Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 199 (2022) 95–102

Procedia Computer Science

www.elsevier.com/locate/procedia

The 8th International Conference on Information Technology and Quantitative Management

(ITQM 2020 & 2021)

Development of a Real Time Smishing Detection Mobile Application using Rule Based Techniques

Oluwatobi Noah Akande[a]*, Hakeem Babalola Akande[b], Aderonke Anthonia Kayode[c], Adegun Adekanmi Adeyinka[d], Folorunsho Olaiya[e], Gbenle Oluwadara[a]

[a*]Computer Science Department, Landmark University, Kwara State, Nigeria
[b]Department of Telecommunication, University of Ilorin, Kwara State, Nigeria
[c]Department of Computer Science, Redeemer's University, Ede, Osun State, Nigeria
[d]School of Maths, Statistics, and Computer Science, University of Kwazulu-natal, Durban, South Africa
[e]Department of Computer Science, Federal University Oye-Ekiti, Nigeria.

Abstract

The introduction of alternative messaging platforms on mobile devices have not been able to phase off Short Messaging Service (SMS) as the most widely used means of textual communication. Over the decades, SMS has remained the most responsive way of communication that has been embraced by individuals and organizations in passing information across to their intended recipients. However, hackers have been employing this tool as a way to deceive the gullible into divulging sensitive information about their financial dealings as well as gain access to their mobile devices. A lot of innocent but ignorant individuals have become victims of this smishing acts and have lost huge sum of money as a result. Though existing research have extensively proposed and implemented different techniques for detecting and separating spam SMS from ham SMS, a mobile application that uses a rule-based RIPPER and C4.5 classifiers in detecting smishing acts is proposed. The rule-based classifiers were used to formulate rules used in detecting and separating spam from ham while a mobile application was developed to use the rule-based model in smishing detection. An Application Programming Interface (API) was designed to intercept incoming SMS, forward them to the rule-based model for analysis and then relay the results to the user via the developed mobile application. The user then decides to either retain or discard the SMS.

Keywords: Data Security; Smishing; Spam SMS Detection; Rule based Learning; Mobile Application Development

* Corresponding author.
E-mail address: akande.noah@lmu.edu.ng

1877-0509
10.1016/j.procs.2022.01.012

1. INTRODUCTION

Smart phones have become an inextricable part of our daily lives that we cannot do without. It has garnered 5.22 billion unique users worldwide with a growth rate of 93 million in the last 12 months [1]. Smartphone penetration in Africa rose from 198 million in 2016 to 498 million in 2020, with Nigeria expected to produce 145 million new mobile connections by 2025, ahead of South Africa and Kenya [2]. However, the most common feature of smartphones is the SMS; it is as old as mobile phone itself, yet it has remained the most efficient and the most widely used means of textual communication with 9.57 and 20.92 billion messages sent and received in 2018 alone [3]. Its relatively low cost, ease of construction with little or no technical know-how and the fact that it does not require internet connectivity for message delivery as made it a relevant and reliable instant messaging application. In addition, unlike emails, SMS has a 98% open rate, while email has 45 percent open rate [4]. In contrast, SMS has 45% response rate versus 6% for email. These are some of the reasons why its use has not been limited to individuals alone as organizations have also embraced it as a cost-effective means of reaching their customers. Organizations need to find the most cost effective medium of information dissemination that could be affordable by all. It was observed that the best medium of information dissemination must be information and communication technology-driven [5,6] and SMS fits into this. Therefore, it has been widely used by organizations in reaching out to their numerous staff and customers. However, its growth rate and wide acceptability has made it attractive to malicious individuals for illegal activities. Several gullible and innocent individuals have lost huge sum of money to smishing acts. While phishing acts is a fraudulent attempt aimed at obtaining sensitive information from targeted individuals, smishing is a form of mobile phishing attack in which hackers send SMS messages to users enticing them to visit fraudulent websites, applications, or links.

