Mining Social Networks for Software Vulnerabilities Monitoring

Slim Trabelsi, Henrik Plate, Amine Abida, M. Marouane Ben Aoun
Product Security Research
SAP Labs France
Mougins, France
{first.lastName}@sap.com

Anis Zouaoui, Chedy Missaoui, Sofien Gharbi, Alaeddine Ayari
ESPRIT
School of Engineering
Tunis, Tunisia
{first.lastName}@esprit.tn

***Abstract*— Staying informed about security vulnerabilities, work-arounds and the availability of patches regarding the components of a given system is crucial to ensure system security. Several channels can be used to the monitor the new vulnerabilities publications, but these channels are scattered. We propose in this paper a vulnerability monitoring system based on twitter analysis that aggregates and analyses different sources of data and extracts zero-day vulnerabilities.**

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

I. INTRODUCTION

Staying informed about security vulnerabilities, work-arounds and the availability of patches regarding the components of a given system is crucial to ensure system security. However, despite huge advances in the standardization and communication of security disclosures, in particular achieved through the various elements of the Security Content Automation Protocol (SCAP), it remains being a time-consuming activity. This is due to the fact that multiple, heterogeneous information sources need to be consulted on a regular basis in order to get a comprehensive picture of vulnerabilities and corresponding mitigations for exactly those component versions that are relevant for a given user.

The classical information sources comprise security advisories issued by the individual software providers, and public and commercial vulnerability databases such as the National Vulnerability Database (NVD). Moreover, there exists a growing market of commercial offerings that support organizations in the search, collection and assessment of security information. What is common to those sources is that the information has been validated regarding its correctness prior to its disclosure, e.g., the presence of a given security vulnerability. The advantage of validation, however, comes at the cost of delayed publication, because not only the validation of information in collaboration with software vendors is time-consuming, but also because classical channels typically wait until a corresponding security patch is available.

This paper describes 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 [2] and [3], 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, work-arounds and exploits.

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 description of quality criteria for Social Media content and, based thereon, a trust model for Social Media users, all of which is meant to cope with the informal, non-validated character of Social Media content.
- In section 5, the presentation of results of an empirical study that compares the availability of information

978-1-4799-8784-9/15/$31.00 ©2015 IEEE

published through Social Media tools (at the example of Twitter) and classical sources (at the example of NVD).

II. 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[1]) defined by the National Institute of Standards and Technology (NIST[2])

A. 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.

B. 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 system 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 [2] and [3] 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.).

[1] http://scap.nist.gov/

[2] http://www.nist.gov/

III. 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.

A. Architecture

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

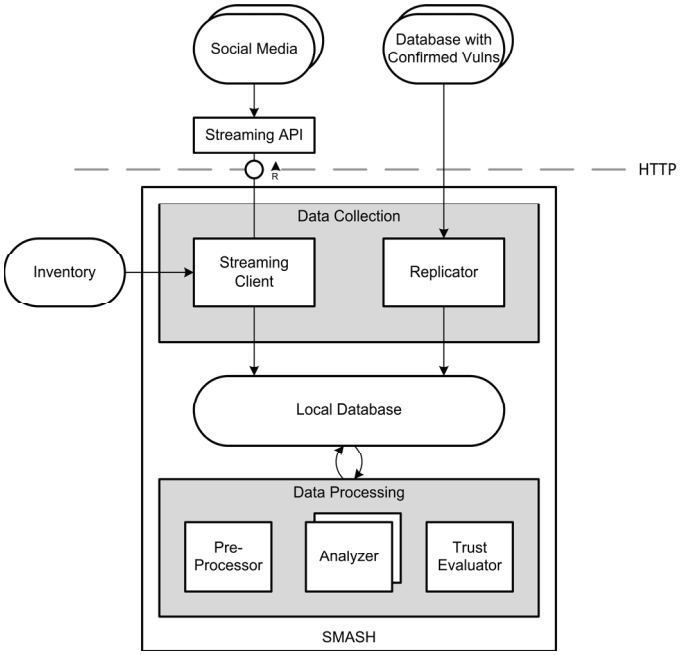

Fig. 1. Generic architecture of the Software Vulnerability Monitoring System

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.

