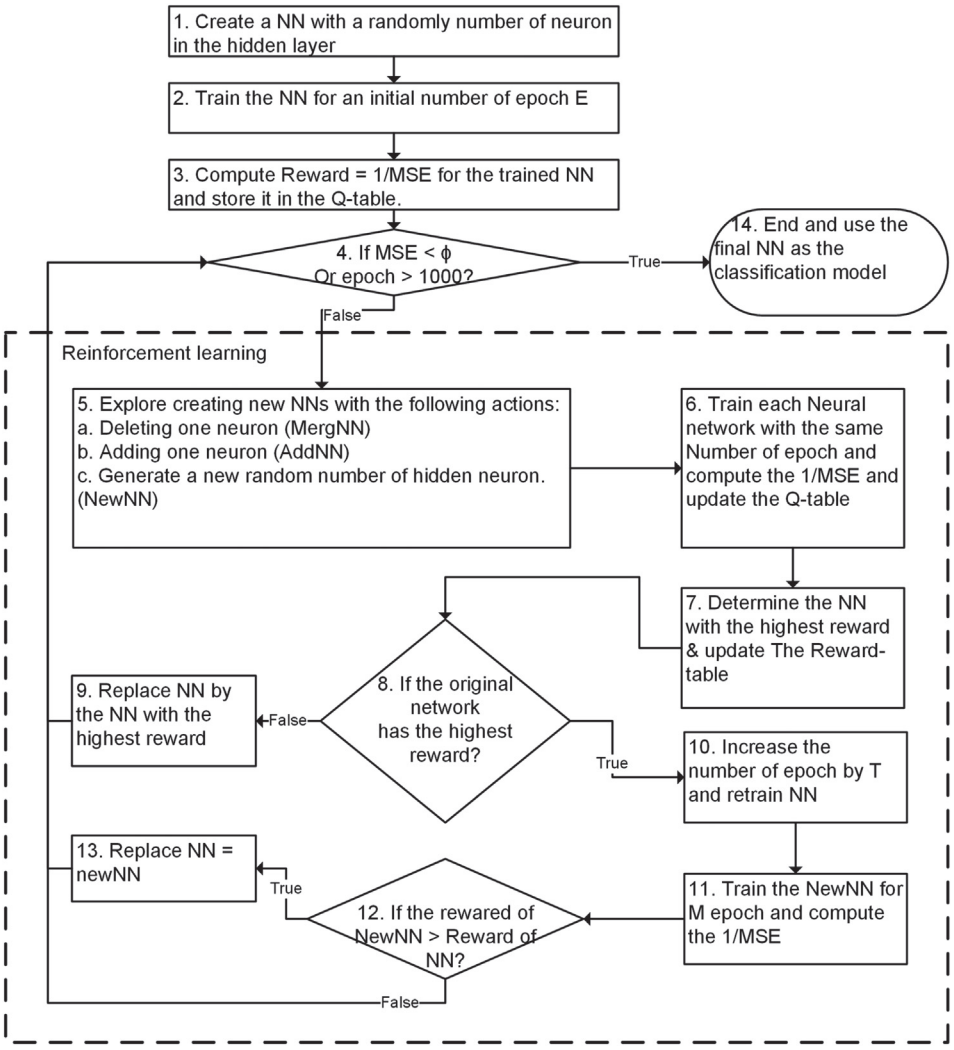

Fig. 3. Dynamic evolving neural network using reinforcement learning (DENNuRL).

where (A, B) is the output matrix of two hidden neurons for all the examples in the training dataset, μ_A is the mean of A, and μ_b is the mean of B.

$$S_d = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}|A_i - \mu|^2} \tag{3}$$

where S_d is the standard deviation and A is the output matrix for one neuron for all examples in the training dataset, and μ is the mean of A:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} A_i \tag{4}$$

3.8. Reinforcement learning agent (RL-agent)

After producing the first PEDS using the DENNuRL algorithm based on the training dataset in the offline mode, the system starts the development process by reading unclassified email in the online mode. Fig. 4 shows the RL-Agent algorithm, the system classifies emails into phishing and ham email. The RL-agent keeps watching the output of the PEDS in the online mode and works as shown in Algorithm A3.

To handle the possibility of reinforcing a wrong result, the proposed model tests the new PEDS for a reference dataset which is the offline dataset that is used to generate the first detection model. If the new PEDS cannot achieve better performance compared with the first PEDS, then we assume that the system is reinforcing a wrong classification. In such a case, the enhancement is ignored and the system is reinitialized. After all there is a possibility of miss-classification for some email that the model is not trained on (zero-day attacks); the proposed model tries to solve this problem using exploration-exploitation mechanism as shown in step 2 in the RL-Agent Algorithm. The RL-Agent Algorithm explores adding a group of emails (10% of the offline dataset) to the NewDataset. These emails are classified by the current PEDS with high accuracy ($\geq$ 95%). The NewDataset will contain the old dataset plus the new group of email.

4. Results and discussion

4.1. Dataset

The experiments were conducted using a dataset combination from three publicly available datasets, two of which are email