Smishing can include links to malicious applications or websites; this is done not only to steal money from users, but also to obtain information such as their contacts, images, and other data stored on their mobile devices. Furthermore, most mobile users are vulnerable to smishing attacks because of their device's small display scale, which makes it difficult for them to independently examine the contents of suspicious links. Also, most smartphone users are expected to grant malicious applications access to their emails, addresses, photos, and other insecure areas on their computer as part of the installation process. This approach allows hackers to easily obtain access to the contents of users' phones. According to a poll, SMS Spam affects 68 percent of mobile phone subscribers, with teenagers being the most vulnerable [3]. Several SMS messages containing malicious links have recently been forwarded, with some of the links installing malware on users' phones and stealing stored financial information or addresses. Furthermore, as a result of the recent use of SMS for two-factor authentication, people have become accustomed to receiving codes in SMS. In less than two weeks, $68,000 was stolen from 17 ATMs in Illinois, Michigan, and Ohio, and $3000 was withdrawn from a single individual [7,8]. Although many people believe that installing anti-malware software is a good idea, determining whether the anti-malware software isn't malware is still a challenge. Nevertheless, several techniques have been proposed in literature to tackle insecurity of sensitive data and information [9,10,11,12]. Nonetheless, in order to track and detect spam messages in real time, this research proposed a rule-based mobile application that intercepts and analyzes incoming SMS and alerts the user if the SMS is a spam or ham.

2. MATERIALS AND METHOD

The phases of implementation employed in this research include into data collection and pre-processing, rules formulation, model development and training using phyton-weka, evaluation of the rule-based spam detection model and mobile application development

2.1. Data Collection and Pre-processing

The developed spam detection model was trained and tested using the UCL SMS spam dataset v.1, which comprises 5547 text messages in English language (4827 ham messages and 747 spam messages). However, since the datasets cannot be used in their raw form since they contain many abbreviations and unstructured words, they were pre-processed using the WEKA programming method to improve spam identification. The dataset, which was collected in CSV format, was translated to ARFF format so that Weka could use it. The preprocessing stages involves case conversion, punctuation mark removal, abbreviation expansion, tokenization, stemming and stop words removal.

2.2. Rules Formulation using RIPPER Algorithm

The RIPPER algorithm (Repeated Incremental Pruning to Produce Error Reduction) is a rule-based classifier that derives its rules from the training set. The rule formulation process is divided into three stages: development, pruning, and optimization. With a priority on the SMS dataset, which has two classes: SPAM and HAM, the RIPPER classifier splits the dataset into two using a divide and conquer technique. The majority class is made up of the classes with the most features, while RIPPER seeks to learn features that separate the two classes by growing features from the minority class to the majority class. This raises the rule's specificity while decreasing its entropy; if a rise in specificity would not result in a decline in entropy, the rule is pruned. Iterating on the rising and pruning of the regulations before the rule is optimized to accurately define the features that classify SMS as spam or ham. As a termination condition, the minimum definition length principle was applied. The RIPPER algorithm's pseudocode is:

```
procedure BuildRuleSet (P, N)
        P = positive examples    N = negative examples
RuleSet = {}
        DL = DescriptionLength (RuleSet, P, N)
 while P6= {}
        // Grow and prune a new rule
        split (P, N) into (GrowPos, GrowNeg) and (PrunePos, PruneNeg)
Rule: = GrowRule (GrowPos, GrowNeg)
        Rule: = PruneRule (Rule, PrunePos, PruneNeg)
        add Rule to RuleSet
   if DescriptionLength (RuleSet, P, N) > DL+ 64 then
        // Prune the whole rule set and exit
        for each rule R in RuleSet (considered in reverse order)
                if DescriptionLength(RuleSet−{R}, P, N) < DL then
                        delete R from RuleSet
                        DL: = DescriptionLength(RuleSet,P,N)
                end if  end for
        return (RuleSet)
   end if  DL:= Description Length(RuleSet,P,N) delete from P and N all examples covered by Rule
end while  end BuildRuleSet
```