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 III.B and III.C 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.

B. Algorithm for detecting 0-day vulnerabilities

In order to detect 0-day vulnerability information we identify clusters of similar messages dealing about the same software component, containing specific vulnerability keywords and some additional common terms. For that, we used a modified version of the K-mean algorithm [6] that is traditionally used in social media analysis [7] in order to cluster messages per context. The goal of this algorithm is to detect trendy vulnerabilities that have no relation with the current published CVEs (for now the verification is made manually).

The algorithm is presented in Figure 2 and executed as follows:

- Group the twitter messages by keywords, so that all the tweets containing the same keyword are grouped together.
- Group the messages by combination of keywords. All the messages containing the same combination of keywords are grouped together.
- Drop those groups that share irrelevant keywords like: and, the, then, etc. (using specific filtering libraries)
- Create clusters of messages sharing the same context.
- Compare the keyword combination with the recent CVEs in order to eliminate those messages that actually relate to a new CVE without mentioning the CVE identifier.

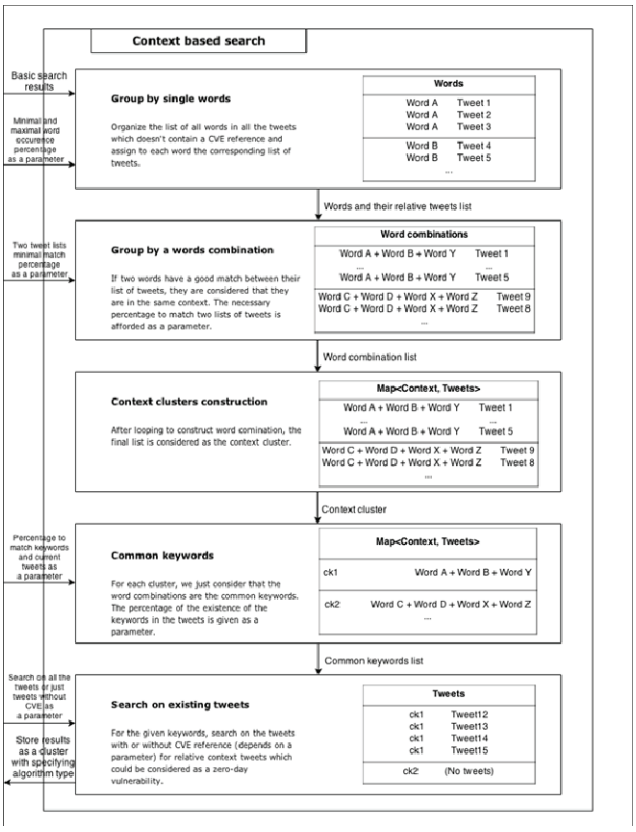

Fig. 2. Zero-day search algorithm

The final clusters correspond to groups of messages that deal with the same topic at the same period of time without any relation to a known vulnerability (CVE). This kind of information is usually related to a new 0-day disclosure that was shared by several sources and has not yet a CVE assigned. It can also contain CVE requests that are supposed 0-day vulnerabilities details sent to the NVDs asking for verification and confirmation.

C. 3.3 Algorithm for detecting CVE requests and updates

The CVE based search algorithm is quite simple (see Figure 3), 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.