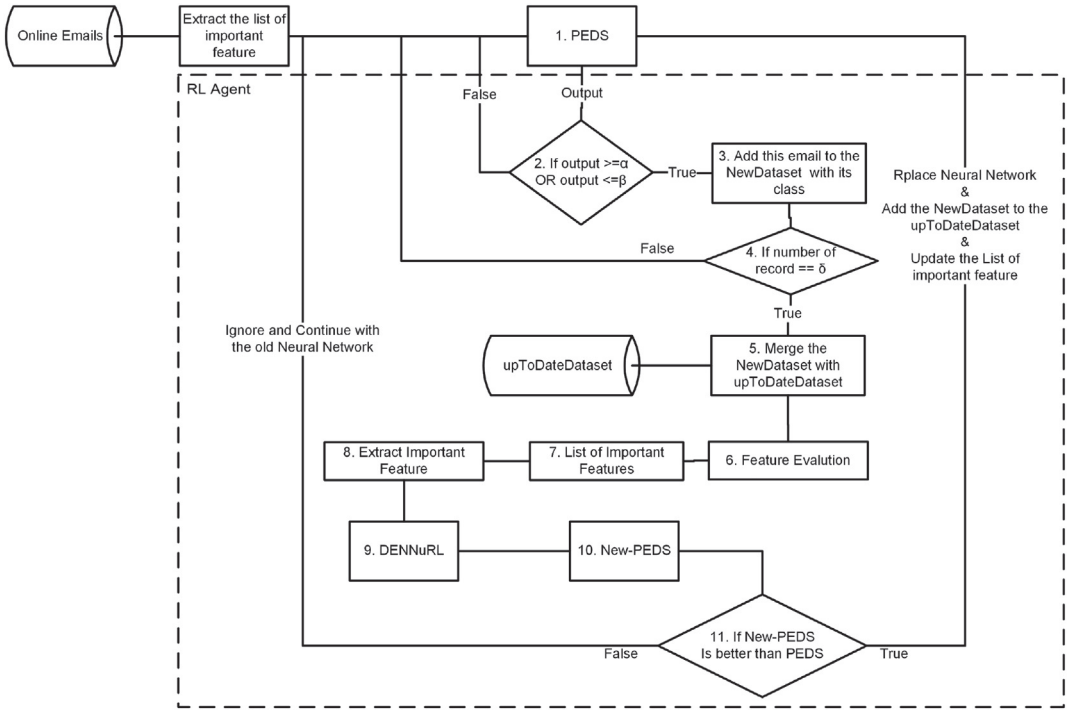

Fig. 4. Reinforcement learning agent.

datasets and one is for phishing URL datasets. These datasets are described next.

1. PhishingCorpus [26] PhishingCorpus is a phishing email dataset. This dataset was collected manually, and researchers in this field use this dataset extensively. The PhishingCorpus was collected from 2004 to 2007, and the last update of those dataset was completed in 2015. The total number of phishing emails collected was 7315.
2. SpamAssassin [27] The SpamAssassin project collected ham and spam email; the total number of emails collected is 6047 of which 4951 are ham emails. In training and testing the proposed model, the ham emails from the SpamAssassin corpus are used.
3. PhishTank [2] The PhishTank URLs dataset was collected by the PhishTank organization, which is a collaborative clearing house for data and information about phishing on the Internet. This dataset is used to update the content of the BlackListURL feature, and the list of blacklisted URLs is updated every 60 min from the PhishTank website automatically. The phishing URLs dataset is updated every hour because PhishTank updates its database hourly. 26 722 phishing URLs were collected, and this number is added to every 60 min.

To train and test the PEDS, the dataset has been divided into an offline dataset and online dataset. The total number of emails used is 4951 for ham emails and 7315 for phishing emails. From the phishing email in every experiment, 4951 emails were chosen randomly. The total number of emails was 9902, and 4000 were selected randomly as an offline dataset. The rest of the emails are used as an online dataset to assess the system's development in response to a zero-day phishing attack. The experiment was conducted and repeated fifty times, and in every iteration the offline dataset was sorted in random order and divided into 70% for training, 15% for validation and 15% for testing. The results are recorded for every iteration and then the average computed for all iterations to ensure the reliable performance of the system.

In the field of phishing email the available datasets are very limited since it contains user confidential data. The proposed model solves this problem by automatically adding more email to the offline dataset as it processes email in the online mode. In addition, the reward value in the DENNuRL algorithm is the MSE error which has the advantage of being independent of dataset size used. Moreover, the richness of the proposed model could be easily enhanced by adding any available dataset to the offline dataset.

4.2. Evaluation metrics

To evaluate the quality of the proposed model the list of the following metrics is used to evaluate the system performance. Suppose that M denotes the total number of phishing emails and D denotes the total number of legitimate emails. Then $nm \in M$ is the number of correctly detected phishing emails, while $nd \in D$ is the number of emails correctly detected as legitimate, nf is the number of legitimate emails detected as phishing, and np is the number of phishing emails detected as legitimate. For each detected email, the following evaluation method is applied: True positive (TP): The number of phishing emails correctly classified as phishing.

$$TP = nm/M \tag{5}$$

True negative (TN): The number of legitimate emails correctly classified as legitimate.

$$TN = nd/D \tag{6}$$

False positive (FP): The number of legitimate emails incorrectly classified as phishing.

$$FP = nf/D \tag{7}$$

False negative (*FN*): The number of phishing emails incorrectly classified as legitimate.

$$FN = np/M \tag{8}$$

Based on these four rules, other factors can be derived: precision, sensitivity, accuracy, and F_Measure [44–46]. *Accuracy* is the sum of *TP* and *TN* (the number of correct decisions) divided by the total number of emails. *Precision* is the ratio of how many decisions were correct, which is *TP* divided by the sum of *TP* and *FP*. *Sensitivity* is the number of *TP* assessments divided by the number of all positive assessments. The *F_Measure* is a measure of the test's accuracy, which refers to the balance between *precision* and *sensitivity*. These four metrics are shown in following definitions:

$$Precision = \frac{|TP|}{|TP| + |FP|} \tag{9}$$

$$Sensitivity = \frac{|TP|}{|TP| + |FN|} \tag{10}$$

$$Accuracy = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|} \tag{11}$$

$$F\_Measure = \frac{2 * Precision * Sensitivity}{Precision + Sensitivity} \tag{12}$$

4.3. Effect of using FEaR algorithm

To evaluate the ability of the FEaR algorithm to detect new behaviour available in the dataset, many experiments were conducted. After applying the pre-processing algorithm and extracting the fifty features for all email in the selected dataset (9900 emails), the dataset is divided into sub datasets, the elements of each dataset were chosen randomly and used as input to the FEaR algorithm to explore the important list of features.

