Detecting Malicious Facebook Applications

With 20 million installs a day, 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 we 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, we ask the question: Given a Facebook application, can we 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, we use information gathered by observing the posting behavior of 111K Facebook apps seen across 2.2 million users on Facebook. First, we identify a set of features that help us distinguish malicious apps from benign ones. For example, we find that malicious apps often share names with other apps, and they typically request fewer permissions than benign apps. Second, leveraging these distinguishing features, we show that FRAppE can detect malicious apps with 99.5% accuracy, with no false positives and a high true positive rate (95.9%). Finally, we explore the ecosystem of malicious Facebook apps and identify mechanisms that these apps use to propagate. Interestingly, we find that many apps collude and support each other; in our dataset, we find 1584 apps enabling the viral propagation of 3723 other apps through their posts. Long term, we see FRAppE as a step toward creating an independent watchdog for app assessment and ranking, so as to warn Facebook users before installing apps.

EXISTING SYSTEM:

1. So far, the research community has paid little attention to OSN apps specifically. Most research related to spam and malware on Facebook has focused on detecting malicious posts and social spam campaigns.
2. Gao *et al.* analyzed posts on the walls of 3.5 million Facebook users and showed that 10% of links posted on Facebook walls are spam. They also presented techniques to identify compromised accounts and spam campaigns.
3. Yang *et al.* and Benevenuto *et al.* developed techniques to identify accounts of spammers on Twitter. Others have proposed a honey-pot-based approach to detect spam accounts on OSNs.
4. Yardi *et al.* analyzed behavioral patterns among spam accounts in Twitter.
5. Chia *et al.*investigate risk signaling on the privacy intrusiveness of Facebook apps and conclude that current forms of community ratings are not reliable indicators of the privacy risks associated with an app.

DISADVANTAGES OF EXISTING SYSTEM:

1. Existing system works concentrated only on classifying individual URLs or posts as spam, but not focused on identifying malicious applications that are the main source of spam on Facebook.
2. Existing system works focused on accounts created by spammers instead of malicious application.
3. Existing system provided only a high-level overview about threats to the Facebook graph and do not provide any analysis of the system.

PROPOSED SYSTEM:

1. In this paper, we develop FRAppE, a suite of efficient classification techniques for identifying whether an app is malicious or not. To build FRAppE, we use data from MyPage- Keeper, a security app in Facebook.
2. We find that malicious applications significantly differ from benign applications with respect to two classes of features: On-Demand Features and Aggregation-Based Features.
3. We present two variants of our malicious app classifier— FRAppE Lite and FRAppE.
4. FRAppE Lite is a lightweight version that makes use of only the application features available on demand. Given a specific app ID, FRAppE Lite crawls the on-demand features for that application and evaluates the application based on these features in real time.
5. FRAppE—a malicious app detector that utilizes our aggregation-based features in addition to the on-demand features.

ADVANTAGES OF PROPOSED SYSTEM:

1. The proposed work is arguably the first comprehensive study focusing on malicious Facebook apps that focuses on quantifying, profiling, and understanding malicious apps and synthesizes this information into an effective detection approach.
2. Several features used by FRAppE, such as the reputation of redirect URIs, the number of required permissions, and the use of different client IDs in app installation URLs, are robust to the evolution of hackers.
3. Not using different client IDs in app installation URLs would limit the ability of hackers to instrument their applications to propagate each other.

MODULES:

1. Data collection
2. Feature extraction
3. Training
4. Classification & Detection

MODULES DESCRIPTION:

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 we 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.

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, we use the fact that we 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’.

Training

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

Classification & Detection

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.

SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

1. System : Pentium IV 2.4 GHz.
2. Hard Disk : 40 GB.
3. Floppy Drive : 1.44 Mb.
4. Monitor : 15 VGA Colour.
5. Mouse : Logitech.
6. Ram : 512 Mb.

SOFTWARE REQUIREMENTS:

1. Operating system : Windows XP/7.
2. Coding Language : JAVA/J2EE
3. IDE : Netbeans 7.4
4. Database : MYSQL