Monitoring Software Vulnerabilities through Social Networks Analysis

Slim Trabelsi[1], Henrik Plate[1], Amine Abida[1], M. Marouane Ben Aoun[1], Anis Zouaoui[2], Chedy Missaoui[2], Sofien Gharbi[2] and Alaeddine Ayari[2]
[1]*SAP Labs France, Mougins, France*
[2]*ESPRIT, School of Engineering, Tunis, Tunisia*

Keywords: Vulnerability, Social, Networks, Zero-day, Monitoring, Data Mining, CVE.

Abstract: Monitoring software vulnerability information requires an important financial and human effort in order to track all the scattered sources publishing the last news about software vulnerabilities, patches and exploits. We noticed that in some social networks like Twitter we can aggregate a lot of information related to software vulnerabilities in a single channel. In this paper, we analyse the Twitter feed in order to monitor most of the information related to software vulnerabilities including zero-day publications.

1 INTRODUCTION

In the software lifecycle management process, the software maintenance step consists in the modification of product after delivery to correct faults, to improve security, performance, usability etc. Security maintenance is a very sensitive and crucial operation in the software maintenance process due to the risk of exposure of the system when security issues are not addressed. Usually, the security maintenance life cycle consists of identifying security breaches and vulnerabilities, fix the security issue, provide a patch for the software and publish the information about the patch and the vulnerability addressed by this fix. Of course this lifecycle does not reflect the reality, because the vulnerability is not necessarily identified by the software developer/vendor but sometimes by external people (Hackers, developers, researchers, etc.). If the vulnerability is discovered by a third party, we talk about Zero-day vulnerability. In that case, the author of the discovery has two solutions: Contact the software vendor/developer and inform him about the issue, or reveal it through another channel and exposing then the software to a vulnerability exploit.

For a software user (a system IT administrator for example) it is mandatory to be informed about security information relevant for software she installed. Security information is a general term comprising, e.g., the disclosure of a new vulnerability, the availability of an exploit, or the provision of a patch from a software vendor. These users can learn about security information mainly through three different channels:

- Official channels: Used by software vendors to report security information related to their software, e.g., vulnerabilities and corresponding patches or work-around. Some vendors inform customers proactively, e.g., by direct email communication.
- Free or commercial 3rd party channels: Used by non-profit or commercial organizations to provide centralized access to security information affecting software from many different vendors (herewith facilitating the work of administrators, who do not need to consider each vendor's official channel). Some of these channels report vulnerabilities even before the affected software vendor provided a patch or work-around (e.g., Vupen Security). Others provide central access to information they have collected from official channels and other 3rd party channels (Security Database). And again others serve as public repositories of security information, e.g., the National Vulnerability Database (a public repository of verified vulnerabilities).
- Informal channels: Used by security professionals, hackers or hacking communities in order to share security information quickly (bug trackers, developer forums, etc.) and informally and by means of social media (blogs, Twitter or forums).

Trabelsi S., Plate H., Abida A., Ben Aoun M., Zouaoui A., Missaoui C., Gharbi S. and Ayari A..
Monitoring Software Vulnerabilities through Social Networks Analysis.
DOI: 10.5220/0005538602360242
In *Proceedings of the 12th International Conference on Security and Cryptography* (SECRYPT-2015), pages 236-242
ISBN: 978-989-758-117-5

Sometimes, information stemming from the above sources is simply repeated, sometimes original information about new vulnerabilities, exploits or work-around is provided.

In this paper we describe an approach and corresponding prototype to gather security information from a fundamentally different kind of information source: Social Media (Twitter), i.e., online tools supporting the creation and exchange of informal, non-validated information in virtual communities. As described by (Bougie, 2011) and (Tian, 2012), Social Media communities such as Stack Overflow, Twitter or developer blogs, became one important instrument of the global software development community, in particular regarding open-source soft-ware, and one can observe that it is many times used for the early discussion and disclosure of software vulnerabilities, exploits and patches.