2.3. Rules Formulation using RIPPER Algorithm

C4.5 algorithm is an extension of ID3 algorithm that is good for classifying discrete and continuous variables. It is also good for pruning tress and handling missing values after tree construction. This is the reason why it was

used to complement any missing values that RIPPER algorithm could miss. So, the output of RIPPER algorithm serves as input to C4.5 algorithm. C4.5 has additional features such as managing missing values, continuous attribute categorization, decision trees pruning, rule derivation and others. C4.5 constructs a very large tree by taking into account all the attribute values and finalizes by pruning the decision rule. The C4.5 algorithm is a variant of the ID3 algorithm for classifying discrete and continuous variables. It's also useful for pruning trees and coping with lost values after they've been installed. This is why it was used to fill in any incomplete values that the RIPPER algorithm may have overlooked. As a result, the RIPPER algorithm's output is fed into the C4.5 algorithm. Missing value control, continuous attribute categorization, decision tree pruning, law derivation, and other functionality are included in C4.5. C4.5 builds a very wide tree by taking into account all attribute values and pruning the judgment rule at the end. It employs a heuristic pruning strategy based on split statistical meaning. C4.5 PART is a Weka variant of the C4.5 algorithm that creates a partial C4.5 decision tree in each iteration and turns the "best" leaf into a guideline. In addition, each rule in the list is compared to new data, and the object is given the first rule class that matches. Following are some of the rules that the algorithm derived after training the dataset with the C4.5 PART model:

```
1: Create a root node N;
2: IF (T belongs to same category C)
        {leaf node = N;
        Mark N as class C;
        Return N;     }
3: For i=1 to n
        {Calculate Information_gain (Ai);}
4: ta= testing attribute;
5: N.ta = attribute having highest information_gain;
6: if (N.ta == continuous)
        {find threshold;}
7: For (Each T in splitting of T)
8: if (T is empty)
        {child of N is a leaf node;}
        else
        {child of N= dtree T)}
10: calculate classification error rate of node N;
11: return N
```

2.4. Feature Extraction using RIPPER and C4.5 Rule-based Approach

The problem with spam SMS identification is that there aren't enough features to reliably identify them as ham or spam. A legitimate SMS could include URLs, phone numbers, account numbers, and terms like call, text, prize, and so on, but spam SMS often includes these features. As a result of this study, dictionaries of commonly used spam and ham terms, smishing URLs and domains, and smishing contact numbers were developed. These were used to verify URLs, domains, and phone numbers in SMS messages. The following are some of the rules that were generated using if...then commands:

i. If the SMS includes a URL component, check it against a phishing URL dictionary; if the test returns positive results, it's spam; otherwise, it's ham.
ii. If the SMS includes contact numbers, run a scan against a contact number dictionary; if the results are positive, the message is spam; otherwise, ham
iii. If the SMS includes contact numbers, run a scan against a contact number dictionary; if the results are positive, the message is spam; otherwise, ham
iv. It is a spam SMS if the number of spam words in the SMS is greater than or equal to two. It's a ham SMS,

v. If number of ham words in SMS is greater or equal to 4, it is a ham SMS
vi. It is a spam SMS whether it includes some symbol such as "$", "£", "#", or "&" and the number of spam words in the SMS is greater than or equivalent to 2.
vii. If the SMS includes some mathematical symbol such as +, -,,>, /, etc., and the number of spam terms in the SMS is greater than or equal to 2, it is considered spam.
viii. It's a ham if the number of emoticons in the SMS is greater than or equal to two.
ix. If the SMS is of the self-answering kind, it is spam.
x. If the SMS is of the self-answering kind, it is spam. It's a ham if the SMS includes abbreviations.

2.5. Mobile Application Development

Using the proposed rule-based SMS spam detection model, an android based mobile application was developed to employ the model in preventing smishing attacks. The mobile application was specifically developed to intercepts incoming SMS to a mobile phone, forward the SMS to the rule-based SMS spam identification model using an API, determine if the intercepted SMS is spam or not then label the SMS as spam or not and notify the recipient using the created smartphone application. The mobile application was created using the Java programming language, and a mobile service was created using Okhttp to capture incoming SMS. Okhttp is a Java application that lets programmers make HTTP requests. The intercepted SMS is then saved in json format and sent to a rule-based SMS spam detection model through a Flask Restful API. Flask is a lightweight Web Server Gateway Interface (WSGI) that acts as a bridge between web applications and web servers, while Flask Restful is a flask extension that allows programmers to quickly create REST Application Programming Interfaces (APIs). The API's appearance is determined by REST. "Representational State Transfer" is what it stands for. It's a series of guidelines that developers would meet when creating APIs. One of the rules specifies that anytime a link request is made to a given URL, a piece of data known as a property should be returned. Each URL request allows a request for the resources on the server associated with that address. As a result, the API was used to provide a communication channel between the SMS spam detection model and the created mobile application. As a result, after the SMS has been processed, the classification result (spam or not spam) will be sent to the mobile app with an Okhttp post request.