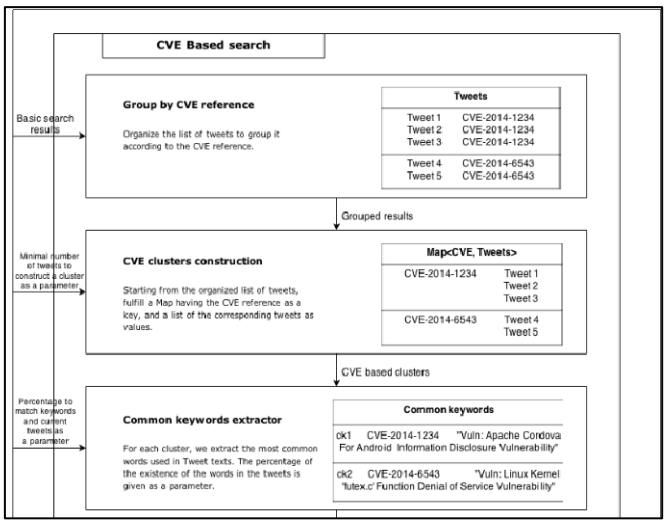

Fig. 3. CVE based search algorithm

D. Implementation

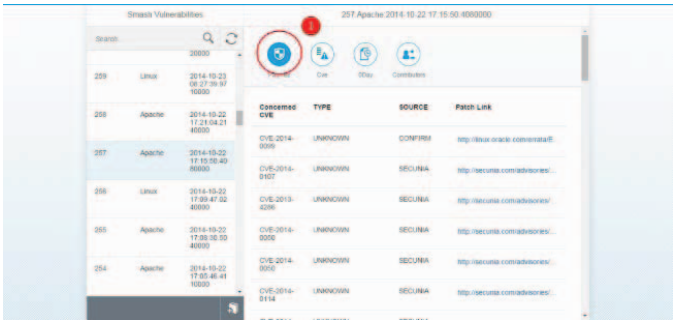

Fig. 4. Screenshot of the SMASH monitoring tool

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.

As shown in Figure 4, 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 follows:

- 0-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.
- Contributors: A list of Twitter users who provided content used in the current analysis, and whose overall trust level is displayed.
- The other two views, CVE and Patches, mainly contain information read from the NVD.

IV. SOCIAL MEDIA SOURCE TRUST MODEL

Extracting unconfirmed security information form Twitter introduces a certain uncertainty in the relevance and the quality of the identified in formation, especially if this information is not validated manually by security experts. Although if a non-confirmed information is shared by a certain number of users there is no guarantee that it is indeed correct. For this reason we proposed to create a trust and reputation system that identifies trusted Twitter users and experts. The goal afterwards is to evaluate a trust level per Twitter user with regards to the quality/validity of the information he shares. For example if the clustering algorithm detects a potential zero-day vulnerability and the user sharing this information has a high trust level the information can be taken a serious. Besides the user trust aspects, the expertise is important information that can be exploited in order to fine-tune the security information collection.

A. Reputation Model

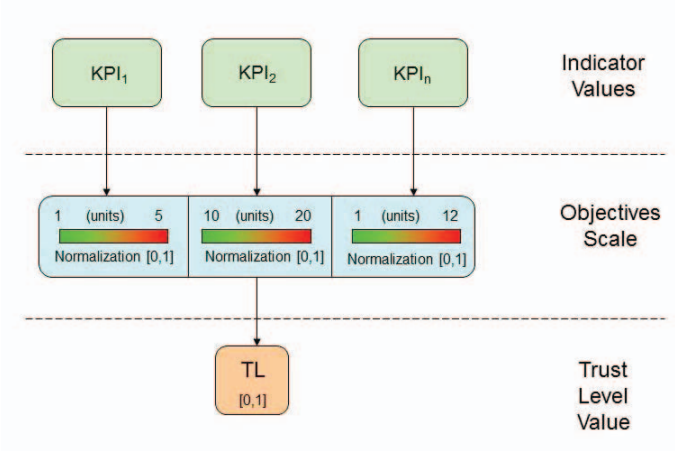

Fig. 5. KPI-based trust Model

In order to assign a trust value to SM users, we need to rely on a reputation model (in opposition to a certification model that requires a chain of trust). This reputation model is evolving over time according to the users' behaviour. The facts and observations for which the reputation should be built upon, must be quantifiable in order to iterate numerical values and compute a global reputation score that will be used as absolute trust value.