For the training dataset the FEaR algorithm uses the CART algorithm [47] to build the RT in the first step. After building the RT the FEaR algorithm uses it as input to the later steps where the important list of features was selected. As shown in Table 2, for a dataset with 500 emails, the FEaR algorithm detected 11 features as important features that can be used to distinguish between phishing and ham emails. With changing in the training dataset, a new phishing email is explored which could mean a new behaviour used by the phisher to lure online customers. The FEaR algorithm builds a different regression tree that can best classify the emails in the training dataset, and in every experiment it changes the tree nodes (features) from the total list of features. The FEaR algorithm showed the ability to explore these behaviours, when the algorithm processes datasets of different list of emails, a different list of features is selected. In the next section, the selected list of features is used to build the classification model, and the system is evaluated to see if it can detect phishing emails with high accuracy. In Table 2, for the sub dataset of size 1000, the FEaR algorithm explores different set of features, this happen due to the fact that different sets of email where chosen randomly where different phishing email may include different behaviours user by phishers to lure the online customers. The ability of the proposed algorithm to dynamically choose different set of behaviours when the training dataset is changed will be a main part of the online phishing email detection system. The proposed algorithm for the selection from a large set of features solves the problem of selecting the right set of features to be used to build a classification model. Furthermore, the number of important features changes dynamically when the dataset changes without any user intervention, whereas other studies [25,34,48] have used fixed numbers of features to classify phishing email and the sets of features chosen are changed manually. Moreover, the proposed algorithm can be used to rank the selected features, as shown in Table 3 where the ranks ($V\,i$) are evaluated in step 3 in the FEaR algorithm. In the online mode, when building a new dataset that contains new phishing emails, the FEaR algorithm will explore the new behaviours that the new dataset may contain. In this section the FEaR algorithm is evaluated with different sizes of dataset, using the complete list of features as input and selecting the most effective list of features that can classify email in the dataset under investigation. If the dataset size is changed, it will contain different groups of phishing and ham email. The FEaR algorithm successfully selects different lists of features for different datasets. The experiments conducted in the next section will show that the selected list of features by FEaR algorithm can be used to build a classification model that can classify emails with a high level of accuracy.

Some of the newly proposed features are selected by the FEaR algorithm. As shown in Table 2, features (3, 4, 22, 23, and 48) are the

Table 2
Important features discovered using FEaR algorithm.

Dataset size	Count of important feature	Features
500	11	3, 6, 11, 20, 24, 28, 29, 35, 37, 39, 41
1000	17	1, 3, 6, 7, 11, 17, 20, 25, 26, 28, 29, 34, 35, 37, 39, 40, 44
1000	18	3, 6, 7, 11, 20, 21, 24, 25, 26, 27, 28, 29, 35, 37, 38, 39, 40, 41
1000	18	3, 6, 7, 8, 10, 11, 20, 24, 25, 26, 28, 29, 34, 35, 37, 39, 40, 41
1500	19	3, 6, 9, 10, 11, 17, 20, 22, 23, 24, 26, 27, 28, 29, 35, 37, 38, 39, 40
2000	23	1, 3, 6, 7, 10, 11, 16, 17, 20, 22, 24, 26, 27, 28, 29, 34, 35, 36, 37, 39, 40, 41, 42
2500	27	1, 3, 6, 7, 8, 11, 16, 17, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 34, 35, 36, 37, 38, 39, 40, 41, 48
3000	28	3, 4, 6, 7, 9, 10, 11, 12, 20, 22, 23, 24, 25, 26, 27, 28, 29, 32, 35, 36, 37, 38, 39, 40, 41, 42, 43, 48
3500	29	1, 3, 6, 7, 8, 10, 11, 16, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44
4000	30	1, 3, 6, 7, 8, 9, 10, 11, 16, 17, 20, 22, 23, 24, 25, 26, 27, 28, 29, 32, 34, 35, 36, 37, 38, 39, 40, 41, 42, 44
9900	33	1, 3, 6, 7, 8, 9, 10, 11, 16, 17, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 31, 32, 34, 35, 36, 37, 38, 39, 40, 41, 42, 44, 48

Table 3
Feature ranking evaluated by FEaR algorithm with 4000 emails.

Feature ID	Rank	Feature ID	Rank
3	100	25	0.663486
7	13.96487	17	0.546267
24	6.499948	11	0.498457
35	5.17843	22	0.387983
26	3.771794	23	0.346812
29	2.266412	1	0.341654
28	2.033311	16	0.295951
6	1.810769	38	0.28526
39	1.278216	32	0.282259
41	0.926993	34	0.214601
20	0.914429	36	0.13632
27	0.878601	10	0.06674
44	0.829632	42	0.045931
37	0.702157	8	0.036652
40	0.684341	9	0.036301

Table 4
An example of the NN adaptation using the DENNuRL algorithm.

NN No.	Number of neuron in the hidden layer	Number of epoch	MSE error
1	27	10	0.024355
2	37	10	0.023192
3	38	10	0.023105
4	39	10	0.023026
5	6	10	0.022724
6	7	10	0.022211
7	7	60	0.012845
8	8	60	0.01118
9	8	360	0.010688
10	20	610	0.010609
11	19	610	0.010607
12	19	1010	0.009976

new features proposed in this study that are selected as important features by FEaR algorithm. The selection of these new features by FEaR algorithm depends on the size of the dataset, as an example, if the dataset size is 500 emails, then the FEaR algorithm selects one new feature (feature 3). For 3000 emails Features (3, 4, 22, and 23) are selected as important. The rank of the newly proposed features shows that such features are part of the most important features when compared with the previously proposed features as shown in Table 3. As an example, feature 3 is selected as the most important feature.

In this section the FEaR algorithm is evaluated with different sizes of dataset, using the complete list of features as input and selecting the most effective list of features that can classify email in the dataset under investigation. If the dataset size is changed, it will contain different groups of phishing and ham email. The FEaR algorithm successfully selects different lists of features for different datasets. The experiments conducted in the next section will show that the selected list of features by FEaR algorithm can be used to build a classification model that can classify emails with a high level of accuracy.

The pre-processing algorithm extracted a large set of features which cover all aspects of emails where features are selected and extracted from the email header and content. The selected list of features is not intended to be optimal, it represents an information in the email which could be used in future attacks. Furthermore, the richness of the proposed model can be enhanced by adding any possible feature that represents any information in the email which may be used by phishers in future attacks. The FEaR algorithm solves the problem of selecting the active list of features that can classify the training dataset with the highest accuracy.

4.4. DENNuRL

As discussed before, the DENNuRL algorithm, as shown in Fig. 3, is used to generate the first PEDS employed in online mode. Later in online mode, the same algorithm will be utilized as part of the RL-Agent as shown in Fig. 4. In both cases, it will be used to automatically choose the best NN to solve the desired problem.

The first PEDS is built by using the DENNuRL algorithm based on the offline dataset (4000 emails). As an example of the effect of DENNuRL algorithm, Table 4 shows the enhancement of the NN, from the NN randomly generated in the first step to the final version that will be used as a PEDS. The MSE error guides the enhancement process where the NN with the lower MSE error will be better. Fig. 5 shows the enhancement performed by DENNuRL algorithm in terms of MSE error, with the system developments. MSE error is computed based on Eq. (A.4) as described before. In the first NN shown in Table 4, the number of neuron 27 is selected randomly by the algorithm, then the system tries the different actions and selects the NN that returns the maximum rewards (minimum MSE error). In the second enhancement, a new random NN is selected, at the third enhancement, one more neuron is added to the hidden layer. At step 11, two neurons were merged in the hidden layer, and finally, in the last enhancement the training epoch is increased in order to reduce the MSE error.

