www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 6 Issue 2 Feb. 2017, Page No. 20240-20247
Index Copernicus Value (2015): 58.10, DOI: 10.18535/ijecs/v6i2.14

Detecting Spammers on Social Networks

Mrs. A.Jesila Banu, Mr. N.Nasurudeen Ahamed, Mr.B.Manivannan, Mrs.K.Vanitha, Dr.M.Mohamed Musthafa
II M.E., (CSE),
M.E., Assistant Professor,
M.E., Associate Professor,
M.E., (Ph.D.,) Assistant Professor
M.Tech., Ph.D., Associate Professor
Al-Ameen Engineering College, Erode, Tamilnadu, India.
jesilapeer@gmail.com

ABSTRACT

Third-party apps are a major reason for the popularity and addictiveness of Facebook. Unfortunately, hackers have realized the potential of using apps for spreading malware and spam. The problem is already significant, as system find that at least 13% of apps in our dataset are malicious. So far, the research community has focused on detecting malicious posts and campaigns.In this paper, system ask the question: Given a Facebook application, can system determine if it is malicious? Our key contribution is in developing FRAppE—Facebook's Rigorous Application Evaluator—arguably the first tool focused on detecting malicious apps on Facebook. To develop FRAppE, system use information gathered by observing the posting behavior of 111K Facebook apps seen across 2.2 million users on Facebook. First, system identify a set of features that help us distinguish malicious apps from benign ones. For example, system find that malicious apps often share names with other apps, and they typically request fewer permissions than benign apps. Second, leveraging these distinguishing features, system show that FRAppE can detect malicious apps with 99.5% accuracy, with no false positives and a high true positive rate (95.9%). Finally, system explore the ecosystem of malicious Facebook apps and identify mechanisms that these apps use to propagate. Interestingly, system find that many apps collude and support each other; in our dataset, system find 1584 apps enabling the viral propagation of 3723 other apps through their posts. Long term, system see FRAppE as a step toward creating an independent watchdog for app assessment and ranking, so as to warn Facebook users before installing apps.

INTRODUCTION

1.SOCIAL NETWORK CONCEPT

Wikipedia defines a social network service as a service which "focuses on the building and verifying of online social networks for communities of people who share interests and activities, or who are interested in exploring the interests and activities of others, and which necessitates the use of software."

A report published by OCLC provides the following definition of social networking sites: "Web sites primarily designed to facilitate interaction between users who share interests, attitudes and activities, such as Facebook, Mixi and MySpace."

2. Characteristics of Networking:

The following characteristics should be considered in network design and ongoing maintenance:

1. Availability is typically measured in a percentage based on the number of minutes that exist in a year. Therefore, uptime would be the number of minutes the network is available divided by the number of minutes in a year.
2. Cost includes the cost of the network components, their installation, and their ongoing maintenance.
3. Reliability defines the reliability of the network components and the connectivity between them. Mean time between failures (MTBF) is commonly used to measure reliability.
4. Security includes the protection of the network components and the data they contain and/or the data transmitted between them.
5. Speed includes how fast data is transmitted between network end points (the data rate).
6. Scalability defines how well the network can adapt to new growth, including new users, applications, and network components.
7. Topology describes the physical cabling layout and the logical way data moves between components.