As such, the automated search in Social Media offers the unique opportunity to gather security information earlier than using the classical information sources. 0-day vulnerabilities, in particular, will never be published through classical information sources like the NVD. Knowing that this immense body of knowledge is publicly accessible to malicious adversaries, it is important to make it also available to software users, in particular security-sensitive ones. Moreover, some Social Medias like Twitter, can be considered as an aggregator of both validated and non-validated security information, hence, avoid the burden to observe many information sources in parallel for getting up-to-date information.

Having as goal to gather security information as early as possible from as less sources as possible, the contributions of this paper are as follows:

- In section 3, the description of a clustering algorithm for social media content, grouping all information regarding the same subject matter, which is a pre-requisite for distinguishing “known” from “new” security information. Also in section 3, the description of a generic architecture and proof-of-concept implementation called SMASH (Social Media Analysis for Security on HANA).
- In section 4, the presentation of results of an empirical study that compares the availability of information published through Social Media tools (at the example of Twitter) and classical sources (at the example of NVD).

2 VULNERABILITY MANAGEMENT CONCEPTS AND PROBLEMS

In this section we introduce some definitions and vocabulary related to software vulnerability management. There are several standards and protocols that define specifications frames for the software vulnerability management and the most popular and adopted one is the Security Content Automation Protocol (SCAP) defined by the National Institute of Standards and Technology (NIST).

2.1 Security Content Automation Protocol (SCAP)

The Security Content Automation Protocol (SCAP) is a compilation of several open standards defined to categorize and list software weakness and configuration issues related to security. This standard offers scales to score those findings in order to evaluate the potential impact. These standards offer the possibility to automate vulnerability management, measurement, and policy compliance evaluation. In this paper we focus on a subset of elements defined by the SCAP that are related to our study:

- Common Vulnerabilities and Exposures (CVE): allows the structured description of software vulnerabilities.
- Common Platform Enumeration (CPE): is a standardized method of identifying classes of applications, operating systems, and hardware devices affected by CVEs.
- Common Vulnerability Scoring System (CVSS): is a vulnerability scoring system designed to provide an open and standardized method for rating IT vulnerabilities.

2.2 Software Vulnerability Management Problem Statement

Software vulnerability management is a wide topic covered by many standards and protocols especially when vulnerabilities are identified, analysed and confirmed by security organizations or software vendors. This kind of management systems becomes less efficient and organized when the vulnerability information is not yet confirmed, alike the case of zero-day vulnerabilities. The same disorganization is observed for exploit classification and management. Usually, profit-based security organization companies that sell software vulnerability information spend a lot of efforts to monitor public

and private bug-trackers (especially open source software bug trackers, developers and hacker forums, exploit databases and websites, etc.) to detect bugs that potentially can lead to 0-day vulnerability. This task requires a lot of efforts and human resources. One interesting solution to aggregate such sources can be found in the social media streams like Twitter. Some studies like (Bougie, 2011) and (Tian, 2012) validate our observation about the software developer activity on social media. In fact, they also noticed that the software development community extensively leverages twitter capabilities for conversation and information sharing related to their development activities. We exploit this fact and use the content of the information published by this community to simplify the data collection and reduce the human and financial cost of this task. To achieve this goal we focus on the software vulnerability information that we can extract from Twitter and that aggregates in a certain way the information scattered over different platforms (blogs, bug trackers, forums, etc.).

3 SOCIAL NETWORK ANALYSIS FOR SECURITY

This section outlines the architecture and main building blocks of SMASH (Social Media Analysis for Security on HANA, cf. Figure 1), a prototype aiming to proof the various concepts described in this paper. With its current functionality, SMASH is a live monitoring tool for vulnerabilities concerning a defined set of software components.

3.1 Architecture

The system is divided into two principal subsystems, related to data collection and processing respectively.

