Online social networks (OSNs), such as Facebook and Twitter, allow users to tweet or post short update messages reporting on their daily activities. The proposed system provides a platform to alarm against the spread of new malware attacks such as viruses, worms, or Trojan horses using the OSNs. Currently, network administrators and operators use manual and traditional ways of communication, such as phones and e-mails to warn each other against such attacks. Instead, a prototype system is proposed that mines Twitter posts to provide real-time alerts of malware propagation (Twitter is taken as our target to implement this system).

The system is composed of four important modules: a) Data extracting where system continually and periodically queries Topsy’s APIs (mechanism to get twitter logs) for specific keywords such as Malware, backdoor, and cyber-attack then returns the results in JSON format. b) Data filtering system that extracts the tweets that contain one of the following phrases: computer security, new, discovers, hit, infect, warn, and watch out. c) Smoothing the data implement “Exponentially Weighted Moving Average (EWMA) algorithm and Exponentially Weighted Moving Variance (EWMV)”, which identifies threshold based on time intervals associated with the keywords. d) Finally, a malware alert is triggered when the actual number of tweets in a given interval exceeds the threshold value.
# TABLE OF CONTENTS

Abstract ........................................................................................................................................ ii

Table of Contents .................................................................................................................... iii

List of Figures ........................................................................................................................ vi

List of Tables .......................................................................................................................... viii

1. Background and Rationale ..................................................................................................... 1
   1.1 Historical approach ......................................................................................................... 6
      1.1.1 CANTINA ........................................................................................................... 7
      1.1.2 Netcraft ............................................................................................................. 7
      1.1.3 SpoofGuard ......................................................................................................... 7
      1.1.4 Using social Network ......................................................................................... 8
   1.2 Malwares ....................................................................................................................... 8
      1.2.1 Virus .................................................................................................................. 8
      1.2.2 Worm ................................................................................................................ 8
      1.2.3 Trojan horse ....................................................................................................... 8
      1.2.4 Logic Bomb ........................................................................................................ 9
      1.2.5 Backdoors ......................................................................................................... 9
      1.2.6 Spy ................................................................................................................... 9

2. Previous Research, Motivation ............................................................................................... 10
   2.1 Anomaly-based Detection ............................................................................................ 10
      2.1.2 Signature-based Detection ................................................................................... 11
   2.2 Application of CLIPS Expert system ......................................................................... 11
LIST OF FIGURES

Figure 1.1 Graph showing the Social Media Use in each country ..........................4
Figure 1.2 Frequency of social media site use..................................................5
Figure 1.3 Graph showing number of monthly active Twitter users Worldwide........6
Figure 2.1. Framework of CLIPS expert system for Malware Detection...............12
Figure 2.2 System Overview.............................................................................12
Figure 2.3 Process Modules.............................................................................14
Figure 3.1 Proposed System Architecture..........................................................17
Figure 3.2 Web application architecture.............................................................18
Figure 3.3 Main Use Case Diagram.....................................................................19
Figure 3.4 Sequence diagram for user authentication.........................................20
Figure 3.4.1 Sequence diagram for user authentication and data extraction........21
Figure 3.5 Topsy’s API – for social search.........................................................23
Figure 3.6 Flow Chart Diagram for the Proposed System.................................24
Figure 3.7 Work flow Diagram..........................................................................25
Figure 3.8 Showing the XAMPP 1.8.3-4 Application..........................................26
Figure 3.9 Shows the Manage Servers window on XAMPP..............................27
Figure 4.1 Home page for Malware Alert System...............................................29
Figure 4.2 Registration Form page......................................................................30
Figure 4.3 Login page.........................................................................................31
Figure 4.4 Welcome Page....................................................................................32
Figure 4.5 Data Extraction..................................................................................34
List Of Tables

Table 3.1 Keyword used for data extraction and data filtration…………………………38
BACKGROUND AND RATIONALE

Online social networks (OSNs), such as Facebook and Twitter, allow users to tweet or post short update messages reporting on their daily activities. The social networking sites allow people to communicate and connect with one another. One of the examples of networking sites is Twitter-an online community of mostly students and celebrities in the United States and all over the world. If Twitter were a country, it would be the eighth largest country in the world. Unfortunately, the propagation of the viruses and malware has also continued to be on the rise same as the population of networking sites [8].

Malevolent users and writers have found the size of the social network as a natural propagation of their mischievous activities and spreading the attacks over the network using the traditional ways [9].

The objective of this project is to implement a prototype for the malware alert system for the social network using Twitter as an example, and Topsy’s API for social database. With this malware alert system, it has the capability of warning the users from preventing malware attacks on the network and to overcome the fraud, which happened recently on social networks. Some of the recent worst attacks include:

Pornspace: This malware took the advantage of security flaws in MySpace, released a worm on profiles for password stealing and sends the porn-based spam to users, which redirected to illegal sites over the network.

My webcam thingy: My webcam thingy is another example for such malware attack. This malware attacked nearly thousands of Twitter accounts and tweeted the followers to
check the webcam link of a female, which redirects the users to a portal and stole the passwords and credit card information from the users.

Twitter Worm: This malware as the name indicated, generally spreads through the twitter network. This twitter worm has many versions, one of the versions is a Profile Spy worm: This type of worm is the version of twitter worm, which spreads by downloading the application from the links of another party application. This helps the attacker to get the personal information of the user and his followers once he downloaded the application and uses it.