Table 4 and Fig. 5 show the enhancement performed by the DENNuRL algorithm in terms of MSE error, according to the system's development. In the first NN shown in Table 4, the number of 27 neurons is selected randomly by the algorithm, and then the system tries the different actions and selects the NN that returns the maximum rewards (minimum MSE error). In the second enhancement, a new random NN is selected, and at the third enhancement one more neuron is added to the hidden layer. At step 11, two neurons are merged in the hidden layer, and finally in the last enhancement the training epoch is increased in order to reduce the MSE error.

4.5. The phishing email detection system (PEDS)

After building the first PEDS using the DENNuRL algorithm, the system starts classifying the online dataset described in Section 4.1. While PEDS classifies email in the online mode the RL-agent keeps track of the output of the system. To compare the ability of the PEDS to detect phishing emails for the system before and after the adaptation, a detection error trade-off (DET) curve [49,50] is used. The DET curve is a graphical representation that shows a trade-off between the two types of error: missed detections (y-axis) and false alarms (x-axis). Missed detection (false negative) is where a phishing email is classified as legitimate, and a false alarm (false positive) is where a legitimate email is classified as phishing. The two kinds of error were evaluated for the two FEDS for the same dataset selected randomly from the original dataset. The first system is the PEDS developed initially before applying the adaptation process, and the second is the PEDS after applying the adaptation using the RL-Agent. The DET gives a good indication of the efficiency of the system at different operation points. The DET curve shown in Fig. 6 clearly indicates that the adapted system has lower probabilities of false rejection and false acceptance rates at all operation points. The DET curve is generated by sweeping the decisions threshold over the range of scores produced by the detection system, and performance is evaluated at small, successive intervals. Then the normal deviations of the miss versus false alarm rates at each interval are plotted. The detection error trade-off function, which takes in a list of prediction values and a list of truth values, produces output containing the

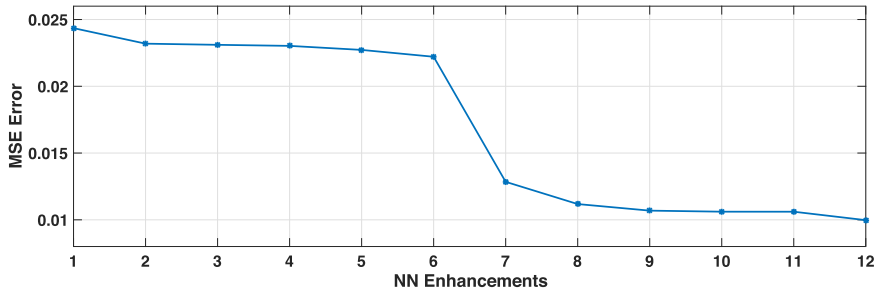

Fig. 5. An example of NN enhancement using the DENNuRL in terms of MSE error.

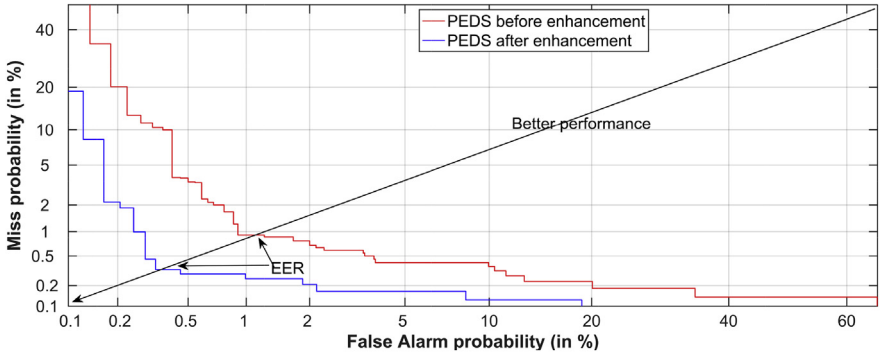

Fig. 6. Detection error tradeoff (DET) for PEDS before and after adaptation.

false alarm (FPR) and missed detection (FNR) for all threshold values. The DET curve, which is the line showing the trade-off between FPR and FNR, is typically viewed in logarithmic coordinates. The DET curve shown in Fig. 6 shows that the two kinds of error, false alarms (x-axis) and miss detection (y-axis) are related. The nature of this relationship is determined using the following equation [51]:

$$y = -\left(\frac{1}{\sigma_T}x + \frac{\mu_T}{\sigma_T}\right) \tag{13}$$

where (μ_T, σ_T) are the parameters of the target distribution, which is assumed to be Gaussian. A special point in the DET curve, where the probability of false alarms is equal to the probability of missed detection of (the targeted email) is called the equal error rate (EER) as shown in Fig. 6. The EER can be evaluated using Eq. (14).

$$x = y = -\frac{\mu_T}{1+\sigma_T} \quad \text{and} \quad P_{EER} = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-t^2/2}dt \tag{14}$$

By visual inspection of the DET curve, we can confirm that the PEDS after adaptation is better taking into consideration the two kinds of error. A weighted average of missed detection and false alarm rates may be used as a kind of figure of merit or cost function. The point on the DET curve where such an average is minimized is called the EER. In Fig. 6 these points are indicated by arrows.

Table 5 summarizes the enhancement of the PEDS. To show how the system evolves in the online mode, eight different metrics were computed, which were accuracy, precision, recall, F_Measure, TPR, FPR, TNR, and FNR after every adaptation of the FEDS in the online mode. Fig. 7 shows how the system evolves in terms of accuracy, where fifteen enhancements have been conducted throughout the processing of the online dataset of 5902 emails, and the results clearly show an enhancement in all metrics over time.

Finally, to show the reliable performance of the proposed model, the experiment was repeated fifty times. Every experiment started from a random case and the final PEDS was produced after the enhancements were tested. Table 6 shows a summary of the different metrics used in the assessment of PEDS performance; the experiment was repeated fifty times and an average is taken.

4.6. Comparative analysis

Table 7 shows the outcome of the comparison of the present results with those of previous work, the phishing and ham emails used in these works were mainly collected from Monkey.org [26] and the SpamAssasin [27] corpora.