Data collection: This subsystem establishes a common data basis for the later processing, making all relevant raw data available in a local database. The Social Media tool considered by the proof-of-concept is Twitter, but the approach is equally applicable to other Social Media tools. Twitter has been chosen due to its wide-spread use by both individuals and organizations. The Twitter content to be replicated in the local database is obtained by performing a search in the Twitter live stream. The search terms are combinations of common security terminology ("vulnerability", "exploit", etc.) and software component names relevant for the user of SMASH. A list of such component names can be obtained from, for example, configuration management databases that maintain comprehensive software inventories. Other Social Media tools may offer different APIs, however, the advantage of streaming APIs is the immediate availability of new information compared to pull APIs.

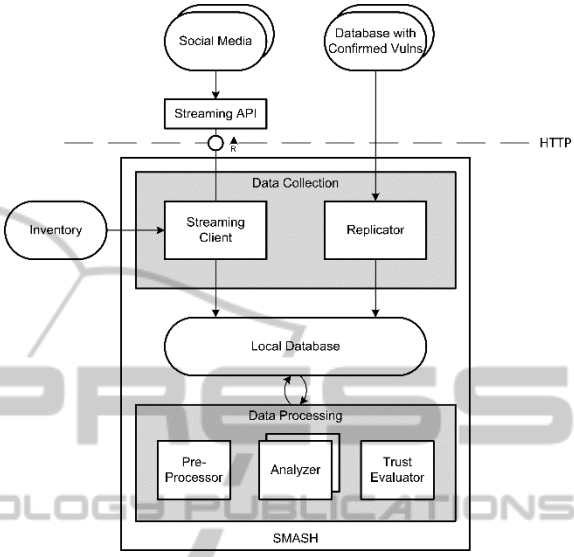

Figure 1: Generic architecture of the Software Vulnerability Monitoring System.

Besides replicated data from Twitter, the proof-of-concept maintains a local copy of a classical information source with confirmed vulnerabilities, namely the National Vulnerability Database (NVD), built using its downloadable XML data feeds. This information source is used to assess security information collected from Twitter, in particular to distinguish the publication of original (new) information from the repetition of well-known information (already accessible through the classical sources). Moreover, the NVD database comes with a comprehensive list of software names following the CPE.

Data processing: This subsystem is responsible for the identification, evaluation and classification of various kinds of security information communicated through Twitter. This is done using different data mining algorithms, each implemented by a so-called analyzer. Two such algorithms have been developed until now (cf. Sections 3.2 and 3.3 for details):

- The search for the description of 0-day vulnerabilities, i.e., vulnerabilities not yet published in classical channels like the NVD, and classification according to the existing CPE notation and database.

- The search for information about the future creation of new CVEs or update of existing ones.

Besides, the subsystem also computes the trust level (value) of Twitter users, thereby relying on a well-defined trust model and various quality criteria for Social Media content (cf. Section 4 for details).

The third component part of the data processing subsystem, executed prior to the above-mentioned components, is responsible for the pre-processing of Twitter content, e.g., the deletion of duplicates, the deletion of content not meeting certain criteria (e.g., regarding the minimum length), or the enrichment with additional information regarding the respective author or obtained from referenced websites.

3.2 Algorithm for Detecting 0-Day Vulnerabilities

In order to detect 0-day vulnerability information we identify clusters of similar messages dealing about the same issue related to the same software component, containing specific vulnerability keywords and some additional common terms. The approach used for the clustering consists of three main steps performed for each new tweet received:
1-Sanitizing and filtering tweet text by removing irrelevant content and stemming relevant terms. 2-Transforming the cleaned text into a vector. 3-Clustering similar vectors; similarity is based on a distance measure between vectors. The idea behind this approach is to make analysis of the set of data more manageable as human agents or automated analysis processes can mine relevant information from each cluster of messages; relevant information may include: vulnerability description, availability of patches and availability of exploits to name a few.

The first step is a simple removal of the terms which are considered irrelevant regarding the clustering process, tokens such as strings of non-alpha-numeric characters and strings of numbers are removed, the remaining terms which are considered relevant are stemmed to their root form.

