Development of Basic Web Spambots Android Application

Khushboo Sharma[1] Rishabh Patel[2] Sahil Bhatia[3]

[1,2,3]Galgotias College of Engineering and Technology, Greater Noida, Uttar Pradesh, India

Abstract— Internet is the global system that connects billion of devices wirelessly worldwide by following the Internet Protocol suite (TCP/IP). One of the main and solemn threats on the Internet is Spam. Spam refers to the abuse of electronic messaging system by sending unrequested bulk messages desultorily. Botnets are considered one of the main contributors to the sources of spam. Botnet refers to a group of software called bots. The function of these bots is to run on several compromised computers autonomously and automatically. Spamming causes illicit consumption of network resources in general and mail system in particular. A Spambots or Web Spambots harvest e-mail addresses found on web pages over internet in order to build mailing list. These Spambots are the type of web crawler with added functionality of sending mails to the email addresses mined from the web pages. The task of this research is to design the Web Spambots android application that runs on any android device running Kitkat (4.4.0) or ameliorated version. The development include and describe the three main steps pre-processing, pattern finder and expression extraction.

Key words: Internet, WWW, Email, Spam, Web Spambots, Web Crawler, E-Miner, Android, Rskbots

I. INTRODUCTION

21 century is the era of getting digital. Every human on this planet is addict of fast and agile service of internet and internet related technologies. One of the best known service over this internet is email and special technique of extracting emails from the World Wide Web (WWW) is email miner (also known as e-miner). This email miner is the type of web crawler that crawl the WWW to extract the emails and broadcast a message to these extracted emails. This full process is known as Spambots (or generally Web Spambots). Our project Development of Basic Web Spambots Android Application demonstrates the development and working of Web Spambots on the android platform as an app called RSKBots and hence review the behaviour of Spambots.

RSKBots an android app, preforms email mining on the specific web page given by the user and then sends email to all the recipients extracted from that specific page and to the members which are present in the cloud database that are extracted previously from other web pages on the user command. This app is developed on the ellipse's mars version that will runs on android devices that have KitKat (4.4.0) or ameliorated version of android. The app uses API of online database system called backendless that stores email addresses in the tabular form. A test email id that is used by RSKBots to send to send emails to other is rskbots@gmail.com which gets possible only by using API of the Google's Gmail service. To run the app one should connect his android device with the high speed internet.

Algorithm that used in this application is crawling the web surface only. It means that app will crawl only that web page which is input by user in the app. It does not dig the page deep into its core to extract more email. This is because the app needs the faster internet connection and also needs comparatively much more time as compare to that of surface crawling.

This paper discuss and describe the development of the three main phases of the Web Spambots that are

1) pre-processing,
2) pattern finder,
3) expression extraction,

and the last step of consideration i.e. broadcasting mails to all the email addresses extracted by the e-miner.

II. PHASES

The first phase after getting the web page address is the pre-processing. Pre-processing refers to the extraction of the HTML code of the provided web page. In the app RSKBots function getInternetData() extract the HTML code from the give we page. It stores whole code in the string variable data after some alterations. After getting an organised code as required it return the variable data for pattern finding and extraction or it can also generate the error if there is any problem in network.

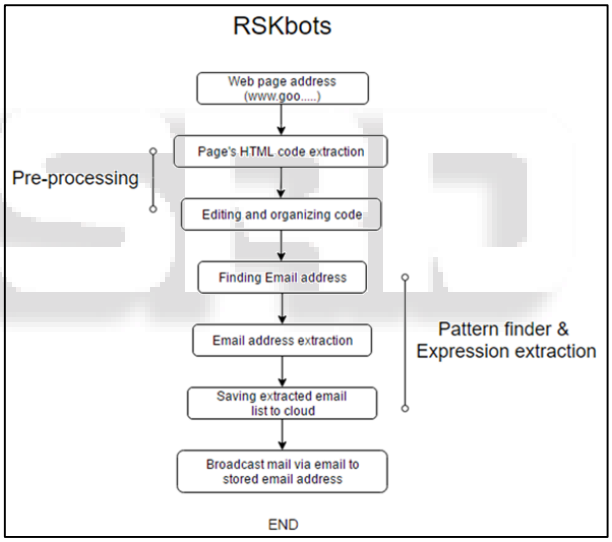

Fig 1: Architecture of RSKBots.