SYSTEM MODEL

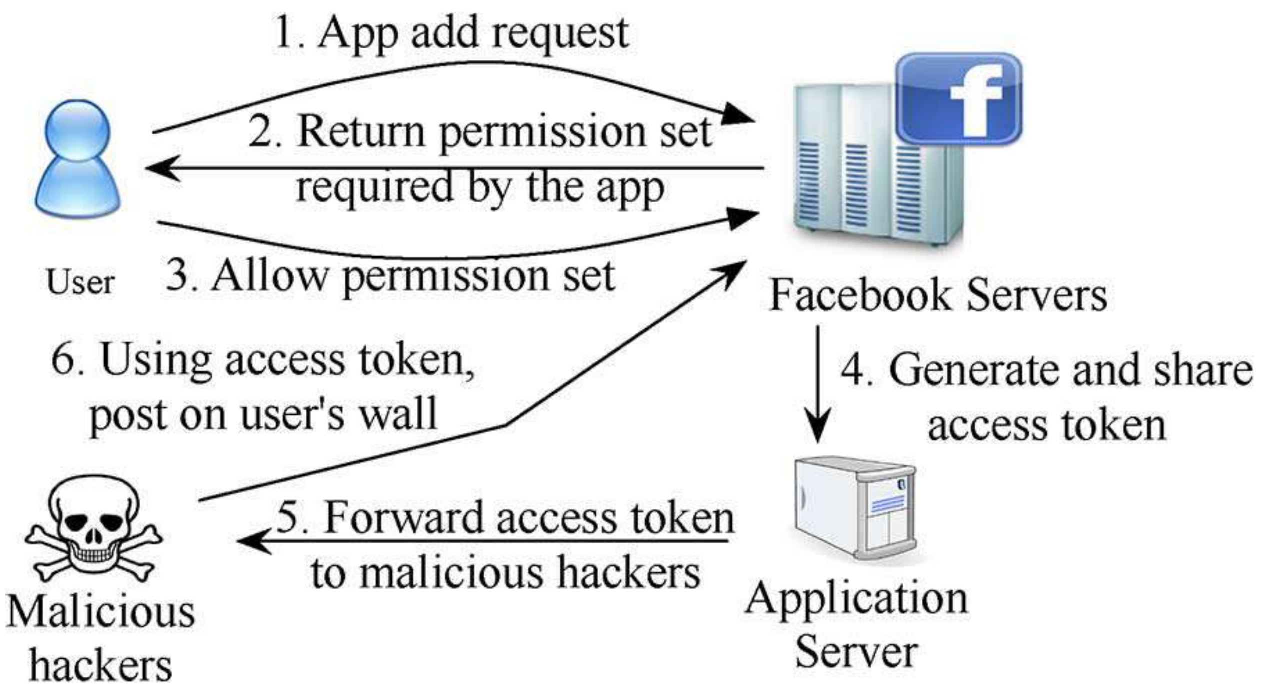
1. App add request
2. Return permission set required by the app
User
3. Allow permission set
Facebook Servers
6. Using access token, post on user's wall
4. Generate and share access token
5. Forward access token to malicious hackers
Malicious hackers
Application Server

Figure 3.1.SYSTEM MODEL

DOI: 10.18535/ijecs/v6i2.14

SYSTEM ARCHITECTURE

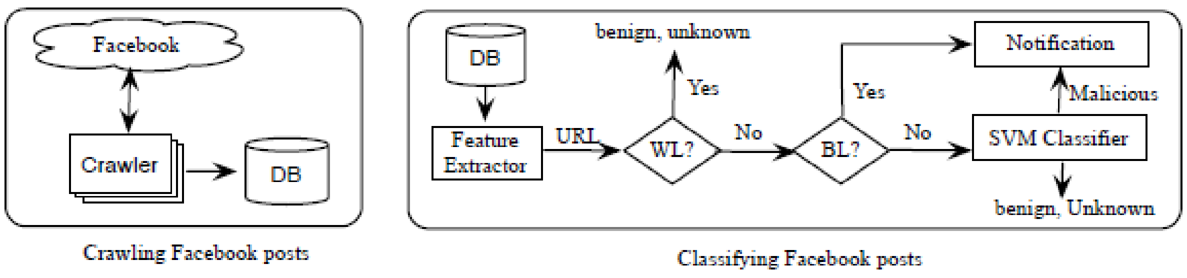

Figure 3.2.SYSTEM ARCHITECTURE

3.INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

4.OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

5.Software Environment

Java Technology

Java technology is both a programming language and a platform.

The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust

- Dynamic
- Secure

The following figure illustrates how this works.