The second step consists of building a vector representation of the text; for each term a new dimension is created then the term is weighted using TF (term frequency) weighting (Turney, 2010), this is a classical bag of words representation.

The third step is based on a straightforward clustering algorithm that is capable of clustering a stream of continuously incoming tweets. The algorithm is initialized with a threshold distance T, when the first tweet is delivered it is placed in a first cluster then for each subsequent tweet delivered the closest cluster is fetched if the distance between the tweet and the centroid of the closest cluster is lesser than the specified threshold T then the tweet is absorbed by the closest cluster if not the case a new cluster is created containing the currently processed tweet. The notion of similarity in this algorithm is expressed through a distance measure essentially if two vectors representing two documents have a relatively small distance between them this implies that the two tweets are similar.

The distance measure used in our algorithm is a modified form of the Euclidean distance which is expressed as follows:

The Distance D between two vectors A and B in an n dimensional space is:

$$D = \sqrt{\sum_{i=0}^{n}(Normed(Ai, A) - Normed(Bi, B))^2}$$

Where

$$Normed\ (v, V) = \frac{v}{Magnitude(V)}$$; where v is a vector component and V is a vector

$$Magnitude\ (V) = \sqrt{\sum_{i=0}^{n} Vi^2}$$; where V is a vector in an n dimensional space

The difference regarding the classical Euclidean measure is that each vector component is normed i.e. it is divided by the magnitude of the vector this allows to have a distance value that is comprised in the interval [0, 2]. The consequence of this propriety is that the threshold as to be in the same [0, 2] interval which is more convenient than having to choose a distance threshold in an infinite interval.

As the streaming process is progressing clusters of related tweets are forming thus similar content is grouped in real time

3.3 Algorithm for Detecting CVE Requests and Updates

The CVE based search algorithm is quite simple (see Figure 2), we execute a search for all the collected data looking for the regular expression “CVE-*-*” to obtain all the messages dealing with CVEs. Then we group messages by CVE numbers in order to obtain cluster of messages dealing with the same CVE. From this clusters we ex-tract the common keywords in order to identify the purpose of the vulnerability.

The reason why we create such clusters is to distinguish between a new CVE publication or update, from old ones that can reappear Twitter for some other reasons. Usually, when a new CVE is published or updated, several sources relay this information (not only one isolated source); this is the reason why we rely on the cluster concept.

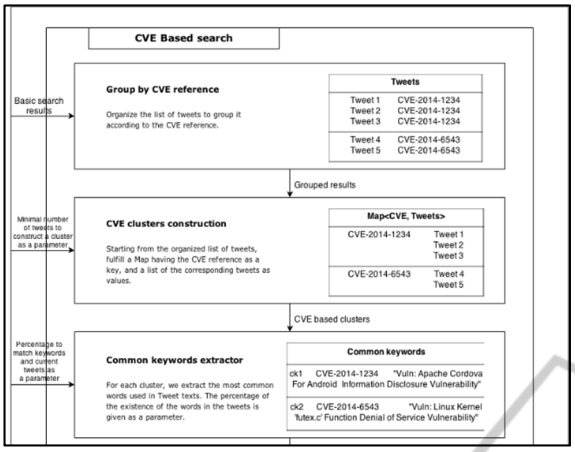

Figure 2: CVE based search algorithm.

3.4 Implementation

The implementation is done using SAP HANA, an in-memory database, SAP UI5, a modern Html5 framework for developing browser-based application frontends, and Java concerning the data collection and processing. It offers users the possibility to monitor software components of their choice, e.g., by registering a certain component name.