The code of the function getInternetData() is as follow:

```
getInternetData() {
   BufferedReader in = null;
   String data = null;
   HttpClient client = new DefaultHttpClient();
   URI website = new URI(URL);
   HttpGet request = new HttpGet();
   request.setURI(website);
   HttpResponse response =  client.execute(request);
   response.getStatusLine().getStatusCode();
   in = new BufferedReader(new
InputStreamReader(response.
   getEntity().getContent()));
   StringBuffer sb = new StringBuffer("");
   String l = "";
   String nl = System.getProperty("line.separator");
   while ((l = in.readLine()) != null) {
```



```
        sb.append(l + nl);
    }
    in.close();
    data = sb.toString();
    return data;
}
```

Next important step involved in mining are pattern finder and expression extraction. Pattern finder, as cleared by its name, searched the pattern of string needed and expression extraction extract the searched pattern. In RSKBots, after getting the required organized code pattern finder searched for the email addresses and with the help of expression extraction emails are fetched. As both these steps are contiguous and their working is simultaneous that is one after another thus, they involved in same function by the name GetEmailsFromString() within the app. Thus the function GetEmailsFromString() extract all email addresses embedded in the web page and return the string variable named list which contains the list of emails. This list is saved in cloud database named backendless in tabular format along with the address of the web page before returning the variable list in the same function GetEmailsFromString(). These emails are displayed in the app as well. If on emails are found than the function return the same variable list valued as null and it will generate the pop up title in the screen saying "No e-mail addresses found". The function GetEmailsFromString() is coded as :

```
GetEmailsFromString(String s) {
    list1.clear();
    Matcher m = Pattern.compile("[a-zA-Z0-9_.+- ]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-.]+").matcher(s);
    int i=1;
    while (m.find()) {
        System.out.println("email loop" + m.group());
        list1.add(m.group().trim());
    }
    return list1;
}
```

The final step of consideration is to send the mails to all the email addresses that are stored in the database. The function sendemailtocandidate() in RSKBots send mail to all the email addresses that are extracted from the web page as well as to the other recipients that are stored in the database extracted previously from other web pages. App uses Gmail email id for sending the mail that is rskbots@gmail.com. The mail send to the recipients are not to harm them. It is only a test mail that says "This is a test e-mail. The quick brown fox jumps over the lazy little dog." and have the subject as "Sample e-mail!!" Function sendmailtocandidata() is coded as follows:

```
sendemailtocandidate(String mail) {
GMailSender sender = new GMailSender(Email, password);
sender.sendMail(subject, body, senderEmail, mail);
System.out.println("Result Value is try" + mail);
}
```

III. Complexity

The application has no loops and it only parse the emails ids. As no function in the RSKBots has loops thus complexity of the all functions in the application will be constant as K. App only parse all the mail address saved in the cloud database backendless linearly thus the complexity equivalents n. Where n represents the number of email address in database. So complexity of the whole application RSKBots is

Complexity,

$$T(n) = n + K,$$

$$\text{So, } T(n) = O(n)$$

K: complexity of all functions in app

n: complexity for parsing the email addresses from database

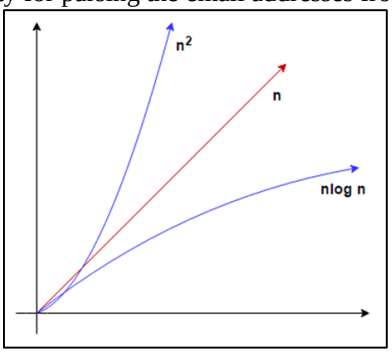

Fig. 2: Complexity of app shown by red curve

IV. Conclusion and Future Work

For this generation internet is vital to connect, communicate, and perform many different tasks in everyday life. Having different businesses using the cloud and expanding connectivity, on the one side, and increasing the contacts for business on other. This whole paper talked about the development and behaviour of the Web Spambots. There are many limitations in the app which can be further amended and reviewed according to the different users. Future scope include the improvement of the app RSKBots in terms of

1) Sending mails to selected members,
2) Using more optimize depth in crawler,
3) Signing the app with one's own email address,
4) Editing ones owe database and
5) Editing the subject and the body of the mail

References

[1] Pedram Hayati*, Kevin Chai, Vidyasagar Potdar, and Alex Talevski, HoneySpam 2.0: Profiling Web Spambot Behaviour, In Principles of Practice in Multi-Agent Systems: Springer Berlin Heidelberg, 2009.

[2] Vida Ghanaei, Costas S. Iliopoulos, Solon P. Pissis, Detection of Web Spambot in the Presence of Decoy Actions, In IEEE Fourth International Conference on Big Data and Cloud Computing, 2014

[3] Mohammed Fadhil Zamil, Ahmed M. Manasrah, Omar Amir, Sureswaran Ramadass, a behaviour based algorithm to detect spambots, In IEEE Fourth International Conference on Computer Communications, 2010

[4] Pedram Hayati, Vidyasagar Potdar, Kevin Chai, Alex Talevski, Web Spambot Detection Based on Web Navigation Behaviour, In 24th IEEE International Conference on Advanced Information Networking and Applications, 2010

[5] Manish Saxena, P. M. Khan, Spamizer: An Approach to Handle Web Form Spam, In 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015

[6] Nilani Algiriyage, Sanath Jayasena, Gihan Dias, Amila Perera, and Kushan Dayananda, Identification and Characterization of Crawlers through Analysis of Web Logs, In IEEE 8th International Conference on Industrial and Information Systems, ICIIS 2013, Aug. 18-20,2013, Sri Lanka, 2013
[7] Miloš Pavković, Prof. Jelica Protić, Intelligent Crawler for Web Forums based on Improved Regular Expressions, In 21st IEEE Conference on Telecommunication, 2013