3. RESULTS AND DISCUSSION

The proposed rule-based SMS spam identification models were tested using TP rate, FP rate, Precision, Recall, and F-Measure, among other metrics. In addition, the created mobile application was run in real time to see if incoming SMS is intercepted, evaluated using the proposed templates, and labeled as spam or not spam.

3.1. Evaluating the Performance of the Rule based SMS Spam Detection Models

The performance of the RIPPER classifier was carried out using TP rate, FP rate, Precision, Recall, F-Measure etc. the results obtained are:

```
Evaluating JRip classifier on spam with cross fold validation
============================================================
[0. 0. 1. ... 0. 0. 0.]

Correctly Classified Instances        5467               98.0804 %
Incorrectly Classified Instances       107                1.9196 %
Kappa statistic                          0.914
Mean absolute error                      0.0369
Root mean squared error                  0.1358
Relative absolute error                 15.8813 %
Root relative squared error             39.8596 %
Total Number of Instances             5574

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.996    0.116    0.982      0.996    0.989      0.915    0.941     0.982     ham
                 0.884    0.004    0.971      0.884    0.925      0.915    0.941     0.895     spam
Weighted Avg.    0.981    0.101    0.981      0.981    0.980      0.915    0.941     0.971
```

It can be seen that the correct classification rate was 98.08 percent (5467 instances), while the incorrect classification rate was 1.92 percent (107 instances). With precision and recall rates of 0.982 and 0.989, this obtained a TP rate of 0.996 and an FP rate of 0.116. The 1.92 percent of wrongly labelled instances were due to missing values, which affected our decision to use the C4.5 algorithm to catch the missing values. Evaluating the performance of the C4.5 PART algorithm using TP rate, FP rate, Precision, Recall, F-Measure etc achieved the following results:

```
Evaluating PART classifier on spam with cross fold validation
============================================================
[0. 0. 1. ... 0. 0. 0.]

Correctly Classified Instances        5486               98.4212 %
Incorrectly Classified Instances        88                1.5788 %
Kappa statistic                          0.9305
Mean absolute error                      0.0287
Root mean squared error                  0.1198
Relative absolute error                 12.3617 %
Root relative squared error             35.1665 %
Total Number of Instances             5574

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.995    0.084    0.987      0.995    0.991      0.931    0.983     0.995     ham
                 0.916    0.005    0.965      0.916    0.940      0.931    0.983     0.959     spam
Weighted Avg.    0.984    0.074    0.984      0.984    0.984      0.931    0.983     0.990
```

The number of correctly categorized instances was found to be 98.42 percent (5486 instances), which was higher than the 98.08 percent (5467 instances) obtained by RIPPER, indicating that RIPPER was more accurate. In addition, 1.58 percent of the instances were wrongly labelled, compared to 1.92 percent when RIPPER was used. The C4.5 algorithm has obtained a TP rate of 0.995, an FP rate of 0.084, and precision and recall values of 0.987 and 0.995, respectively. This demonstrated that the C4.5 PART algorithm greatly enhanced the rule-based technique's efficiency.

3.2. Evaluating the Performance of the Mobile Application

To replicate the success of the proposed mobile application in real life, an android emulator included with android studio was used. Fig. 1 depicts the Flask Restful API interface, which allows SMS to be sent to the mobile app. Fig. 2 shows how a SMS was intercepted and forwarded for review to a rule-based spam detection model. The

results of the review are then sent to the smartphone application, where the recipient will determine whether or not the SMS is spam. The mobile application has received the information. The result of the analysis as it will be displayed to the user is shown in Fig. 3.