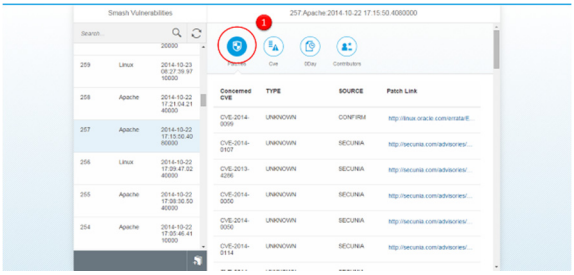

Figure 3: Screenshot of the SMASH monitoring tool.

As shown in Figure 3, the list of monitored software is displayed on the left side of the application. Upon selection, several kinds of security information identified by the tool are displayed on the right side of the screen. The two views presenting information from Twitter analysis are as follows0-day: A list of vulnerabilities collected through the two algorithms, including the identifier of the Twitter user who published the respective content. Note that this list also comprises CVE requests, i.e., vulnerabilities having a provisionary CVE identifier until their actual validation. Such requests are typically published on Twitter, including details, prior to the actual publication through the NVD.

The other two views, CVE and Patches, mainly contain information read from the NVD.

4 ZERO-DAY STUDY AND ANALYSIS

In this section, we describe two studies that we conducted related to the freshness of the data collected compared to the traditional sources. A first study concerns the comparison between the publication time/date of the new CVEs or the CVE updates of the official NIST NVD with the publications on SM. The second study concerns the 0-day early publication time/date on SM with regards to the correspondent CVE published in NIST NVD. We focus on the Linux-Kernel vulnerabilities detected in 2014.

4.1 CVE Publication Date Study

The main objective of the vulnerability monitoring study through the SM analysis is not to propose yet another monitoring system, but to propose a system that offers new information that was not yet exploited by the traditional systems like for example the freshness of the information. Being informed as early as possible about a potential thread on your system can give to the system administrator a serious advantage to protect his infrastructure. For this reason, we decided to perform a study on the time that we can gain by exploiting the SMASH results with regards to the official publication date of the official NIST NVD.

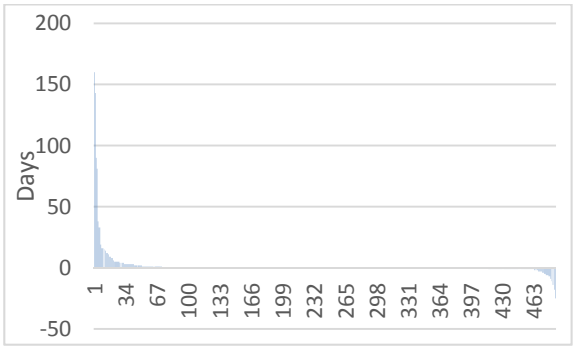

Figure 4: CVE Publication dates comparison between NIST NVD and SMASH.

Methodology: To do the study we selected the last 486 CVE published in NIST NVD before December 5th 2014. The CVE concerns any kind of software (we did not focus on a specific vendor). For every CVE-Number in the list we performed a search with SMASH in order to obtain the same CVE numbers published in Twitter. For every mapping we compared the publication dates.

Observations: The first interesting result is that 100% of the NIST CVEs were also published Twitter.

The most interesting result shown in Figure 4 is that 41% of the CVEs were published on Twitter before the official NIST NVD. The average advance time of these 41% is approximately 20 days.

We can observe in this graph that to the largest gap between the two publication dates is about 13879928 seconds ~ 160 days. If we look closer to the vulnerability, published in Twitter on 5/14/2014, we can see a link to a developer forum where the author of the vulnerability discovery talks about the details of the CVE request that he sent to the NIST NVD. 160 days later the vulnerability was confirmed and published on the NIST website on 10/21/2014. One possible for this delay is the low severity of the vulnerability (CVSS score = 2).

4.2 Zero-day Publication Date for Linux-Kernel Vulnerabilities

A zero-day vulnerability disclosure is a rare information in general. Most of the big software vendors spend a lot of efforts (and money) to keep such information secret. The exception concerns some Open Source software like Linux series. It seems that the phenomena described in (Tian, 2012) can be confirmed by our study especially for the disclosure of bugs and 0-day vulnerabilities. We decided to focus our study on the Linux-Kernel software component, for which the developer community on SM is quite important and verbose.