For mapping the quality of the information shared by the sources into quantifiable values, we propose to define a set of KPI (Key Performance Indicators) as metrics used to measure progress concerning the goals of relevant software vulnerability information. For that we use the KPI trust model (KPITM) [4] and [5] that is a model for which the trust evaluation is based on the experience on KPIs shared by different users.

This trust model gets as input indicator values that correspond to numerical values related to the quality of a post or the quality of the behavior of the user. These values can have different scales and different measurement units. Every indicator value is then normalized according to a minimum and maximum value defined statically or dynamically (dynamically can be for example choosing the maximum boundary according to the best indicator value measured). The normalization generates a value between 0 and 1. According to the importance of the indicator we can add a priority factor that adds some weight to the indicator. Finally, a normalization (taking into account the priorities) is made in order to obtain a single trust value comprised be-tween 0 and 1.

B. Trust Parameters

After defining the KPI based trust model, we define the different parameters or indicators that will be used to feed the trust engine.

1) Regularity

This parameter corresponds to the number of relevant posts identified per user. A relevant post is one that corresponds at some moment in time to validated information published by the NVD. Posts belonging to 0-day cluster, for instance, are validated through the fact that the NVD publishes an official CVE sometime after the post has been done. Usually a valid post is contained in a CVE cluster or a context cluster (0-day). This indicator does not distinguish the information per software component. The max boundary of the objective scale corresponding to the regularity indicator corresponds to the highest regularity value among all the users. This indicator gives a hint about the honesty and the regularity of a source user.

2) Quality

This parameter corresponds to the quality and the completeness of the information provided by the user source. We measure the quality of the information if it contains a link to an URL pointing to a trusted known domain (we maintain a list of trusted security domains where the content is confirmed), if the post contains a CVE reference, if the post contains a clear definition of the software component. This metric distinguishes between posts that only contains URLs or only contains few details about the vulnerability without any URL reference.

3) Expertise

The expertise indicator corresponds to the number of relevant and confirmed posts per software component for every user. The idea of this indicator is to identify who are the experts per software. An expert user can be considered as trusted source for the information related to the software for which she is an expert.

C. Observations and source categorization

In the setup phase of the trust and reputation system, all the indicators are set to zero. Then for every Tweet post detected by the different algorithms and flagged as relevant, the content is analyzed and the different indicator values are updated. After approximately 30 days of continuous computation the reputation system becomes stable. Stable means the values of the reputations becomes less fluctuant, the experts are identified and their position dos not really change. We obtain a unified trust value for every user.

Once we reached the stabilization period, we observed the behavior and the profiles of the different users. For this reason, we proposed a categorization of the different profiles of users.

- Bots: This category of profiles concerns the most active user sources since it publishes information 24/24 7/7 with no interruption. Usually the bot profile is connected to several RSS feeds coming from NVs or Security related websites and publishes instantly the new confirmed vulnerability when it is published on the NVD.
- Human CVE sources: we identified human user accounts that publish in a non-automated fashion new CVE that they heard about. Usually they are not covering as many CVEs as the bots.
- Request for CVE Sources: When a new vulnerability is discovered, some developers can ask NVD to confirm the vulnerability by creating a new CVE. Some-times these requests for CVE are published before the confirmation from an NVD.
- Zero-day sources: very rare profiles composed by users regularly reporting 0-day vulnerabilities confirmed later on (manually for the current version of our model).
- Developer Expert sources: this category represents the cases found in [12], meaning users that are real software developers (open source most of the time) that use social networks to share information, bugs and vulnerabilities related to the software they develop. Most of the time these profiles are mainly publishing about one specific software.

The goal of this categorization is the better understanding of the different profiles that we can find on twitter and adapt the score calculation result. It is unfair to give a very high-ranking score for a bot due to his presence and a very low ranking score for the 0-day expert due to the rarity of his publications. This categorization will help us to adjust the trust evaluation among the different profiles. The reputation model is still in study phase and more useful results will be communicated in the future.