Koobface: Koobface is a kind of malware that is known by everyone who is on social networks like Facebook and Myspace, usually, this malware is propagated between the two users i.e. with the messages. The message consists of video links, once the user check out the links in the messages, he will be re-directed to another site asking him to download the anti-virus software or any other software. Once the software is installed on the user system, the attacker steals the information and may also attack other computers.

The importance of the project

Some of the advantages of this project are: As the present online social networking is emerging to a higher level of importance in everyone’s daily life, researchers can use this opportunity for alerting other computers on the network about the malware. The proposed real-time malware alerting system, which collects data from Topsy’s API or human generated notices i.e. Tweets, can help in reducing the risk of harmful malware and viruses by triggering an alarm against such threats using this online social network (OSN).
Online Social Network: The present human connectivity using the social networks, which include Facebook, Twitter, Google+, Instagram, and YouTube, amounts to around 1 billion users.

This high amount of data from the social networks include information such as date of birth, gender, names, web sites, phone numbers, email addresses, usernames and password, personnel and employment history, current address and general activities of the user. The user may also post or tweet regarding his mood or present status or any updates which indeed reveal a lot about him, which can be of great interest to the attackers or any organizations.

Figure 1.1 shows the graph of social media usage all over the world compared to each country's population. The graph shows that the global average of social media usage is around 29%. Where the highest percentage goes to around 66% and lowest is 7%.
Figure 1.1 Graph showing the Social Media Use in each country [19]

Comparing different social media and their number of active users and the frequency of the social media site use on a daily basis gives a clear explanation of how social media are utilized by the users all around the world. Figure 1.2 shows the percentage of social media site users who use the sites with different frequencies on a daily basis.
Figure 1.2 Frequency of social media sites use [20]

Twitter as a social media: How well are people connected with twitter? How well the data is shared among the users via re-tweets. Twitter is a micro blogging service, which can be described as both the networking site and social media news commanding more than 41 million users. In Twitter, the users usually tweet about any topic, within the strict range of 140-character limit and there consists of one user who follows the other to receive their tweets. Every day, the users all around the world tweet regarding the current
affairs and trending topics, such as Malaysian Airlines plane crash or elections in U.S.

Figure 1.3 Graph showing number of monthly active Twitter users Worldwide [20]

Figure 1.3 gives the number of active Twitter users worldwide. This microblogging service averaged at 288 million active users all over the world explains how many tweets can be tweeted daily and how such information circulates in social media.

1.1 Historical approaches: Malware research has taken from the time personal computers were invented. In 2004, researchers from the Stanford University released the plugin for the Internet browsers to find whether the site is fraud or not. Below are some of the examples of tools for malware detection.
1.1.1 CANTINA:

CANTINA [11] is a security toolbar for Internet Explorer which is based on the PILFER algorithm. When the CANTINA tests on a database of 200 fraudulent sites, it detects around 190 fraudulent sites i.e., a limitation of the CANTINA cited that it shows the sites written in English. This CANTINA toolbar uses the Term frequency and other two algorithms to generate keywords lists on the site, which can be used by the Google for domain determination.

1.1.2 Netcraft:

The Netcraft Toolbar [3] is one of the toolbar which detects the phishing sites, it is similar to CANTINA which is a security plug-in for Internet Explorer. The advantage of the Netcraft toolbar is that it is unsusceptible to DNS poisoning. It resolves all the URLs present into their IP addresses for the analysis of the domain. And it completely depends on the visual cues in the toolbar. Therefore, users may miss the warning given by the Netcraft.

1.1.3 SpoofGuard

Spoofguard is also one of the browser plug-in for the Microsoft Internet Explorer, it prevents the malicious attacks like web spoofing. The SpoofGuard is a user-defined setting which aids in setting the parameters. It saves the data and alerts the system. As it is a user defined setting, it alarms the indicator, which depends on the user parameter settings.
1.1.4 Using Social Network:

Using social network, for example Facebook that have millions of users increasing every year and have a huge database of users can also help the researchers and developers to construct an application to detect the malware on the network. Unfortunately, some of the limitations of the Facebook include the access of Facebook API related and Non-API related.

1.2 Malware:

Malware's are of different kinds: based on attacks and behaviors, some of the examples include viruses, worms, spywares, and rootkits.

1.2.1 Virus:

A virus is a code which attaches itself to other programs such as operating systems and affects them by corrupting or destroying data.

1.2.2 Worm:

A worm is different from a virus, viruses publishes themselves with the help of programs while worms publishes themselves from system to system in a network and tries to pollute the network.

1.2.3 Trojan horse:

Trojan horse is a kind of malware that appears in the form of pieces of software code which are taken for useful purposes. It acts like a useful function for users, but secretly runs its actions beside them. It may cause major problems to the system.
1.2.4 Logic Bomb:

A logic bomb is not the same as other malware's that publish themselves, but it can install itself on a system and waits for a special event to occur, such as modifying or deleting of a special file, which can then lead to system damage.

1.2.5 Backdoor's:

Backdoor's enters into the system without the user authorization and does its work by damaging the files in the system.

1.2.6 Spy:

Spy is a different kind of malware which collects the user’s data such as browser history, keyboard actions and other important credentials.
2. PREVIOUS RESEARCH, MOTIVATION

Present research on the Online Social Networks (OSN) is divided into two categories: one is related to the social type of research which consists of relation and communications, and the second type is related to mining the data from the posts or tweets from the users [10]. Our present research is based on the second type of research that deals with the digital viruses, malwares and is related to cyber pendant.