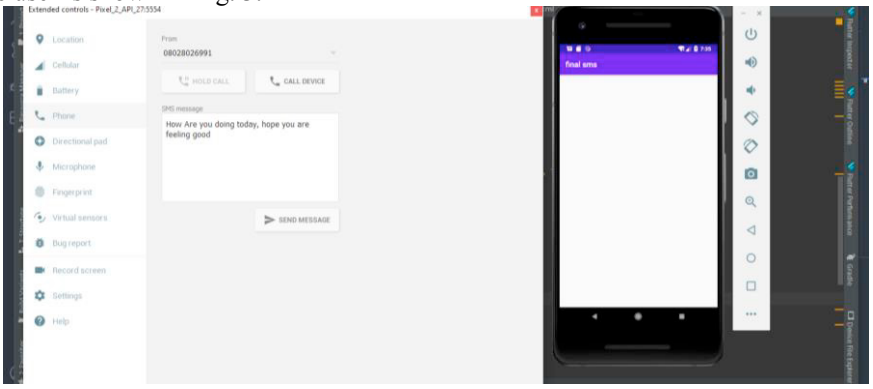

Fig. 1. Flask Restful API Interface

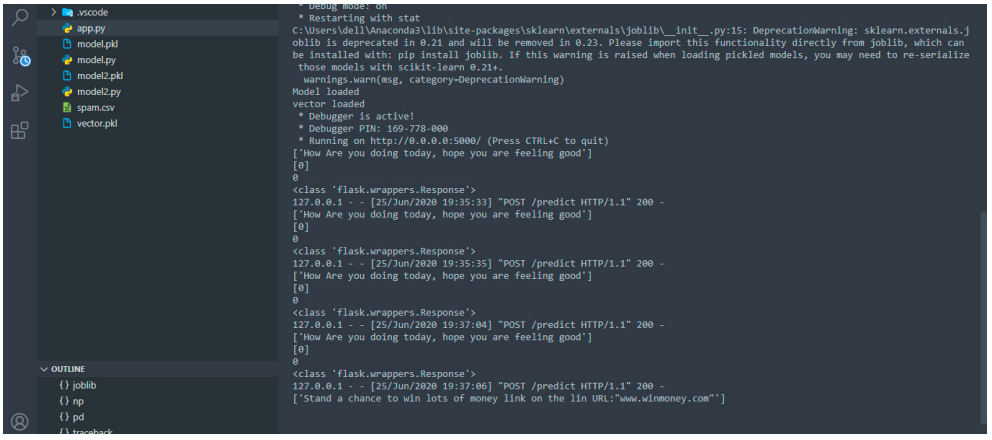

Fig. 2. Model Intercepting, Analyzing and Forwarding the Analysis Result to the Mobile Application.