Islam and Abawajy [25] proposed a multi-tier classification model that used a combination of three classification algorithms (SVM, AdaBoost, and NaiveBayes). In this study twenty-one features were extracted from email header and content. The highest accuracy registered is 97% with the arrangement ($C1C3C2$) where $C1$ as support vector machine, $C2$ as Boosting (AdaBoost) and $C3$ as Bayesian (Naive Bayes). However, the proposed algorithm could not be modified to handle new type of attacks that the classifier combination does not have the ability to be adapted to reflect changes in the online emails. Almomani et al. [4] proposed a novel model that handles zero-day phishing attacks. Twenty-one features were used in building the detection model and 98% accuracy was achieved, but the proposed model in the online mode was tested with only a tiny dataset of 300 phishing emails and 2000 legitimate emails. Khonji et al. [31] used 47 features to represent phishing emails and achieved 97% accuracy [31]. However, this study did not take into consideration metrics such as true positive and true negative rates or AUC and false negative rates, despite the fact that these metrics are extremely important. Gansterer and Pölz [34] used 30 features, and applied the J48 and SVM algorithms for classification and achieved 97% accuracy, but this study only used the metrics of accuracy and false positive rate. Ramanathan and Wechsler [28] used the PLSA topic trainer to build the trained model, and a high accuracy was achieved at 97.7% while

Table 5
PEDS enhancements in terms of FNR, FPR, TPR, Accuracy, Precision, Recall, and F_Measure.

Enh.	FNR	FPR	TPR	TNR	ACC	Precision	Recall	F_Measure
1	1.89%	5.15%	98.11%	94.85%	96.49%	95.06%	98.11%	96.56%
2	1.89%	4.82%	98.11%	95.18%	96.65%	95.36%	98.11%	96.72%
3	1.52%	5.15%	98.48%	94.85%	96.67%	95.08%	98.48%	96.75%
4	1.52%	5.15%	98.48%	94.85%	96.67%	95.08%	98.48%	96.75%
5	1.52%	5.11%	98.48%	94.89%	96.69%	95.11%	98.48%	96.77%
6	1.56%	5.11%	98.44%	94.89%	96.67%	95.11%	98.44%	96.75%
7	1.60%	4.36%	98.40%	95.64%	97.02%	95.79%	98.40%	97.08%
8	1.60%	4.48%	98.40%	95.52%	96.96%	95.68%	98.40%	97.02%
9	1.48%	3.57%	98.52%	96.43%	97.48%	96.53%	98.52%	97.52%
10	1.07%	2.82%	98.93%	97.18%	98.06%	97.25%	98.93%	98.08%
11	1.07%	2.82%	98.93%	97.18%	98.06%	97.25%	98.93%	98.08%
12	1.11%	2.82%	98.89%	97.18%	98.04%	97.25%	98.89%	98.06%
13	0.99%	2.62%	99.01%	97.38%	98.20%	97.45%	99.01%	98.22%
14	1.03%	2.03%	98.97%	97.97%	98.47%	98.00%	98.97%	98.49%
15	1.15%	1.74%	98.85%	98.26%	98.55%	98.28%	98.85%	98.56%

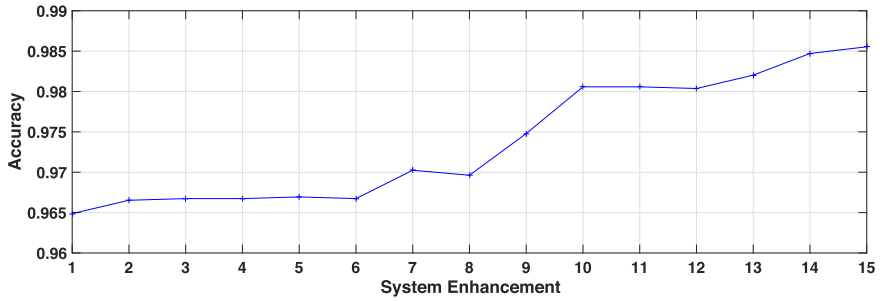

Fig. 7. System enhancements in terms of accuracy.

using the largest dataset in comparisons with other studies. The main disadvantages of this study including ignoring many features that are related to URLs, which are the main connections between the emails and phishing websites. Furthermore, in the selection of the dataset it was assumed that the Enron email dataset is free from spam and phishing email, which has not been proven by the dataset producer. Finally, they did not consider redundancy in this vast dataset. Ma et al. [52] proposed a hybrid approach with seven features, and it successfully achieved 99% accuracy. However, this study did not use a verified dataset of phishing and legitimate emails, and also only 7% of the dataset were phishing emails. Toolan and Carthy [29] used 22 features to test three datasets. Approximately 97% accuracy was achieved for the first test which did not include phishing emails, but for tests two and three which included phishing emails much lower accuracy rates of 84% and 79% were achieved. Hamid and Abawajy [48] also used seven hybrid features with several datasets and achieved 96% accuracy, but this accuracy rate was reduced when dataset size increased. The present approach used fifty hybrid features which was reduced to a number of effective features by the feature evaluation and reduction phase. The number of important features changed dynamically to reflect changes in the dataset. The dataset grows dynamically by running the proposed model in the online mode. Only a few studies have considered handling zero-day phishing attacks, yet our model has been proven to have the capability to handle zero-day phishing attacks with high accuracy. In the field of phishing email detection any small enhancement in the overall metrics is significant [25,34]. Because when the system is wrongly classified one email as phishing and the user cannot see that email it could cause a loose of a significant information. Moreover, the proposed model successfully achieved an enhancement in accuracy from 96.5% to 98.6% as shown previously in Fig. 7, and the enhancement continues as more email received by the system. The proposed model takes 4000 emails as a training dataset which is 44% of the overall dataset, while other researcher used 90% of the dataset as a training dataset. Nevertheless, the proposed system shows an ability to enhance itself in the online mode and overwhelms the existing studies. The proposed model can overcome the problem of high false positive rate by increasing the size of training dataset. Moreover, the system shows an ability to enhance itself in the online mode in terms of all metrics as shown previously in Table 5 and Fig. 7, while it processes in the online mode. On the other hand, the error rate for previous studies will be increased dramatically when handling zero-day phishing attacks, because it is not modifying the list of features to reflect changes in the zero-day attacks.