Malware Detection:

Malware detection process is the implementation of some malware detection techniques. This malware detection helps to protect the system by detecting various malware and malicious behavior. Techniques involving detecting malware can be categorized into two: anomaly-based detection and signature-based detection. This specification-based technique is referred by anomaly-based detection. These signature-based and anomaly-based detection techniques are further categorized into three types: static, dynamic, and hybrid.

2.1 Anomaly-based Detection

Anomaly-based detection occurs in two phases, one is training phase, which is also known as learning phase. The second phase is the detection phase also known as monitoring phase. During the first phase, Training parts the detector and tries to learn the general behavior of the user. The advantage of the anomaly-based detection is the ability of the detector to detect the zero day attacks (attack that exploits unknown vulnerability in a Operating System). The limitations of the anomaly-based detection technique are: 1) its high false alarm rate and
2) It determines the features that should be learned in the first phase of the training phase. This anomaly-based detection technique alone is not sufficient for detecting the malware.

2.1.2 Signature-based Detection

This signature-based detection technique detects the malware by using the malware behavior as a model. This collection of the models for detecting the malware is known as signature-based detection and the behavior of the malware is referred to as a signature.

2.2 Application of CLIPS Expert System (Zhou Ruili, Pan Jianfeng, Tan Xiaobin, Xi Hongsheng 2008)

CLIPS expert system is an application, which detects the malware using the low level techniques. The malicious behavior of the malware is used as a key method to detect the malicious programs. This expert system detects the malware using the five different main parts as seen in figure 2.1. The five different main parts include Knowledge base, which consists of deductive rules, the second part is Behavior gathering component, which collects the data, the third one is the fact list, which consists of facts template, the fourth one is Inference engine, which is used for matching the patterns with the fact lists and the last part is User interface, which is used by user to interact with the system [22].
The drawback of the system is that it can only detect the malware, which is running. The system cannot detect any malicious program if the malware is not active.

2.3 Warning Bird (Sangho Lee, and Jong Kim 2013)

Warning Bird is a detection system, which detects the suspicious URL’s for Twitter. The Warning Bird detection system different from other conventional detection systems, which depends on multiple redirect chains. Beside other conventional system, which uses the concept of malicious landing pages [22]. Figure 2.2 shows the system overview.
The drawback of this detection system is this Warning Bird Real time detection system detects only the suspicious URLs in the Twitter.

2.4 Empirical Evaluation (Chao Yang, Robert Harkreader, and Guofei Gu 2013)

Empirical evaluation to detect the Twitter spammer is a new design in detecting the spammers in Twitter using in depth analysis. The design uses the state of the art solutions for examinations. The system is very effective in detecting the spammers in twitter accounts using the datasets provided [24].

The Empirical Evaluation system in detecting the spammers in twitter has some drawback, as it is nearly impossible to collect huge datasets from such a dynamic online social network real world.

2.5 A social Approach to security (Michael Robertson, Yin Pan, and Bo Yuan)

The social approach to security is an application developed to detect the malicious web content using the heuristics based on social networking data for example Facebook. This research on social network can detect the malicious web content by 34% success rate. The figure 2.3 process modules shows the how a URL link is testes using different applications and systems. As success rate is important for any system, future work can be done in improving in many ways [25].
2.3 Process Modules [25]

2.6 Motivation

In the past years, the growth of malware has been increasing yearly and its attacks attack on the network systems has also been on a constant rise, the number of people affected by these malware has also been increasing as well. Mainly in the case of online social network, the huge number of users and the large amount of information on this network make the intruders and attackers to concentrate on social media. A lot of personal and sensitive information (credit card details, user personal information like contact details) is also stored in the online social network, which helps the attacker to steal the information.

2.7 Project Objective:

The main scheme of the project is to implement a system to protect the user from the new or unknown malware and system security.
2.8 Project functionalities:

Malware can be detected with the present different kind of malware detection systems, but the system is always having threat to the new evolving malwares. There exist different malware detection systems for different operating systems like Android, Windows, and IOS. Each system has its own functionalities in detecting the malware. This prototype system uses the online social network (Twitter) for detecting the malware using the victim’s data. This web application of the prototype system is developed using the following tools.

- Eclipse with J2EE (Java 2 Platform, Enterprise Edition)
- Tomcat 6. X (minimum).
- XAMPP server
- Java SE 7 Development Kit (JDK)
3. Proposed System

The proposed system is divided into four stages starting from; 1. Data Extraction and Parsing 2. Data Filtering 3. Leveling the Data 4. Generating a Malware Alert. Where the process is done stage by stage and a malware alert is triggered at the last stage.