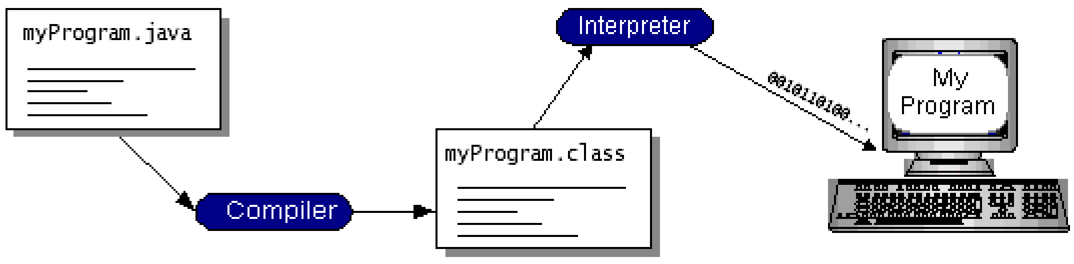

Figure 4.1. JAVA VIRTUAL MACHINE

You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

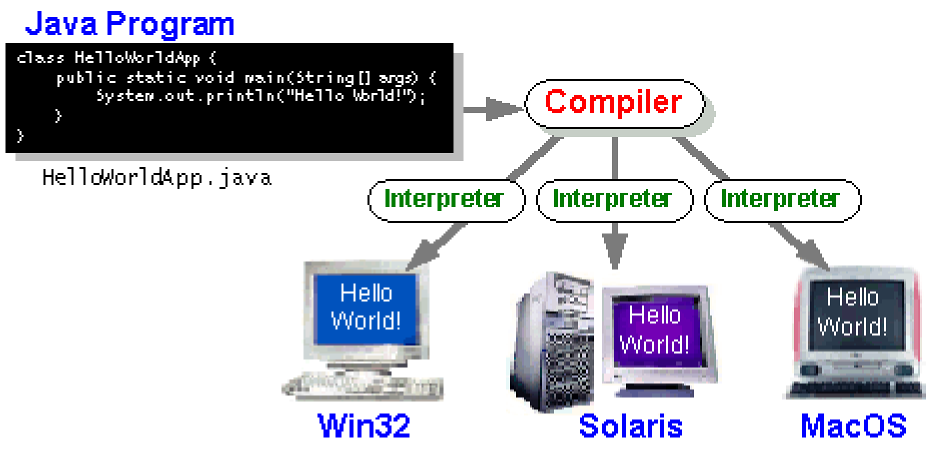
```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Figure 4.2.JAVA ENVIRONMENT

The Java Platform

A *platform* is the hardware or software environment in which a program runs. System have already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

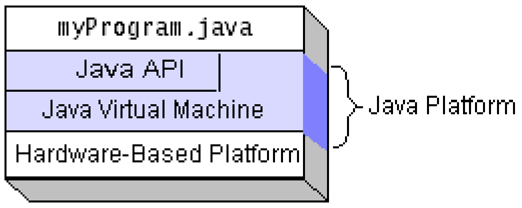

Figure 4.3. JAVA PLATFORM

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

SYSTEM IMPLEMENTATION

MODULES

- Data collection
- Feature extraction
- Training
- Classification
- Detecting Suspicious

1. Data collection

The data collection component has two subcomponents: the collection of facebook apps with URLs and crawling for URL redirections. Whenever this component obtains a facebook app with a URL, it executes a crawling thread that follows all redirections of the URL and looks up the corresponding IP addresses. The crawling thread appends these retrieved URL and IP chains to the tweet information and pushes it into a queue. As system have seen, our crawler cannot reach malicious landing URLs when they use conditional redirections to evade crawlers. However, because our detection system does not rely on the features of landing URLs, it works independently of such crawler evasions.

2. Feature extraction

The feature extraction component has three subcomponents: grouping of identical domains, finding entry point URLs, and extracting feature vectors.

To classify a post, MyPageKeeper evaluates every embedded URL in the post. Our key novelty lies in considering only the social context (e.g., the text message in the post, and the number of Likes on it) for the classification of the URL and the related post. Furthermore, system use the fact that system are observing more than one user, which can help us detect an epidemic spread.

It detects Presence of Spam keywords like ‘FREE’, ‘DEAL’ and ‘HURRY’.

3. Training

The training component has two subcomponents: retrieval of account statuses and training of the classifier. Because system use an offline supervised learning algorithm, the feature vectors for training are relatively older than feature vectors for classification. To label the training vectors, system use the account status; URLs from suspended accounts are considered malicious whereas URLs from active accounts are considered benign. System periodically update our classifier using labeled training vectors.

4. Classification

The classification component executes our classifier using input feature vectors to classify suspicious URLs. When the classifier returns a number of malicious feature vectors, this component flags the corresponding URLs information as suspicious.

The classification module uses a Machine Learning classifier based on Support Vector Machines, but also utilizes several local and external white lists and blacklists that help speed up the process and increase the over-all accuracy. The classification module receives a URL and the related social context features extracted in the previous step.

These URLs, detected as suspicious, will be delivered to security experts or more sophisticated dynamic analysis environments for an in-depth investigation.

5. Detecting Suspicious

DOI: 10.18535/ijecs/v6i2.14

The Detecting Suspicious and notification module notifies all users who have social malware posts in their wall or news feed. The user can currently specify the notification mechanism, which can be a combination of emailing the user or posting a comment on the suspect posts.