V. 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.

A. *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.

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.

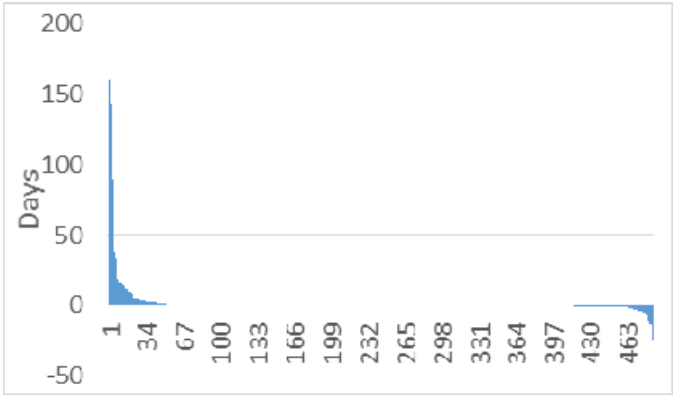

Fig. 6. CVE Publication dates comparison between NIST and SMASH

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

The most interesting result shown in Figure 6 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 on 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).

B. *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 GNU/Linux derivate. It seems that the phenomena described in [12] 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.

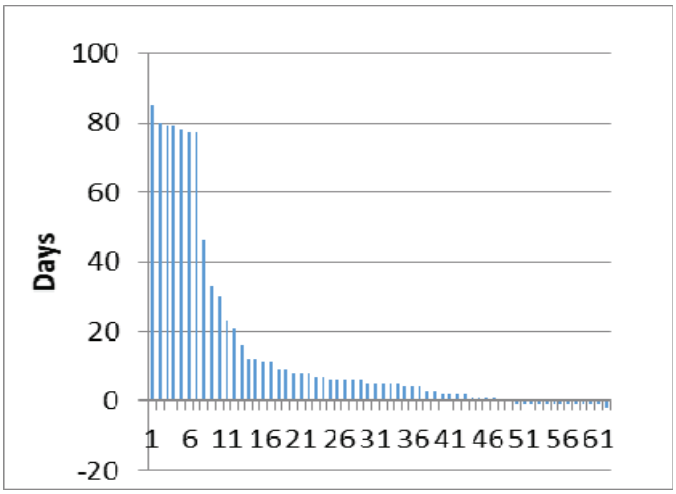

Fig. 7. Zero-day publication days

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 CVSS 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.

VI. 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. [14] 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[3] or Exploit-DB[4] for example. Another study led by Cui et al. [15] 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 [8] 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 [10] then by Surian et al [9] 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 [2] and [3] focused on analyzing software engineering trends on Twitter. The notion of software popularity appeared in these studies.

Bug tracking monitoring on social media was also addressed in [13] for open source public trackers and in [1] 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.

VII. Conclusion

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 conducted a study in this paper to observe the quality of the information collected and how to trust and rate the users sharing security information. 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 by performing an evaluation study on the obtained scores.

[3] www.metasploit.com
[4] www.exploit-db.com

Acknowledgements

A special thanks to Gilles Montagnon and Elton Mathias for their precious help.

References

[1] 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.

[2] 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 engi-neering (pp. 31-36). ACM.

[3] 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.

[4] Trabelsi, Slim, and Luca Boasso, 2011, "The KPI-Based Reputation Policy Language." SENSORCOMM 2011, The Fifth International Conference on Sensor Technologies and Applications.

[5] Böhm, K., Etalle, S., Den Hartog, J., Hütter, C., Trabelsi, S., Trivellato, D., & Zannone, N. (2010). A flexible architecture for privacy-aware trust management. Journal of theoretical and applied electronic commerce research, 5(2), 77-96.

[6] 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.

[7] 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.

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

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

[10] 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

[11] 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.

[12] 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.

[13] 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.

[14] Arafin, Md Tanvir, and Richard Royster. 2013 "Vulnerability Exploits Advertised on Twitter."

[15] 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 Heidelber