3.1 System Architecture:

The proposed architecture is divided into three layers (as shown in Figure 3.1)- Client layer, model layer and database layer. In client layer, a web application is designed and a request is sent to the controller, which then access the model and collects the data from the database. The model then gets the scope and sends the result to the controller. Next, the controller dispatches the result to the view, which in turns sends the results to the client.
In general view, the web application architecture is divided into four layers i.e. Presentation Layer, Business Rules Layer, Data Access Layer, Database (as seen in figure 3.2 web application architecture).
3.2 Use Case Diagrams

In this use case diagram, user is an actor who performs the actions like Registering, Login, Selecting the keywords, Entering the date and time, Selecting the social network for the case study and many more. Next, the data are extracted, filtered, smoothed and the final alarm is triggered. Figure 3.3 shows the Use Case diagram.
3.3 Sequence diagram for user authentication

The sequence diagram shows the system logic in a diagrammatic view which is used for the design process. Figure 3.4 shows the user authentication using the sequence diagram. The diagram checks if the user is authenticated to the database or not, or else, user have to register to log in.
Figure 3.4 Sequence diagram for user authentication

Another sequence diagram showing the system flow of user authentication and data extraction is shown below. Figure 3.4.1 shows the sequence diagram for user authentication and data extraction.
3.4 Twitter/Topsy:

Topsy is a social analytic tool for twitter designed to help the users or organizations check the tweets posted on twitter. The Topsy tool for twitter is a free tool and a powerful one for searching billions of tweets on tweeter. Topsy also helps the marketers and organizations check for the brand and how well their content is shared on social media like Twitter. Twitter also has the advance and a default tool search for tweets, but it is not more efficient than the Topsy search. The main advantage of Topsy is that it has most search operators and a wide variety of options such as searching for a specific person or his/her conversation.

Figure 3.4.1 sequence diagram for user authentication and data extraction.
Topsy functionalities:

- Provides all pieces of information when it is provided with specific keys to generate.
- Provides all pieces of information of a specific person on a specific topic.
- Provides all tweets and information about a user and the links used in his tweets or sites.
- Provides clear and detailed analytics of the keywords used.
- Provides the graphical way of representation of the analytics for a specific time period given.

3.5 Advantages of Topsy’s API:

The proposed social approach uses the Topsy’s API [12] for retrieving the tweets instead of using the Twitter’s API. The Topsy is one of the real time search engine, which preserves the online posts or tweets and removes the noise from the data. The important point of taking Topsy beside Twitter is, twitter’s API preserves the data not more than a week where as Topsy’s API gives data from mid 2008 where it allows for searching and collecting data for a specific time period which is helpful for malware detection. Figure 3.5 shows how Topsy works. Topsy has the flexibility for searching the tweets for the past one hour for a specific range of time and the search flexibility includes the influence search, photo search, video search, tweet search, and links. Language selection is one of the options in Topsy tool, which make it powerful when compared to other search tool like twitter search.
In the proposed approach, a few of the consideration must be done by the user, like giving a set of keywords for Topsy’s API for data collection and parsing in the first stage. In the third stage, monitoring is important when it comes to checking the threshold value as if the number of tweets is increasing or decreasing. If the number of tweets exceeds the threshold value, an event is triggered which then generates the malware alert.

3.6 Flow of Execution:

The flow of execution of detecting malware is shown in the figure 3.6. The total flow of the malware alert system is explained in the figure below.
3.7 Work flow of the system

Figure 3.7 shows the workflow of the proposed system. Here, the query engine gives the set of keywords to the Twitter/Topsy API. The Twitter/Topsy API searches the tweets according to the keyword provided. The results obtained from the API are send to the sand for the next process of data extraction.
3.8 Environment

3.8.1 Eclipse Luna

Eclipse is an Integrated Development Environment (IDE), written in any programming language, mostly in Java. The Eclipse consists of a wide variety of plugins and a workspace to load or save the files. It is used for developing the applications like mobile applications for android platforms, providing sufficient tools for development and web applications providing required plug-ins and libraries.

3.8.2 Tomcat 6. X (minimum)

Tomcat is an open source web server used to implement the Java EE and Java server pages.

3.8.3 XAMPP server

XAMPP is a free, open source for a web server solution. XAMPP consist of the MySQL database, Apache HTTP, ProFTPD. Figure 3.8 shows the general XAMPP application for MAC OS X.
Figure 3.8 showing the XAMPP 1.8.3-4 Application

Figure 3.9 shows the servers in the XAMPP. This window is used to manage the servers to start, stop and restart. MySQL Database, Apache Web server can be managed through the XAMPP server.
Figure 3.9 shows the Manage Servers window on XAMPP
4. FUNCTIONALITY OF THE APPLICATION

The outcome of the project is to implement a web application for malware alert system in an effective way by using social network to detect the new malware in the network and alerts about the new malware to the users. There are different modules; each module consists of techniques and process for simulating the data. The malware alert system is implemented by creating an application – Malware alert system, which uses Twitter as a case study. The Malware alert system application consists of user registration and login page, selecting the keywords, entering the attributes required, extracting the tweets or data from Topsy, data filtering, data smoothing, and triggering an alert.

4.1 Malware Alert system web application

In the Malware alert system web application, users log in to the profile they have registered before. The Topsy API (application program interface) provides the tweets from the twitter.

User Interface

The proposed system is designed as a web application shown as a website, which uses the JavaServer Pages (JSP).

4.2 Malware alert system

The malware alert system is a web application, which detects the malware and triggers an alert. Some of the user interfaces are:
4.2.1 Home

The home page (as shown in figure 4.1) consists of three buttons “HOME”, “REGISTER”, “LOGIN”. The user needs to register or login to continue with the application.