Methodology: To do the study we took 62 Linux-Kernel CVEs from January to July 2014 (Study made end of July 2014). Starting from the vulnerabilities descriptions we executed the SMASH search on twitter in order to detect related 0-day publications on twitter. Once we detect a matching with a publication on twitter we verify the official Linux publications with regards to the vulnerabilities and the patches in order to verify the relevance of the 0-day. If nothing appears before the twitter publications, we count it as a new 0-day detected.

Observations: 75,8% of the CVE vulnerabilities where disclosed before the official disclosure as 0-day information. Most of the tweets refer to Linux-Kernel bug trackers or Linux developer forums. The average advance time is approximately 19 days. The average CVSSS score is 5.88 and 34% of the 0-day vulnerabilities rated between 6 and 10.

The 0-day disclosure is much more critical than the early CVE disclosure, due to the fact that most of the time the software vendor is not aware of the vulnerability, and the exploitation can be easily done by malicious persons. Having, the 0-day information before the exploit publication is valuable information supporting system administrators in the protection of their systems against early vulnerability exploits.

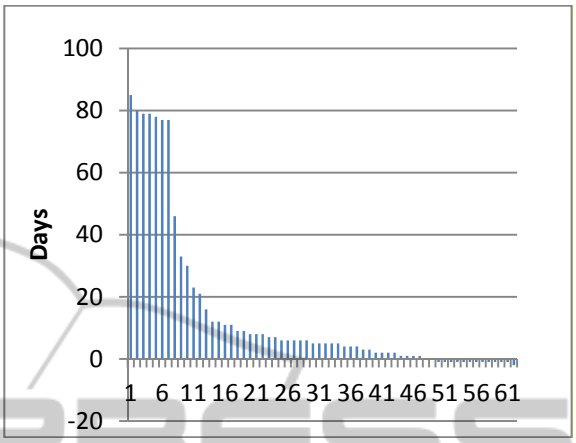

Figure 5: Zero day advance publication dates.

5 STATE OF THE ART

To our knowledge the first reference to the idea of software security monitoring through SM analysis was initiated by Arafin et al. (Arafin, 2013) that proposed the idea of searching exploits published on Twitter that are related to known CVEs. Due to the lack of experience of the study (student project report) the results were not really quantitative, they only detected the presence of exploits published on twitter, but they did not verified if these exploits were already published on Metasploit or Exploit-DB for example. Another study led by Cui et al. (Cui, 2013) focused on the tracking of users publishing cybersecurity related information, for that they proposed a trust model to verify the trustworthiness of the sources. Compared to the trust model presented in our paper, their model relies on the "influence parameter" that is to our point of view misleading dues to the uncertainty of the notion of influence where the number of followers is not a good criteria for computing the influence impact.

Most of the studies in the literature are mainly focusing on the presence presence of software engineering topics in SM. Bird et al (Bird, 2006) analysed SM based mailing lists, where developer communities exchange their work in public/private social groups. They highlighted the relationship between the level of email activity and the level of activity in the source code, and a less strong relationship with document change activity. A social network connection topography of open source

software developers in Source-Forge.net was realized by Xu et al (Xu, 2006) then by Surian et al (Surian, 2010) in order to study the interaction and the influence between software developer and code source evolution. From this study appeared the notion of experts in specific technologies. Other studies like (Xu, 2006) and (Tian Y. 2012) focused on analysing software engineering trends on Twitter. The notion of software popularity appeared in these studies.

Bug tracking monitoring on social media was also addressed in (Sureka, 2011) for open source public trackers and in (Jiang, 2013) for mobile OS Android bug reporting community. These studies focus on the bug reporting lines and management. They identify the strategies and the authority organization structure for handling bugs during the software development phase.