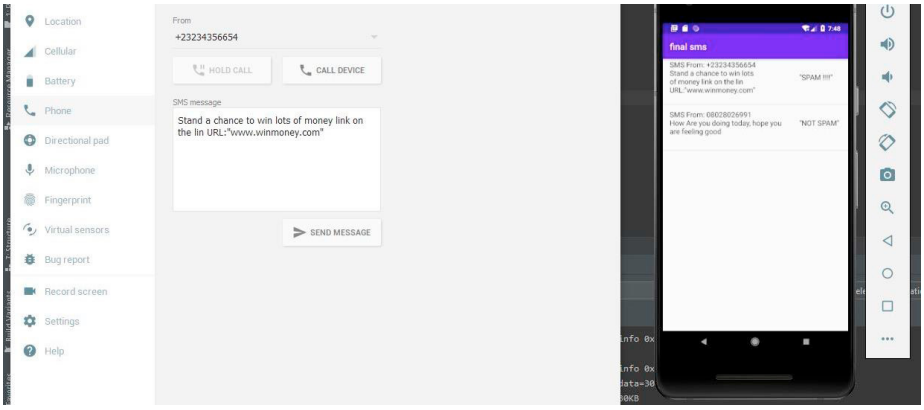

Fig. 3. Mobile Application with SMS labelled as SPAM and NOT SPAM

4. CONCLUSION

Smartphones have evolved into a mechanism that completes our everyday activities, with many people unable to function without them. SMS is the most commonly used feature on a smartphone out of all the options. Despite the introduction of various alternative messaging systems, SMS remains the fastest, most commonly used, and feature with the highest response rate. As a result, hackers have identified SMS as a mechanism that can be used to commit smishing crimes. While many methods for identifying and distinguishing spam from ham have been suggested, most studies to date have not shown how the proposed models will be used to potentially deter smishing in practice. As a result, this study has shown how to diagnose and avoid smishing using a rule-based classification methodology. The rule-based spam identification program used RIPPER and C4.5 classification algorithms. Following that, a mobile service was created to catch and forward incoming SMS to the rule-based spam detection app. The analysis' outcome was then communicated to the consumer using the built mobile application. The customer will now decide whether or not to agree or deny the study' findings. The proposed smartphone framework can be used to track and deter smishing activities on mobile devices.

Acknowledgements

Authors appreciate Landmark University Centre for Innovation and Development for sponsoring the publication of this article.

References

[1]. Smartphone Users in Nigeria, https://www.statista.com/statistics/467187/forecast-of-smartphone-users-in-nigeria/
[2]. GSMA Intelligence (2021). Retrieved January 12, 2021 from https://www.gsmaintelligence.com/
[3]. The Real Value of SMS to Businesses (2021). Retrieved January 12, 2021 from https://www.smscomparison.co.uk/sms-gateway-uk/2018-statistics/.
[4]. Chris Pemberton (2016) Tap into the Marketing Power of SMS accessed on 15th February, 2021 available at: https://www.gartner.com/en/marketing/insights/articles/tap-into-the-marketing-power-of-sms
[5]. Filip, F. G., Zamfirescu, C. B., & Ciurea, C. (2017). Collaboration and Decision-Making in Context. In Collaboration and Decision-Making in Context. In: Computer-Supported Collaborative Decision-Making. Automation, Collaboration, & E-Services (pp. 1–30). https://doi.org/10.1007/978-3-319-47221-8
[6]. Nunamaker Jr, J.F., Briggs, R.O., & Romano Jr, N.C.R. (2014). Collaboration Systems: Concept, Value, and Use (1st ed.). Routledge. https://doi.org/10.4324/9781315705569
[7]. SMS Phishing + Cardless ATM = Profit https://krebsonsecurity.com/2018/11/sms-phishing-cardless-atm-profit/ accessed on 30th June 2020
[8]. Yang, Y., Hu, R., Qiu, C., & Sun, G. (2018). A Spam Message Detection Model Based on Bayesian Classi fi cation. Lecture Notes on Data Engineering and Communications Technologies, 2, 424–435. https://doi.org/10.1007/978-3-319-59463-7
[9]. Akande O.N., Abikoye O.C., Kayode A.A., Aro O.T., Ogundokun O.R. (2020) A Dynamic Round Triple Data Encryption Standard Cryptographic Technique for Data Security. In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2020. ICCSA 2020. Lecture Notes in Computer Science, vol 12254. Springer, Cham. https://doi.org/10.1007/978-3-030-58817-5_36
[10]. Abikoye Oluwakemi Christiana, Abubakar Abdullahi, Ahmed Haruna Dokoro, Akande Noah Oluwatobi (2020), Kayode Anthonia Aderonke, A Novel Technique to Prevent SQL-Injection and Cross-Site Scripting Attacks using Knuth-Morris-Pratt String Matching Algorithm, EURASIP Journal on Information Security, Vol. 14, Pp. 1-14. https://doi.org/10.1186/s13635-020-00113-y
[11]. Abikoye Oluwakemi Christiana, Benjamin Aruwa Gyunka, Akande Noah Oluwatobi (2020), "Optimizing Android Malware Detection Via Ensemble Learning", International Journal of Interactive Mobile Technologies (iJIM), Vol. 14, No. 9, pp. 61-78. https://doi.org/10.3991/ijim.v14i09.11548
[12]. Abikoye Oluwakemi Christiana, Haruna Ahmad Dokoro, Abdullahi Abubakar Akande Noah Oluwatobi, Asani Emmanuel Oluwatobi (2019), "Modified Advanced Encryption Standard Algorithm for Information Security", Symmetry, Vol. 11, No. 12, Pp. 1-17.