![Figure 4.1 Home page for Malware Alert System](image)

4.2.2 Register page

A new user must register for log in to the system. The Register page asks the new user to provide the username details and password details. The registration page has two fields for password entry to double check the given password. Once the registration process is done, the new user gets the username and password as shown in figure 4.2.
4.2.3 Login

The login page consists of two fields: username, password. The user has to enter the username and password to login. Figure 4.3 shows the login page.
4.2.4 Welcome page:

After a successful log in, the user navigates into a welcome page. The user selects kind of social network for example Twitter as a case study. One of the Keyword is selected from the options “Malware”, Cyber Attack”, “Back Door”. The user selects a
specific range of date and time from date time format “YYYYMMDDHHMMSS” to date time “YYYYMMDDHHMMSS”. The user submits the form to extract the tweets from Topsy source. Figure 4.4 shows the welcome page. For example date and time is taken as from date 20150402000000 to 20150403000000.

Figure 4.4 Welcome Page
4.2.5 Data Extraction

Once the user submits the form with the required fields, the page redirects to Data Extraction page. Here, the data is extracted from the selected fields with the keyword mentioned and description of the malware for a specific time period mentioned in the form. Figure 4.5 shows the data extraction of tweets for the keyword “malware”.

In this stage, first step is to build a web application that takes out the results from Topsy’s API [4] by using specific key terms in JSON format [2]. For example key words such as “malware”, “virus”, “cyber attack” gave stable results beside the larger set of sentences or keywords. The Topsy’s API is preferred beside the Twitter API because of its specific scope towards the social networks in general. The Topsy’s influence algorithm detects the noises, irregular, spam, data or tweets and ignores them where as Twitter API have no such functionalities like the Topsy’s API and doesn’t have data which are older than a week [13].

```java
public class ExtractService {
    private String apiKey = "09C43A9B270A470B8EB8F2946A9369F3";

    public List<Post> extractTweets(ExtractTO extractTO) {
        List<Post> resultsList = fetchTweets(extractTO);
        return resultsList;
    }
}
```

Extractservice.java class is implemented for extracting the tweets from the Topsy API using the API key (09C43A98270A470B8EB8F2946A9369F3).
4.2.6 Data Filtering

Once the data extraction is done, the next phase is data filtering. The data filtering is an method for getting the required data from large amounts of data. Figure 4.7 shows the data after the filtration process, which is made according to the pre-defined parameters (Computer security, new, discover, hit, infect warn, watch out).
Data Filtering is the second stage where data is filtered for unnecessary tweets such as those which include the keywords used in the first stage but not actually related to the malware detection. For example, a tweet like “How harmful a malware is?” In this example, the keyword “malware” is included, but the tweet is not actually related to the malware alert system. To overcome this problem, an effective refining algorithm is used to filter the tweets that are unnecessary for the system. To do this, the idea is to add extra phrases like: warn, infect, security, new, attack, alert, Danger [14]. Figure 4.6 shows the code snippet for data filtration.

```
package com.services;

import java.util.ArrayList;

public class FilterService {
    private String[] filters = {"computer security", "new", "discover", "hit", "infect", "warn", "watch out"};

    private ArrayList<String> FilterKeywords = new ArrayList<String>();

    public void setFilterKeywords() {
        for (int i = 0; i < filters.length; i++) {
            this.filterKeywords.add(filters[i]);
        }
    }
}
```

Figure 4.6 Code snippet explaining the filters used for filtration
4.2.7 Data Smoothing:

Data Smoothing is the next step of data evaluation. The results from the data filtration are processed by two algorithms Exponential Weighing Moving Average (EWMA) and (EWMV) with a threshold value and the number of tweets. A threshold value must be maintained in the system to compare the flow of increasing and decreasing trends of the tweets. Figure 4.8 shows the final result of keyword malware with time intervals, detected date and time.
In this stage, events are triggered based on a number of tweets. For example, if the number of tweets increases on a given keyword, as a specific event is triggered that made the twitter users to tweet about the event. An easy method for leveling the data is to check the number of tweets at regular intervals of a given time. Comparing or observing the given number of tweets from the smoothing stage to the number of tweets from the data filtering stage within a given specific range of time with a difference in value then the event is triggered.

In this stage, Exponentially Weighted Moving Average (EWMA) and Exponentially Weighted Moving Variance (EWMV) algorithms are used to filter or level the number of tweets after the filtering stage and to monitor the tweets for any variance in increase or decrease in the number of tweets. Here, Exponentially Weighted Moving Average (EWMA) is used to filter the tweets from the previous stage i.e. Data filtering, Exponentially Weighted Moving Variance (EWMV) algorithm is used to check the tweets for any increase or decrease in the number of tweets of a given keyword, and the time intervals of 6 hours. Here X [k] is taken as the number of tweets for a time interval of k, and the EWMA value Y [k]. [21]

\[ Y[k] = \lambda \cdot X[k] + (1 - \lambda) \cdot Y[k - 1], \forall k > 1, \]

Where the smoothing factor Lamda value belongs to the set from 0 to 1. The best value is 0.2 according to some trails done and there is no specific value for the Lamda Value.