5. Conclusions and future work

Phishing is one of the most serious cybercrime threats that reduces the costumer trust in the trade-commerce. This paper shows how a NN with RL can be used to build a powerful model to detect zero-day phishing attacks with high performance levels and acceptable error rates. A novel algorithm has been proposed for developing the best NN, as well as an algorithm to explore new behaviour and adapt the phishing email detection system dynamically, with the capability to explore new behaviours. The significance of the proposed model lies in the following aspects. Firstly, the PEDS is capable of online detection with the ability to adapt itself to reflect changes in the dataset. Secondly, the reinforcement learning methodology used in the proposed approach increases the ability of the system to evolve based on changes in the environment. Finally, the performance of the online PEDS was evaluated using a set of metrics and compared with previously proposed approaches for phishing email detection. The proposed approach registers high accuracy, TPR, and TNR at 98.63%, 99.07%, and 98.19% respectively. In addition, it shows low FPR and FNR, at 1.81% and 0.93% respectively, and the rest of the metrics used to evaluate the proposed approach are summarized in Table 6. Future work may include adding more datasets to the offline dataset to increase the richness of the model, and the model may be extended to classify spam email as well as phishing and ham email.

Table 6
Results for the online PEDS averaged over 50 independent runs.

FNR	FPR	TPR	TNR	ACC	Precision	Recall	F_Measure	AUC
0.93%	1.81%	99.07%	98.19%	98.63%	98.21%	99.07%	98.64%	99.43%

Table 7
Comparison of our approach with previous work.

Author	No. of features	Feature approach	Sample	Results
Islam and Abawajy [25]	21	Hybrid	Undetermined size	ACC 97% FP 2% FN 9%
Almomani et al. [4]	21	Hybrid	Phishing 4300 Legitimate 6000	ACC 98%
Khonji et al. [31]	47	Hybrid	Phishing 4116 Legitimate 4150	ACC 97% FP 0.60%
Gansterer and Pölz [34]	30	Hybrid	Phishing 5000 Legitimate 5000	ACC 97%
Ramanathan et al. [28]	200 topics	Content	400,000 email	ACC 97.7%
Ma et al. [52]	7	Hybrid	Phishing 46,525 Legitimate 613,048	ACC 99%
Toolan and Carthy [29]	22	Hybrid	Ham 4202 Spam 1895 Phishing 4563	ACC 97%
Hamid & Abawajy [48]	7	Hybrid	Total 4594	ACC 92%
Proposed approach	50	Hybrid	Phishing 4559 Legitimate 4559	ACC 98.6% FP 1.8%

Appendix A. Algorithms

Algorithm A1. Feature evaluation and reduction(FEaR).

1: Building a classification and regression tree using the CART algorithm. A brief description of how the CART algorithm works is as follows:
   a. Start with the training dataset.
   b. Take into consideration all possible values of all features; in this case, fifty features.
   c. Choose one feature x_i with a specific value $t_1, (x_i = t_1)$ that generates the largest division in the email class, which will be the feature that can have the best division in relation to the email class.
   d. Split the data into two branches: if $(x_i < t_1)$, then send the data to the left-hand branch; otherwise, send the data to the right-hand branch.
   e. Repeat the same process on these two nodes. The final output is a tree as shown in which is used as input to the second step. The nodes in this tree will be the selected features that can generate the best division of the training dataset.
2: Evaluate the importance of every feature, where every node in the tree generated in the first step will represent one feature. The importance of every feature (node 1) that has two children (nodes 2 and 3) is estimated by applying Eq. (A.1):

$$V_1 = \frac{(R_1 - R_2 - R_3)}{Num\_node} \tag{A.1}$$

where R_1, R_2 and R_3 are the node risks for the parent and children nodes, and Num_node is the total number of nodes in this tree. The risk is defined as shown in Eq. (A.2).

$$R_i = P_i * E_i \tag{A.2}$$

where P_i is the node probability and E_i is the node error computed by the CART algorithm for every node in step 1.
3: Create a crisp value for each feature by dividing the importance value of each feature computed in step 2 by the maximum importance value as shown in Eq. (A.3):

$$\left\{ V_i = \frac{V_i}{max(v))} * 100 | \forall i = 1, 2, 3 \ldots n \right\} \tag{A.3}$$

where V_i is the importance of feature i computed in step 2, n is the number of features, v is the vector of all features, and $max(v)$ is the maximum value in the vector v.
4: Select the features with crisp values $V_i > 0$ as important features. The features with $V_i = 0$ do not have a corresponding node in the tree constructed in step 1.
5: Reduce the number of features extracted in the preprocessing phase from 50 features to the list of important features determined in step 4.

Algorithm A2. Dynamic evolving neural network using reinforcement learning (DENNuRL).

1: A NN is created that contains a random number of neurons in the hidden layer, and the numbers of neurons in the input and output layers will depend on the problem being solved as discussed previously.
2: The NN constructed in the first step is trained for an initial number of epochs E, and initially only a small number of epochs is chosen ($E = 10$).
3: The NN trained in step 2 is tested, and the MSE computed as shown in Eq. (A.4). This determines the reward value since it has the advantage of being independent of dataset size:

$$MSE = \frac{\sum_{i=1}^{n} (o_i - t_i)^2}{n} \tag{A.4}$$

where o_i is the output for an email, t_i is the desired target for the same email, and n is the number of emails used in the evaluation process.

$$Reward = 1/MSE \tag{A.5}$$

4: Check if the termination condition is satisfied ($MSE < \Phi$) or ($epoch > 1000$) and if so go to step 14. The value of Φ is manually selected, and depends on the acceptable error rate for that problem.
5: In the proposed model, three possible actions are explored. These actions are as follows:
   a. One neuron is deleted from the hidden layer by merging two neurons.
   b. One neuron is added and the NN is retrained.
   c. A new random number of hidden neurons s generated and the NN is retrained. This action has a crucial effect in avoiding being trap into local minima, where a new architecture is investigated which is away from the local NN architecture.
6: Each of the NNs explored in step 5 is trained and the reward table is updated, where the reward from each NN is computed based on Eq. (A.5).
7: Determine the action to be exploited using Eq. (A.6), and the NN that returns the maximum rewards will be chosen.

$$Q(s,a) \leftarrow Q(s_t + a_t)\alpha[r_{t+1} + \gamma max Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{A.6}$$

where r_{t+1} is the reward computed for every NN explored using Eq. (A.5), α is the learning rate, γ is the discount factor, and $Q(s,a)$ is a table of s rows representing the system states and a columns representing the actions to be taken. The values of the learning rate is $0 < \alpha \leq 1$ and the discount factor $0 < \gamma \leq 1$ are manually selected by the user. The action that returns the maximum reward is chosen.
8: If the new NN has a higher reward, go to step 9. Otherwise go to step 10.
9: The old NN is replaced by the new one that has the higher reward.
10: The number of epochs is increased by a small amount, $T = 10$, and create a new NN.
11: The new NN is trained and the reward computed.
12: If the reward from the new NN is greater than the reward from the old NN, then go to step 13. Otherwise, go to step 4.
13: Replace the NN with the NN that has the highest reward.
14: The adaptation process is stopped and the final NN is used as the classification model.