6 CONCLUSIONS

In this paper, we explore a new information source, namely Social Media streams, to aggregate information about new software vulnerabilities. This channel offers the possibility to track announcements coming from software vendors, NVD but also other non-structured sources publishing 0-day vulnerabilities, CVE requests, exploits etc. We obtained some interesting results especially about the impressive number of 0-day vulnerabilities related to the Linux-Kernel software published before the official NVD announcements. We claim that SM analysis can offer a cheap and easy way to efficiently monitor system security. It also offers many other possibilities to handle and monitor patching and security maintenance for complex systems that we are currently under exploration as future work. The current version of the tool relies on many manual tasks, especially for the validation of the detected information; the goal in the short term is to automate these tasks. We are also working on the validation of the trust model about the validity of the score estimation.

REFERENCES

Jiang, Feng, Jiemin Wang, Abram Hindle, and Mario A. Nascimento., 2013. *"Mining the Temporal Evolution of the Android Bug Reporting Community via Sliding Windows."* arXiv preprint arXiv:1310.7469.

Bougie, G., Starke, J., Storey, M. A., & German, D. M., 2011. *Towards understanding twitter use in software engineering: preliminary findings, ongoing challenges and future questions*. In Proceedings of the 2nd international workshop on Web 2.0 for software engineering (pp. 31-36). ACM.

Tian, Y., Achananuparp, P., Lubis, I. N., Lo, D., & Lim, E. P., 2012. *What does software engineering community microblog about?* In Mining Software Repositories (MSR), 9th IEEE Working Conference on (pp. 247-250). IEEE.

J. B. MacQueen, 1967. *"Some methods for classification and analysis of multivariate observa-tions,"* in Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability (L. M. L. Cam and J. Neyman, eds.), vol. 1, pp. 281–297, University of California Press.

Rajput, D. S., Thakur, R. S., Thakur, G. S., & Sahu, N. 2012. *"Analysis of Social net-working sites using K-mean Clustering algorithm"*. International Journal of Computer & Communication Technology (IJCCT) ISSN (ONLINE), 2231-0371.

C. Bird, A. Gourley, P. T. Devanbu, M. Gertz, and A. Swaminathan, 2006 *"Mining email social networks,"* in MSR, pp. 137–143.

D. Surian, D. Lo, and E.-P. Lim, 2010 *"Mining collaboration patterns from a large developer net-work,"* in WCRE, pp. 269–273.

Xu, Jin, Scott Christley, and Greg Madey. 2006 *"Application of social network analysis to the study of open source software."* The economics of open source software development: 205-224.

Bougie, Gargi, Jamie Starke, Margaret-Anne Storey, and Daniel M. German. 2011 *"Towards un-derstanding twitter use in software engineering: preliminary findings, ongoing challenges and future questions."* In Proceedings of the 2nd international workshop on Web 2.0 for software engineering, pp. 31-36. ACM.

Tian, Yuan, Palakorn Achananuparp, Ibrahim Nelman Lubis, David Lo, and Ee-Peng Lim. 2012 *"What does software engineering community microblog about?"* In Mining Software Re-positories (MSR), 2012 9th IEEE Working Conference on, pp. 247-250. IEEE.

Sureka, Ashish, Atul Goyal, and Ayushi Rastogi. 2011 *"Using social network analysis for mining collaboration data in a defect tracking system for risk and vulnerability analysis."* In Proceed-ings of the 4th India Software Engineering Conference, pp. 195-204. ACM.

Arafin, Md Tanvir, and Richard Royster. 2013 *"Vulnerability Exploits Advertised on Twitter."*.

Cui, B., Moskal, S., Du, H., & Yang, S. J. (2013). *Who shall we follow in twitter for cyber vulnerability?*. In Social Computing, Behavioral-Cultural Modeling and Prediction (pp. 394-402). Springer Berlin Heidelberg.

Turney, Peter D., and Patrick Pantel. "From frequency to meaning: Vector space models of semantics." Journal of artificial intelligence research 37.1 (2010): 141-188.