Here, the differential D [k] and the variance are calculated for threshold value T [k], whether the number of tweets increased or decreased. [21]

\[ D[k] = X[k] - Y[k - 1] \]
Exponentially Weighted Moving Variance (EWMV) is computed as [21]

\[(V[k])^2 = \gamma \cdot (D[k])^2 + (1 - \gamma) \cdot (V[k - 1])^2, \forall k > 1,\]

Here, the smoothing factor constant (gamma) is also a set from 0 to 1 and the value taken is 0.2

The final phase is to compute the threshold value for an increase number of tweets and decrease in the number of tweets. [21]

\[T[k] = Y[k - 1] + \delta \cdot V[k - 1],\]

Here, the sigma value is taken as 3. (There is no best way to ascertain the values of lamda, gamma, and sigma, assuming the values with trail and error method). From the above input fields from the home page the threshold value is 130 and the number of tweets is 150 for one of the time interval. An alert is triggered for this particular time interval, as number of tweets is greater than the threshold value. Table 3.1 shows the list of keywords used in the data extraction and data filtering steps. These are some of the keywords and phrases used, but can add any number of keywords and phrases in data extraction and data filtration.

**Table 3.1 Keywords used for data extraction and data filtration**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Data Extraction</th>
<th>Data Filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Malware</td>
<td>Computer security, new, discover, hit, infect, warn, watch out</td>
</tr>
<tr>
<td>2.</td>
<td>Backdoor</td>
<td>Computer security, new,</td>
</tr>
</tbody>
</table>
3. Cyber Attack

<table>
<thead>
<tr>
<th>3.</th>
<th>Cyber Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Computer security, new, discover, hit, infect, warn, watch out</td>
</tr>
</tbody>
</table>

Figure 4.8 Data Smoothing
4.2.8 Generating Alert

Data smoothing is the last step in obtaining the required data. With the threshold value and number tweets, if the number of tweets increases drastically more than the given threshold value, then system triggers an alert.

Figure 4.9 shows the curves of EWMA, Number of tweets and Threshold values with the keywords as taken in the data extraction step i.e. Malware, cyber attack, backdoor.

Figure 4.10 shows the list of tweets at the particular time interval calculated by threshold value.

Figure 4.9 Number of tweets, EWMA, Threshold curves using three keywords [22]
4.2.9 History

The history page shows the results to the user when a malware is detected; it also shows the time the malware is detected with the keyword. Figure 4.11 shows the history of the event occurred.
4.2.10 Saving Results in Database

Figure 4.12 shows the list of tweets at time interval of a specific user in the database “malarealert”.
Figure 4.12 List of Tweets at Time Interval

<table>
<thead>
<tr>
<th>username</th>
<th>keyword</th>
<th>description</th>
<th>posteddate</th>
<th>detecteddate</th>
</tr>
</thead>
<tbody>
<tr>
<td>sai</td>
<td>malware</td>
<td>[BLOB - 12 B]</td>
<td>1428910083</td>
<td>1428910083</td>
</tr>
<tr>
<td>sai</td>
<td>malware</td>
<td>[BLOB - 11 B]</td>
<td>1428793877</td>
<td>20150414211834</td>
</tr>
<tr>
<td>sai</td>
<td>malware</td>
<td>[BLOB - 39 B]</td>
<td>1428795159</td>
<td>20150414211834</td>
</tr>
<tr>
<td>sai</td>
<td>malware</td>
<td>[BLOB - 07 B]</td>
<td>1428787147</td>
<td>20150414211834</td>
</tr>
<tr>
<td>sai</td>
<td>malware</td>
<td>[BLOB - 40 B]</td>
<td>1428783796</td>
<td>20150414211834</td>
</tr>
<tr>
<td>sai</td>
<td>malware</td>
<td>[BLOB - 76 B]</td>
<td>1428785564</td>
<td>20150414211834</td>
</tr>
</tbody>
</table>
5. TESTING AND EVALUATION

Testing is a process to make sure the system is working efficiently. To ensure that the system is working effectively, testing must be done on the entire cycle of the system besides testing only at a particular address of the system.

The Testing and evaluation as explained before is a process where the system is compared to the specifications and requirements throughout the testing.

The proposed system is an implementation of an approach for providing a malware alert system using social network i.e. Twitter. Using Topsy’s API for data collection. This project can be evaluated by examining few test cases, according to the events:

5.1 Test Cases

The first level of testing the proposed system is by making the intermediate level first. Giving improper inputs and keywords that were not related to the system. The following are the some of the test cases to check the proposed project.

Test case 1:

Topsy Social Search: Using Multiple Keywords

In the Data extraction step, the user must select a keyword for data extraction from the Topsy Search. Figure 5.1 shows the results from the Topsy for multiple keywords. Keyword selected plays important role in extracting the tweets from the Topsy’s API.
**Figure 5.1 Extracting tweets using “Malware” keyword**

**Tweets From the Multiple keywords: Figure** 5.2 shows the list of tweets for the given multiple keywords.
Test case 2:

Using keyword “Cyber Attack”: Extraction

In the Data extraction step, the user must select a keyword for data extraction. Figure 5.3 shows the results for data extraction for the keyword “Cyber Attack”. Keyword selected plays important role in extracting the tweets from the Topsy’s API.
**Filtration Step**

Here, data is filtered based on predefined parameters. Figure 5.4 shows the tweets after the data extraction process.
Figure 5.4 Data Filtration

Smoothing Step:

Smoothing process is done using the number of tweets and threshold value. Figure 5.5 shows the Smoothing step.
DATA IS SMOOTHED BY APPLYING EXPONENTIAL WEIGHTED MOVING AVERAGE AND EXPONENTIAL WEIGHTED MOVING VARIANCE.
WHEN THE THRESHOLD VALUE OF VARIANCE IS EXCEEDED BY ACTUAL NUMBER OF TWEETS ALERT WILL BE DISPLAYED FOR RESPECTIVE INTERVAL.