Algorithm A3. Reinforcement learning agent (RL-agent).

1: Emails are classified one at a time and passed to the evaluation system as shown in Fig. 1, and are then passed to step 2.
2: If an email is classified with a very high accuracy $ACC \geq \alpha$ and $ACC \leq \beta$, go to step 3. Otherwise read next email, returning to step 1.

3: Email classified using the PEDS are collected in a new dataset (NewDataset), and all features originally extracted in the pre-processing phase are considered in this step.
4: If the number of emails collected in the *NewDataset* $= \delta$, go to step 5. Otherwise the next email is read, returing to step 1. The value of δ is dynamically changes which is 10% of the upToDateDataset, which is initially equal to the offline dataset.
5: The system merges the NewDataset with the upToDate-Dataset, and the result is stored in the NewDataset.
6: The FEAR algorithm is applied to the NewDataset to explore new phishing behaviours in the NewDataset.
7: The list of important features is determined.
8: The list of important features from the NewDataset is extracted.
9: A new PEDS is generated using the DENNuRL algorithm to reflect changes in the dataset.
10: The newly generated PEDS is tested.
11: If the new PEDS is better, it replaces the old one, the upTo-DateDataset is replaced with the NewDataset and the list of important features is updated. Otherwise, the adaptation process is ignored and the old PEDS is used.

References

[1] Federal Trade Commission, et al. Consumer Sentinel Network Data Book for January–December 2015, 2016.
[2] L. OpenDNS, PhishTank, 2016, https://www.phishtank.com/index.php.
[3] APWG, Phishing activity trends report, 1st–3rd Quarters 2015, Report, APWG. 2015. https://docs.apwg.org/reports/apwg_trends_report_q1-q3_2015.pdf.
[4] A. Almomani, T.-C. Wan, A. Manasrah, A. Altaher, M. Baklizi, S. Ramadass, An enhanced online phishing e-mail detection framework based on evolving connectionist system, International Journal of Innovative Computing, Information and Control (IJICIC) 9 (2013) 169–175.
[5] G. Ramesh, I. Krishnamurthi, K.S.S. Kumar, An efficacious method for detecting phishing webpages through target domain identification, Decision Support Systems 61 (2014) 12–22.
[6] R.M. Mohammad, F. Thabtah, L. McCluskey, Predicting phishing websites based on self-structuring neural network, Neural Computing and Applications 25 (2014) 443–458.
[7] A. Kang, J.D. Lee, W.M. Kang, L. Barolli, J.H. Park, Security considerations for smart phone smishing attacks, Advances in Computer Science and its Applications, Springer. 2014, pp. 467–473.
[8] D. Schniederjans, E.S. Cao, M. Schniederjans, Enhancing financial performance with social media: an impression management perspective, Decision Support Systems 55 (2013) 911–918.
[9] M. Collier, D. Endler, Hacking Exposed Unified Communications & VoIP Security Secrets & Solutions, McGraw-Hill Osborne Media. 2013.
[10] X.R. Luo, W. Zhang, S. Burd, A. Seazzu, Investigating phishing victimization with the heuristic-systematic model: a theoretical framework and an exploration, Computers & Security 38 (2013) 28–38.
[11] M.-E. Maurer, Counteracting Phishing Through HCI: Detecting Attacks and Warning Users, München, Ludwig-Maximilians-Universität, Diss., 2014. 2014. (Ph.D. thesis).
[12] A. Vishwanath, T. Herath, R. Chen, J. Wang, H.R. Rao, Why do people get phished? Testing individual differences in phishing vulnerability within an integrated, information processing model, Decision Support Systems 51 (2011) 576–586.
[13] C.L. Tan, K.L. Chiew, K. Wong, PhishWHO: Phishing webpage detection via identity keywords extraction and target domain name finder, Decision Support Systems 88 (2016) 18–27.
[14] K. Krombholz, H. Hobel, M. Huber, E. Weippl, Advanced social engineering attacks, Journal of Information Security and Applications 22 (2015) 113–122.
[15] D.D. Caputo, S.L. Pfleeger, J.D. Freeman, M.E. Johnson, Going spear phishing: exploring embedded training and awareness, IEEE Security & Privacy 12 (2014) 28–38.
[16] A. Verma, Effects of phishing on E-commerce with special reference to India, Interdisciplinary Perspectives on Business Convergence, Computing, and Legality (2013) 186.
[17] T. Kobayashi, H. Okada, The effects of similarities to previous buyers on trust and intention to buy from E-commerce stores: an experimental study based on the SVS model, IT Enabled Services, Springer. 2013, pp. 19–38.
[18] M. Alsharnouby, F. Alaca, S. Chiasson, Why phishing still works: user strategies for combating phishing attacks, International Journal of Human-Computer Studies 82 (2015) 69–82.
[19] A. Vishwanath, B. Harrison, Y.J. Ng, Suspicion, cognition, and automaticity model of phishing susceptibility, Communication Research (2016)p. 0093650215627483.
[20] J. Wang, R. Chen, T. Herath, H.R. Rao, Visual e-mail authentication and identification services: an investigation of the effects on e-mail use, Decision Support Systems 48 (2009) 92–102.
[21] L. Li, M. Helenius, E. Berki, A usability test of whitelist and blacklist-based anti-phishing application, Proceeding of the 16th International Academic MindTrek Conference, ACM. 2012, pp. 195–202.
[22] C.K. Olivo, A.O. Santin, L.S. Oliveira, Obtaining the threat model for e-mail phishing, Applied Soft Computing 13 (2013) 4841–4848.
[23] S. Sheng, B. Wardman, G. Warner, L.F. Cranor, J. Hong, C. Zhang, An empirical analysis of phishing blacklists, Proceedings of Sixth Conference on Email and Anti-Spam (CEAS), 2009.
[24] R.M. Mohammad, F. Thabtah, L. McCluskey, Tutorial and critical analysis of phishing websites methods, Computer Science Review 17 (2015) 1–24.
[25] R. Islam, J. Abawajy, A multi-tier phishing detection and filtering approach, Journal of Network and Computer Applications 36 (2013) 324–335.
[26] Nazario, Phishing Corpus, 2015, http://monkey.org/jose/phishing/.
[27] J. Mason, The Apache SpamAssassin Public Corpus, 2005, http://spamassassin.apache.org/publiccorpus.
[28] V. Ramanathan, H. Wechsler, phishGILLNET phishing detection methodology using probabilistic latent semantic analysis, AdaBoost, and co-training, EURASIP Journal on Information Security 2012 (2012) 1.
[29] F. Toolan, J. Carthy, Feature selection for spam and phishing detection, eCrime Researchers Summit (eCrime), 2010, IEEE. 2010, pp. 1–12.
[30] S. Smadi, N. Aslam, L. Zhang, R. Alasem, M. Hossain, Detection of phishing emails using data mining algorithms, 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), IEEE. 2015, pp. 1–8.
[31] M. Khonji, Y. Iraqi, A. Jones, Enhancing phishing E-Mail classifiers: a lexical URL analysis approach, International Journal for Information Security Research (IJISR) 2 (2012).
[32] M. Chandrasekaran, K. Narayanan, S. Upadhyaya, Phishing email detection based on structural properties, NYS Cyber Security Conference, 2006. pp. 1–7.
[33] I. Fette, N. Sadeh, A. Tomasic, Learning to detect phishing emails, Proceedings of the 16th International Conference on World Wide Web, ACM. 2007, pp. 649–656.
[34] W.N. Gansterer, D. Pölz, E-mail classification for phishing defense, European Conference on Information Retrieval, Springer. 2009, pp. 449–460.
[35] C.E. Drake, J.J. Oliver, E.J. Koontz, Anatomy of a Phishing Email, CEAS, Citeseer. 2004.
[36] A. Bergholz, J. De Beer, S. Glahn, M.-F. Moens, G. Paaß, S. Strobel, New filtering approaches for phishing email, Journal of Computer Security 18 (2010) 7–35.
[37] A. Inomata, M. Rahman, T. Okamoto, E. Okamoto, A novel mail filtering method against phishing, PACRIM. 2005 IEEE Pacific Rim Conference on Communications, Computers and signal Processing, 2005, IEEE. 2005, pp. 221–224.
[38] R.A. Berk, Classification and regression trees (CART), Statistical Learning from a Regression Perspective, Springer. 2016, pp. 129–186.
[39] M.M. Islam, M.A. Sattar, M.F. Amin, X. Yao, K. Murase, A new adaptive merging and growing algorithm for designing artificial neural networks, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39 (2009) 705–722.
[40] C.C. Aggarwal, Data Streams: Models and Algorithms, vol. 31. Springer Science & Business Media. 2007.
[41] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting., Journal of Machine Learning Research 15 (2014) 1929–1958.
[42] L. Zhang, B. Zhang, A geometrical representation of McCulloch-Pitts neural model and its applications, IEEE Transactions on Neural Networks 10 (1999) 925–929.
[43] L. Prechelt, Early stopping-but when? Neural Networks: Tricks of the Trade, Springer. 2012, pp. 53–67.
[44] T.D. Wickens, Elementary Signal Detection Theory, Oxford University Press. 2001.
[45] W. Zhu, N. Zeng, N. Wang, et al. Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS® implementations, NESUG proceedings: health care and life sciences, Baltimore, Maryland, 2010. pp. 1–9.
[46] C. Ferri, J. Hernández-Orallo, R. Modroiu, An experimental comparison of performance measures for classification, Pattern Recognition Letters 30 (2009) 27–38.
[47] R.A. Berk, Classification and regression trees (CART), Statistical Learning from a Regression Perspective, Springer. 2008, pp. 1–65.
[48] I.R.A. Hamid, J. Abawajy, Hybrid feature selection for phishing email detection, International Conference on Algorithms and Architectures for Parallel Processing, Springer. 2011, pp. 266–275.
[49] A. Martin, G. Doddington, T. Kamm, M. Ordowski, M. Przybocki, The DET curve in assessment of detection task performance, Technical Report, DTIC Document. 1997.
[50] A. Adler, M.E. Schuckers, Calculation of a composite DET curve, International Conference on Audio-and Video-Based Biometric Person Authentication, publisher. 2005, pp. 860–868.
[51] R. Auckenthaler, M. Carey, H. Lloyd-Thomas, Score normalization for text-independent speaker verification systems, Digital Signal Processing 10 (2000) 42–54.
[52] L. Ma, B. Ofoghi, P. Watters, S. Brown, Detecting phishing emails using hybrid features, Ubiquitous, Autonomic and Trusted Computing, 2009. UIC-ATC'09. Symposia and Workshops on, IEEE. 2009, pp. 493–497.

Sami Smadi received a B.Sc. degree in Computer Science from Hashemite University, Jordan, and the M.Sc. degree in Computer Science from Yarmouk University, Jordan in 2006. He has recently completed his PhD from Northumbria University Newcastle, United Kingdom. His main research areas are network security, phishing detection, reinforcement learning, machine learning and artificial neural network.

Nauman Aslam is a Reader (Associate Professor) in the Department of Computer and Information Science, Northumbria University. Prior to joining Northumbria University as a Senior Lecturer in 2011, he worked as an Assistant Professor at Dalhousie University, Canada. He received his PhD in Engineering Mathematics from Dalhousie University in 2008. Dr. Nauman has research interests that include network security, wireless ad hoc sensor & body area networks, energy efficient protocols and application of Artificial Intelligence (AI) in communication protocols. Dr. Nauman is a member of IEEE and IAENG.

Li Zhang received a PhD degree from the University of Birmingham. She is currently an Associate Professor in Computer Science in Northumbria University, UK and serving as an Honorary Research Fellow in the University of Birmingham, UK. She holds expertise in artificial intelligence, machine learning, and intelligent robotics. She has served as an Associate Editor for Decision Support Systems. Dr. Zhang is a member of IEEE.