<table>
<thead>
<tr>
<th>Id</th>
<th>Keyword</th>
<th>Description</th>
<th>From (Time Interval)</th>
<th>To (Time Interval)</th>
<th>Detected Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>cyber</td>
<td>attack/bound voluminously in time interval from =20150401133000 to 20150401133000</td>
<td>20150401133000</td>
<td>20150401133000</td>
<td>20150427005108</td>
</tr>
<tr>
<td>2</td>
<td>cyber</td>
<td>attack/bound voluminously in time interval from =2015042013000 to 2015042013000</td>
<td>2015042013000</td>
<td>2015042013000</td>
<td>20150427005108</td>
</tr>
</tbody>
</table>

Figure 5.5 Data smoothing

Triggering Alert:

Final tweets can be seen in “view tweets” after the data smoothing. Figure 5.6 shows the data smoothing.
Test case 3:

Using keyword “Back Door”: Extraction

In the Data extraction step, the user must select a keyword for data extraction. Figure 5.7 shows the results for data extraction for keyword “Back Door”. Keyword selected plays important role in extracting the tweets from the Topsy’s API.
Filtration Step

Here data is filtered based on predefined parameters. Figure 5.8 shows the tweets after the data filtration process.
DATA IS FILTERED BASED ON PREDEFINED PARAMETERS!!

...Figure 5.8 Data Filtration

**Smoothing Step:**

Smoothing process is done using the number of tweets and threshold value. Figure 5.9 shows the Smoothing step.
DATA IS SMOOTHED BY APPLYING EXPONENTIAL WEIGHTED MOVING AVERAGE AND EXPONENTIAL WEIGHTED MOVING VARIANCE.
WHEN THE THRESHOLD VALUE OF VARIANCE IS EXCEEDED BY ACTUAL NUMBER OF TWEETS ALERT WILL BE DISPLAYED FOR RESPECTIVE INTERVAL.

NO ALERTS FOUND IN THE SPECIFIED TIME INTERVALS WITH CORRESPONDING KEYWORDS

Figure 5.9 Data smoothing

Triggering Alert:

No alerts found in the specified time intervals with the key word “back door”.
6. CONCLUSIONS AND FUTURE WORK

The present antivirus software can detect and delete the malicious content and sometimes prevent the threats attacking the system, but cannot prevent or detect the new threats or malware on the network.

Online social networks like Facebook, Twitter, MySpace, and YouTube popularity is increasing rapidly from last the 5 years. With increasing number of users, the users affected by malware due to the online social network are also increasing.

On average, a user spends 3-4 hours daily on any popular Online Social networks. Researchers, business people and some experts are attracted to this huge popularity of the online social network to implement creative applications using this network of huge data in one place.

This popularity of online social networking also attracted us to implement a malware alert system to detect the new malware using any one popular online social network using Twitter as a case study.

This malware alert system is a prototype system to warn people all around the world using the data from the victims. This real time malware alert system which collects data from various victims of malware attack and Trojans will generate a malware alert system by early alarming about the victims.

From the testing cases, the system is tested with different keywords to reduce false alarms. From the test case of Topsy social wed search, it shows no results found alert with multiple keywords in the search box. This shows that Topsy works better with one input keyword at a time.
From the test case of keyword cyber attack there are two alerts generated at two different intervals of time and view tweets can be seen when the alerts are triggered. From the test case of keyword backdoor, there exists no malware alert for any time interval. Thus the result is shown as “no alerts found in the specified time interval with correspondence keywords”.

The proposed simple system to detect the malware attacks using social network have several things to be addressed. The system just alerts the malware that is spreading on the network. The system does not detect if two malware or more are spreading at the same time. The system extracts the tweets only with less number of keywords. Adding more keywords to the system can enhance the system.

It is also important to tell what exactly the malware is attacking on which operating system. Enhancing the system with different keywords rather than using the same set of keywords.
BIBLIOGRAPHY AND REFERENCES


Appendix 1

The following code implements the Exponential Weighing Moving Average (EWMA) and Exponential Weighing Moving Variance (EWMV)

SmoothingService.java

```java
package com.services;

import java.util.ArrayList;

public class SmoothingService {

    private int kInterval = 21600; // 6 hours interval
    private List<Long> timesList = new ArrayList<Long>();
    private List<Post> filteredList = new ArrayList<Post>();
    List<Integer> noOfTweets = new ArrayList<Integer>();
    private double lambda = 0.2;
    private double gamma = 0.2;
    private double delta = 0.5;
    private List<Double> differentialsList = new ArrayList<Double>();
    private List<Double> thresholdList = new ArrayList<Double>();
    private List<Double> ewmList = new ArrayList<Double>();
    private List<Long> alertedTimeInterval = new ArrayList<Long>();
    private List<Post> allPostsAtDiffIntervals = new ArrayList<Post>();

    private List<Long> getTimeIntervals() {
        long fromTimeTemp = ExtractService.fromTime;
        long toTimeTemp = ExtractService.toTime;
        int timeSlice = kInterval;
        System.out.println("times=" + fromTimeTemp + "to" + toTimeTemp);
        List<Long> timesList = new ArrayList<Long>();
        if ((toTimeTemp - fromTimeTemp) > timeSlice) {
            while ((toTimeTemp - fromTimeTemp) > timeSlice) {
                timesList.add(fromTimeTemp);
                fromTimeTemp = fromTimeTemp + timeSlice;
            }
        } else {
            timesList.add(fromTimeTemp);
            timesList.add(toTimeTemp);
        }
        return timesList;
    }

    public List<ResultsData> dataSmoothing(List<Post> filteredList) {
        this.filteredList = filteredList;
        this.timesList = getTimeIntervals();
        getNoTweetsIntervals();
        populateEWMAList();
        populateEWMMVarianceList();
        for (int i = 1; i < ewmList.size(); i++) {
            thresholdList.add(getThreshold(i));
        }
    }
```
List<Integer> alertsAt = checkAlert();
List<ResultsData> resultsDataList = new ArrayList<ResultsData>();
for (int i = 0; i < alertsAt.size(); i++) {
    int index = alertsAt.get(i);
    ResultsData resultsData = new ResultsData();
    resultsData.setFromTime(timesList.get(index).toString());
    resultsData.setToTime(timesList.get(index+1).toString());
    String description = "found voluminously in time interval from "
    + CustomUtils.unixTimeToDate(Long.parseLong(resultsData.getFromTime()))
    + " to " + CustomUtils.unixTimeToDate(Long.parseLong(resultsData.getToTime()));
    resultsData.setDescription(description);
    Date date = new Date();
    resultsData.setDetectedDate(CustomUtils.dateToString(date));
    resultsData.setTweetsList(allPostsAtDiffIntervals.get(index));
    resultsDataList.add(resultsData);
}
return resultsDataList;

public List<Long> getIntervalsWhereAlerted() {
    return alertedTimeIntervals;
}

private List<Post> getMalwarePosts(List<Integer> alertsAt) {
    List<Post> listOfPosts = new ArrayList<Post>();
    for (int i = 0; i < alertsAt.size(); i++) {
        List<Post> tempList = getPostsAtTimeInterval(alertsAt.get(i));
        alertedTimeIntervals.add(timesList.get(i));
        listOfPosts.addAll(tempList);
    }
    return listOfPosts;
}

private List<Post> getPostsAtTimeInterval(int ind) {
    List<Post> listOfPosts = new ArrayList<Post>();
    long from = timesList.get(ind);
    long to = timesList.get(ind + 1);
    for (Iterator iterator = filteredList.iterator(); iterator.hasNext();)
        Post post = (Post) iterator.next();
        long firstPostDate = Long.parseLong(post.getFirstPostDate());
        if (firstPostDate >= from && firstPostDate <= to) {
            listOfPosts.add(post);
        }
}
private List<Integer> checkAlert() {
    List<Integer> alertsAtIntervals = new ArrayList<Integer>();
    for (int i = 0; i < noOfTweets.size(); i++) {
        int tweets = noOfTweets.get(i);
        System.out.println("no of tweets = " + tweets);
        System.out.println("threshold = " + thresholdList.get(i));
        if (tweets > thresholdList.get(i)) {
            // alert here
            System.out.println("alert at interval= " + i);
            alertsAtIntervals.add(i);
        }
    }
    return alertsAtIntervals;
}

private void populateEWMAList() {
    for (int i = 0; i < noOfTweets.size(); i++) {
        ewmaList.add(getEWMA(i));
    }
}

private void getNoOfTweetsIntervalK() {
    noOfTweets = new ArrayList<Integer>();
    for (int i = 0; i < getTimeIntervals(); i++) {
        int count = 0;
        List<Post> tempList = new ArrayList<Post>();
        Iterator iterator = filteredList.iterator();
        while (iterator.hasNext()) {
            Post post2 = (Post) iterator.next();
            long firstPostDate = Long.parseLong(post2.getFirstPost_date());
            if (firstPostDate >= timesList.get(i)
                && firstPostDate < timesList.get(i + 1)) {
                count++;
                tempList.add(post2);
            }
        }
        noOfTweets.add(count);
        this.allPostsAtDiffIntervals.add(tempList);
    }
}

private int getTotalTimeIntervals() {
    return timesList.size() - 1;
}
private double getEWMA(int k) {
    if (k < 0)
        return 0;
    else if (k == 0)
        return lambda * noOfTweets.get(k) + (1 - lambda) * getEWMA(k - 1);
    else {
        return lambda * noOfTweets.get(k) + (1 - lambda) * getEWMA(k - 1);
    }
}

private double getDifferentials(int k) {
    if (k < 0)
        return 0;
    else if (k == 0)
        return noOfTweets.get(k) - getEWMA(k - 1);
    else {
        double valDiff = noOfTweets.get(k) - getEWMA(k - 1);
        return valDiff;
    }
}

private double computeEWMV(int k) {
    if (k < 0)
        return 0;
    else if (k == 0)
        return Math.sqrt(gamma * (getDifferentials(k))
                        * (getDifferentials(k)));
    else {
        return Math
                        .sqrt((gamma * (getDifferentials(k)) * (getDifferentials(k))
                                + ((1 - gamma) * (computeEWMV(k - 1) * computeEWMV(k - 1)))));
    }
}

private void populateEWMVVariancesList() {
    for (int i = 0; i < noOfTweets.size(); i++) {
        ewmVList.add(computeEWMV(i));
    }
}

private double getThresholds(int k) {
    return getEWMA(k - 1) + (delta * computeEWMV(k - 1));
}
Class Diagram
Class Diagram for Database Connection: