



Graz University of Technology

Institute for Applied Information Processing and Communications

Master's Thesis

Large Scale Software Vulnerability
Identification, Tracking and Analysis Using
Social Networking Services

Bojan Suzic

Matriculation Number: 0631814

Graz, December 2011

Assessor: O.Univ.-Prof. Dipl.-Ing. Dr.techn. Reinhard Posch

Supervisor: Peter Teufl, Dipl.-Ing.

Abstract

The increasing complexity of software systems and the rate at which the world is being interconnected raise the significance of a proper and effective software vulnerability management process. On the other side, the rapidly growing usage of online social networks lowered the information flow barriers and facilitated the active participation of actors involved in the process of public information exchange. This work studies the possibilities to employ potentials of online social networks in the areas of software vulnerability identification, tracking and analysis. The emphasis is placed on Twitter, a service providing real-time broadcast of short messages.

The first part of this work presents the software framework prepared for the purpose of collecting and augmenting Twitter messages. The framework connects to data sources related to domains of the software vulnerability classification, enumeration and management. The data from those sources are used to enrich Twitter messages, which are then processed and analyzed using semantic queries.

The second part of this work presents the potential usage of the software framework. It shows how the framework can be applied to discover the breakout of an unknown exploit for a popular software package. Furthermore, the work introduces a new dimension in the process of software vulnerability assessment. For such purpose, it uses the aggregate user feedback and activity to measure a relative impact of previously known and enumerated vulnerability. Finally, the work demonstrates how to perform an aggregate analysis of numerous characteristics and features of software vulnerabilities and how to use the framework to monitor, visualize and identify trends and their changes on a large scale.

Zusammenfassung

Die zunehmende Komplexität von Softwaresystemen und die immer stärker werdende Vernetzung sind Gründe für eine steigende Anzahl an Sicherheitsschwachstellen. Für den Umgang mit diesen Gefahren spielt ein korrektes und effizientes Managementverfahren eine entscheidende Rolle. Parallel dazu reduziert die stark wachsende Nutzung von sozialen Online-Netzwerken die Informationsflussbarrieren und erleichtert somit den öffentlichen Informationsaustausch. Der Schwerpunkt dabei wird auf den Dienst Twitter gelegt, der die Veröffentlichung von Kurznachrichten ermöglicht.

Der erste Teil der Arbeit präsentiert ein neues Software-Framework, das für die Extraktion von Twitter-Nachrichten, der darin enthaltenen Informationen und deren Erweiterung durch externe Daten dient. Das Software-Framework verbindet sich mit Datenquellen, die sich in den Feldern von den Sicherheitsschwachstellenklassifizierung, -Aufzählung und -Management befinden. Die aus diesen Quellen extrahierten Daten werden weiterhin für die Erweiterung und Verarbeitung von Twitter-Nachrichten und darin enthaltenen Informationen verwendet.

Der zweite Teil dieser Arbeit präsentiert die potentielle Nutzung des Software-Frameworks für die Erkennung von Durchbruch eines unbekanntes Exploits für ein beliebtes Software-Paket. Darüber hinaus stellt die Arbeit eine neue Dimension dar, die für die Verfahren von Sicherheitsschwachstellenbewertung verwendet werden kann. Für diese Zwecke, auf den aggregierten Feedback und Aktivität der Twitter-Benutzer basierend, misst das Software-Framework die relative Auswirkung von bisher bekannten und quantifizierten Sicherheitsschwachstelle. Die Arbeit zeigt weiterhin, wie das Software-Framework für eine aggregierte Analyse zahlreicher Eigenschaften und Merkmalen der Sicherheitsschwachstellen verwendet werden kann. Schließlich demonstriert sie die Anwendung des Frameworks für die Überwachung, Erkennung und Visualisierung von neuen Trends und deren Veränderungen.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

.....

(date)

(signature)

Acknowledgements

I would like to thank ...

My parents and sister, for absolute trust and support.

Peter Teufl, the supervisor of this work, for helping me with the work, providing valuable suggestions and ideas as well as demonstrating enormous amount of patience.

The IAIK and its head Reinhard Posch, the advisor of this work, for giving me the opportunity to do this work at this institute.

Karl-Christian Posch, for the support shown during the initial phase of this work.

Bojan Suzic

Table of Contents

Abstract	II
Zusammenfassung.....	III
Statutory Declaration.....	IV
Acknowledgements.....	V
Table of Contents	VI
List of Figures	VIII
List of Tables.....	X
List of Abbreviations.....	XI
1 Introduction.....	1
2 Online Social Networking and Twitter	4
2.1 Design Patterns Characterizing Web 2.0.....	5
2.2 Introduction to Twitter.....	7
2.3 Twitter and its Applications.....	11
2.3.1 Usage Patterns and Motivation of Twitter Users.....	11
2.3.2 Safety Critical Issues	14
2.3.3 Markets, Products and Investments	20
2.3.4 Disinformation on Twitter	25
3 Software Vulnerabilities	28
3.1 Definition.....	28
3.2 Software Vulnerability Taxonomies	32
3.3 Initiatives of Mitre Corporation	38
4 Software Framework.....	46

4.1	Rationale behind the Idea	46
4.2	Framework Description	47
4.2.1	TweetCatcher	54
4.2.2	FeatureCatcher	57
4.2.3	Presenter	59
4.3	Processing Steps Done	60
5	Results	70
5.1	Share Ratio of Prominent Software Vulnerability Sources	72
5.2	Share Ratio of Software Products and Vendors	75
5.3	Types of Weaknesses	81
5.3.1	CVE List Publication Dynamics	83
5.3.2	Twitter Chatter Dynamics	87
5.3.3	Dismantling the Numbers	95
5.4	Characteristics of Weakness Type	97
5.5	CVE Entry Distribution in the Term of Publication Time	101
6	Conclusion and Further Directions	107
	Appendix 1: Example of the CWE List Entry	111
	Appendix 2: Definition of Types Weakness and Category in CWE 2.0	116
	Appendix 3: OSVDB Data Model Overview	118
	Bibliography	119

List of Figures

Figure 1: Information quality comparison – Twitter and mainstream media.....	16
Figure 2: The propagation of disinformation in Twitter network.....	25
Figure 3: Taxonomy of software vulnerabilities in operating systems [RISOS76]	34
Figure 4: Example exploit [CMSEI05]	37
Figure 5: CVE entry schema definition (version 2.0).....	40
Figure 6: Definitions of elements in CVE schema (version 2.0)	42
Figure 7: Expanded element <i>baseMetricsType</i> from the CVSS scheme.....	43
Figure 8: Graphical overview of the software framework.....	48
Figure 9: Jena framework overview	50
Figure 10: CPE naming structure and example entries	53
Figure 11: Retrieval of information from Twitter Stream	54
Figure 12: SPARQL query used for the extraction of security exploits	55
Figure 13: Supported features of Presenter component.....	59
Figure 14: Activities executed by the FeatureCatcher component.....	62
Figure 15: Vendor details extracted from CPE: data excerpt.....	63
Figure 16: Product details extracted from CPE, excerpt	63
Figure 17: Product details extracted from Freshmeat, excerpt	64
Figure 18: Data extracted from CVE list, excerpt	66
Figure 19: Example of the augmented tweet.....	67
Figure 20: Example SPARQL query used to get data.....	68
Figure 21: Intermediate repetitive SPARQL query	68
Figure 22: Example markup of Twitter message.....	71
Figure 23: Ratio of tweets containing references to vulnerability sources	74
Figure 24: IBM mentions.....	78
Figure 25: WordPress mentions.....	79
Figure 26: TimThumb relative search and news trends by Google.....	80

Figure 27: Vulnerabilities published in NVD (total, including ones with CWE reference)	83
Figure 28: Distribution of CWE weaknesses in NVD CVE list.....	84
Figure 29: SQL Injection distribution among the products	87
Figure 30: Information exposure weakness among the products	87
Figure 31: Classes of software weaknesses (CWE) derived from CVE publication	89
Figure 32: Classes of software weaknesses (CWE) derived from Twitter conversation	89
Figure 33: Cumulative ratio of Twitter based reports compared to CVE publication.....	91
Figure 34: Line plot and standard deviation	92
Figure 35: Line plot and standard deviation, without types 9 and 17	93
Figure 36: Line plot describing the development of <i>Consequence Scope</i>	98
Figure 37: Line plot describing the development of <i>Impact Scope</i>	99
Figure 38: Time of introduction	101
Figure 39: Comparison between CVE last modified and published time.....	103
Figure 40: Difference between CVE publication time and timestamp of the Twitter message .	104

List of Tables

Table 1: Keywords used for Twitter Streaming API.....	56
Table 2: The cumulative number of tweets representing each vulnerability data source	73
Table 3: Aggregate distribution of the vendors	76
Table 4: Aggregate distribution of the products	77
Table 5: Yearly development of publication of CVE entries.....	82
Table 6: Distribution of CWE weaknesses in NVD CVE list.....	85
Table 7: Trends among the vendors, excerpt.....	85
Table 8: Distribution of the products and weaknesses types in selected June/July weeks.....	95

List of Abbreviations

API	Application Interface
CERT/CC	Computer Emergency Response Team Coordination Center
CPE	Common Platform Enumeration
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CVSS-SIG	CVSS Special Interest Group
CWE	Common Weaknesses Enumeration
DoS	Denial of Service
FIRST	Forum for Incident Response and Security Teams
HTTP	Hypertext Transfer Protocol
IATAC	Information Assurance Analysis Center
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
JMA	Japan Meteorological Agency
MS	Microsoft
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OSN	Online Social Network
OSVDB	Open Source Vulnerability Database
OWL	Web Ontology Language
PDA	Personal digital assistant
POS-tagging	Point-of-Speech tagging
RDF	Resource Description Framework
RDFS	RDF Schema
REST	Representational state transfer
S&P	Standard & Poor's
SA	Situational awareness
SB	Security Bulletin
SDB	SQL Data Base
SMS	Short Message Service
SPARQL	SPARQL Protocol and RDF Query Language
TDB	Tuple Data Base
URL	Uniform Resource Locator
ZDI	Zero Day Initiative

1 Introduction

With fast paced growth and development of the software industry lasting for decades, its role in everyday work and life of persons and organizations became increasingly valuable. Moreover, as the application of the software systems is infiltrating deeper in the life of persons and organizations, both vertically and horizontally, dependencies on their functionality and potential impacts in the case of malfunctions or misuses pose serious threats even more.

The current situation presents an abundance of availability of various software applications. Their production is based on shorter development cycles and performed under an increasingly competitive environment. Additionally, the same software is often maintained and deployed to a lot of platforms and distributed to users through numerous channels, often as a download option or automated update through Internet.

In this dynamic environment it is becoming more and more complicated to keep track of software vulnerabilities. This problem can be viewed from several perspectives. For the users, it is difficult to identify and assess security risks triggered by using the software; for the application developers, it is necessary to have insights into the current situation of the market and competitors, as well as to get an external feedback on the software delivered. On the other side, the software quality should be maintained and improved, while the developer should be kept familiar with the state-of-the art in the field. For the third-parties, like agencies and analysts, it may be complex to track the trends and react accordingly.

Online social networking (OSN) services pushed the new paradigm introduced with the Web 2.0 even more. They accommodated the structure, provided the solid and flexible framework, infrastructure and environment to facilitate and alleviate the information exchange and contribution among the users.

Through the establishment of the architecture of participation and application of the collective intelligence, online social networks made the web systems user-centric. They positioned them into the center of the communication, enabling the entire user base to share and exchange

information publicly. This is, among the other means, done through redistribution, rating, commenting, recommending or ranking of content, in real-time. The new architecture and approach resulted with the great amount of information flow in networked systems, with the large and increasing amount of the participants involved.

Such paradigm shift contributed even more to the information abundance and saturation. However, that great amount of information, under some circumstances, could be used to derive further knowledge and insights which were not possible or could not be undertaken easily.

This work investigates the potentials and possible usage scenarios of the software vulnerability notion and presence in the socially networked online arena. It focuses on semi-structured information flow provided by the users of Twitter, online service for real-time broadcast of short messages. The work also shows how that data stream is disseminated and connected with the other data sources from the fields of the software vulnerability classification, enumeration and management, and how it is further used to derive additional information and perform aggregate analyses of the trends.

The second chapter of this thesis deals with the Twitter services. It reveals the design patterns behind the Web 2.0 paradigm, which enabled such large user participation and growth of the popular online services. Then, the brief introduction to Twitter is presented, following with the overview on the current research and usage of Twitter in different domains. This section demonstrates some of the potentials and use cases related to the monitoring and analysis of the short messages, which are posted by the vast amount of the users worldwide. The examples have been shown for the fields of emergency management and financial markets prediction, which retrieve and derive information from the large amounts of the messages posted.

The third chapter introduces the field of software vulnerabilities. It presents the aspects and goes through issues relevant for the establishment of the definition of the term. Then, it provides the overview and development of the software vulnerability classification field, particularly with the focus on taxonomy and information exchange. It describes the enumeration and classification systems developed by Mitre Corporation, a US Ministry of Defense contractor, which are the subject of wide acceptance in the community. In this work, their databases have been used as an information source in the process of evaluation and augmentation of Twitter messages.

The next chapter deals with the software framework developed during the implementation phase of this thesis. The framework connects different sources, like Twitter messages, their related metadata, Mitre's databases and other public sources of data to perform the crawling, dissemination and conversion of the data to semantic triple representation. It further deals with the storage, grouping, filtering and augmentation of the messages from Twitter, providing the data for further analysis and visualizations of the trends found in them.

The fifth chapter presents and discusses the results of the analyses performed with the help of the software framework introduced in this work and other statistical software packages. The analyses have been done on Twitter messages broadcasted during the period from the May 15th to the September 17th 2011. The chapter covers five analyses, demonstrating the potential and variable usage scenarios of the approach.

Finally, the sixth chapter brings this work to the conclusion. It summarizes the results, provides some notices and comments and suggests further work and directions.

2 Online Social Networking and Twitter

Online social networking gained significant attention during the recent years. In the extremely short time, the services such as Facebook, LinkedIn, Orkut and Twitter noticed growth at an exponential pace, positioning themselves in the group of the most popular web sites on Internet [ALEXA11, DBLCL11].

The paradigm of the online social networking influenced and motivated many other web sites and services. They started to adopt and integrate that concept in their service offerings. Some of them even changed their business concept entirely, basing it on the online social networking idea¹.

The first section of this chapter presents the factors which highly contributed to the wide acceptance and quick growth in popularity of online social networks. It discusses some of fundamental design patterns and their context in relation to the Web 2.0. Based on the formal descriptions and practical examples, it shows how the elements such as *architecture of participation*, *networking effect* and *collective intelligence* contributed to the increased participation and value added from the given structures.

The next section provides the brief description of Twitter, a popular microblogging service based on real-time broadcasting of the short messages. Among the other online social networks, Twitter is specific because it focuses on narrow domain and activity, such as broadcasting of short messages only. Then, it focuses and specializes in providing the infrastructure which is able to handle a vast amount of the messages in real-time. Its infrastructure provides not only the real-time broadcasting of the messages, but it supports advanced integration of its services into third-party applications, which makes it open and suitable for various integrations and research.

¹ For instance, YouTube introduced communication and networking services between the members. Scribd changed from simple document sharing to online social network based around document sharing concept. Even Google recently introduced their social networking service Google+.

The following sections of this chapter provide the overview on the literature and research related to Twitter. They present the characteristics and properties of the Twitter users and their activity on Twitter. Then, several representative use cases are described. Accordingly, the results of the research of the application of Twitter in the several domains are presented. One of the domains analyzed covers the range from situational awareness and emergency situations, with the examples based on the floods, grassfires, earthquakes and the events of relevance for the national security. The other domains involved in the investigation were the relation of the market and Twitter activity, particularly in the financial market developments and Forex trading. Finally, the chapter concludes with the overview of Twitter usage and research in the domain of disinformation spreading, providing the results about the models of information propagation in Twitter.

2.1 Design Patterns Characterizing Web 2.0

The Web 2.0 concept, as it matured over the time, introduced a gradual but compelling phase-shift in the perception of its usage. There are some noteworthy aspects of this evolution which are interesting for further study. One of them can be described by the term *architecture of participation*, which Tim O'Reilly articulated in 2003 [OREILLY03]. In this context, the author relates it to the nature of the systems designed for user contribution, affecting their quick and wide adoption.

There are several examples of the systems or platforms, designed in such way that the wide user contribution is possible or encouraged, with the entry-barriers set low. For instance, the success of the open-source software may be partially attributed to that element. The recognized products, such as the Apache HTTP Server, the Linux Kernel or Perl owe their success to the numerous users who contributed to their development.

The basis of these tools has been designed so that the code and other interdependence are possibly avoided or minimized and controlled. The whole complexity of adding of the new software module is shaped in such way that the developer has to spend as minimally effort as necessary, trying to figure the entire system or to dig into the tight layers of interdependence. Based on that, the entry-barrier for the contribution is lowered. Therefore, the users

(developers) are stimulated to extend their customized contributions and will submit them for inclusion as a module or distinct product.

A similar pattern of architecture of participation may be noticed in other, possibly non code or programming related projects. For example, the IETF documents describing Internet standards are (especially earlier) prepared and created in such way that the participation of everyone interested is possible, no matter of company affiliation or financial backing. The competition there is based around ideas, without dependence or relation to the money or representation level of the participators.

The *network effect* is the next characteristic exploited by the Web 2.0, describing the additional value of a service potentially available to the user that arises from the number of other users consuming the service. The rationale to this statement can be traced to the Metcalfe's law, which hypothesizes that, while the cost to build the network grows linearly, its value grows proportionally to the square of the number of users. Metcalfe's law has been used to explain growth of various technologies, ranging from the faxes, mobile phones, to online social networks. Some authors question Metcalfe's law [METCAL05], stating that the primary limitation of Metcalfe's observation is the fact that it treats all connections in the network as equally valuable and thus it overestimates the value gained from the network growth. However, they propose a slightly different rule based on the $n \log(n)$ value scaling, which is still above linear growth and, therefore, do not neglect the importance of the network effect.

The value in the network effect may be observed in the case of Semantic Web technologies. While the semantic linking between instances in documents and ontologies or between ontologies may be considered like a graph space, merging of two ontologies using, say, an OWL *sameAs* inference will grow this space and expand more connections between elements previously unrelated. The additional inferences can produce even more connections. This way, the value of the network may be raised unproportionally with each addition.

The next key facet leveraged by Web 2.0 evolution is usage of *collective intelligence*, referred also as *wisdom of crowds*. Collective intelligence may be defined as the ability of the group to solve more problems than its individual members [HEYF99]. The other definition mentions collective intelligence "*as a fully distributed intelligence that is continuously enhanced and synergized in real-time*" [PLEVY98].

In the context of Web 2.0, this phenomenon is used to describe the process where useful information or conclusions are inferred on the basis of the user contributions or behavior, usually on a large scale. The example of such process may be found in the collaborative spam filtering products like CloudMark². Its system collects, aggregates and analyses the individual judgments of email users about their email (whether it is spam or not). This way, the messages are not analyzed separately but on an aggregate level, based on collaborative decision of a large group of users.

The other example in this direction may be Amazon and its system which invites and stimulates users to participate actively on various ways. It has a bunch of reviews of products generated by the users, which are further rated and commented. Amazon carefully and in detail analyses user input and behavior on the web site. From this collective, collaboratively gathered information it constructs a base for further ranking, suggestions and search results. This is a continuous, dynamically driven procedure which repeatedly processes gathered information in order to derive new information and improve its output.

Finally, as the particularly illustrative example of collective intelligence the phenomenon of the blogosphere may be mentioned. The blogosphere consists of the dispersed blogs, yet acting as a connected community with ceaseless activity reflected through numerous publications, discussions, or (symmetrical) links. In the blogosphere, users constantly rethink the actual topics and discuss them. Such activity builds a structure which is very interesting for research by the analysts or companies running the search engines. In this structure, the blogger and the user are paying special attention to each other. This way, they act as an intelligent filter, making it possible for the third parties to distinguish and rate an importance or significance of individual entries, trends or attentions.

2.2 Introduction to Twitter

Twitter was developed in 2006 after years of work by Biz Stone and Evan Williams. It allows users to post short-text messages called *tweets*, no longer than 140 characters. At its inception Twitter was one of applications from the Web 2.0 wave. It was conceptualized as a specific

² <http://www.cloudmark.com>

combination of blogging, instant messaging and short messaging service, later known as a *microblogging*.

As of March 2011, there are about 140 million of messages posted daily from roughly 200 million registered accounts³, which represents an increase of 280% compared to the previous year. The highest record of messages posted per second is 6,939. The number of employees is 400, which is an increase of 15% compared to the previous year [TWITTER11].

The main concepts in Twitter are *tweets*, *followers* and *followees*. The *tweet* is any message posted to Twitter by the user, sometimes referred to as a *Twitterer*. All tweets amount 140 characters or fewer. According to Twitter Help⁴, the message length of 140 characters is selected because in many systems a standard text message contains 160 characters. Therefore, the rest of 20 characters is reserved for the user name. The concept of *following* represents an asymmetric relationship between Twitter users, making it possible for a *follower* to receive information e.g. regular status updates from *followees*.

The *timeline* represents the other concept introduced in Twitter, used to describe a collected stream of messages listed in real-time order. Although the timeline message stream usually refers to messages from the accounts followed by user, the timeline may consist of other types of messages, like search results or aggregated messages from lists.

The *retweet* is the original message forwarded by follower. Retweeting enables followers to push received information to their network of followers, making them a bridge between communities of Twitter users.

Boyd et al. [TWRTW10] identified several goals playing the role in a user's decision to retweet a message:

- to amplify or spread tweets to new audiences
- to entertain or inform an audience, or as an act of curation
- to comment to someone's tweet, often by adding a new content to begin a conversation
- to make one's presence as a listener visible
- to publicly agree with someone

³ Based on information from BBC, <http://www.bbc.co.uk/news/business-12889048>, Mai 2011

⁴ <http://support.twitter.com/groups/31-twitter-basics/topics/109-tweets-messages/articles/127856-about-tweets-twitter-updates>

- ❑ to validate others' thoughts
- ❑ as an act of friendship, loyalty or homage
- ❑ to recognize or refer to less popular people and content
- ❑ for self-gain, e.g. by gaining followers or reciprocity
- ❑ to save tweets for future access

As it is the primary mechanism of *information diffusion* in Twitter, retweeting has been further studied in several works [RTWFACT10, TWNEWS10, WMOUTH09 and TWRTW10].

The *hashtag* is the other concept present in Twitter, invented among the user population. It is often used to mark a keyword or topic in a Tweet by putting a hashtag symbol (#) before relevant keywords in the message. Hashtags are often used to categorize messages, to make their finding (retrieval) easier as well as to find trending topics by third parties. Later Twitter introduced other options, like *Tweet Location*, making it easy to the users to publish their location in a structured way, or *lists*, enabling users to categorize other users in the lists and follow their messages on an irregular basis, independently and outside from the personalized Timeline.

Although, in many works, it is considered like a *social network* [PRIVTW10], some authors question that to some extent and suggest the term *information network* [FBTW10] as a more appropriate explanation of its function. They refer to Twitter's ground design, which focuses on the ability to deliver the information as quickly and as widely as possible. Some authors argue that Twitter should be set apart from other OSNs as some of its characteristics are more oriented toward news media than to social networking [TWNEWS10]. In their research they found the *following* relationship is mostly not reciprocated (*not so social*)⁵, what may not be a reflection of social relationships but active subscription of tweets e.g. information.

In their work Mills et al [TWRESP09] compared Twitter to their ideal communications platform or service. According to them, important aspects of this service should be:

- ❑ web based
- ❑ low cost
- ❑ power efficient and scalable
- ❑ easy to use or accessible
- ❑ mobile

⁵ Only 22.1% of user pairs follow each other. Comparison to others: 68% Flickr, 85% on Yahoo! 360

- reliable
- fast
- one-to-many capable
- GIS capable
- support for analytic and visualization tools
- strongly connected with local TV, radio channels and news outlets
- able to receive, generate, provide and usher useful and critical information from a variety of sources

In the analysis, they found that Twitter matches exceptionally well with those requirements. As the advantages, they mention *low-power requirements* and *omnipresence* – access to Twitter is possible also with PDAs, cell phones or similar devices with low-power requirements and high battery efficiency, which ensures operation even in the case of general power failure. For instance, radio and TV local information networks usually may be down during emergency events, especially in remote areas, therefore, not accessible⁶.

Easy-to-use property is set by authors as a prerequisite in order to achieve the scalability of a service. In terms of reliability, the authors noticed that Twitter has problems with the uptime⁷. However, these do not reflect the architecture of the platform but implementation only. In the meantime (since the publication date of research), the stability of the service has considerably improved.

The important advantage of Twitter over voice-based or other similar communication channel is its level of demand for the traffic. As the Twitter message consists of only 140 characters⁸, its operation has *low-bandwidth requirements*. Therefore, there is a higher probability for the message to be delivered in congested environments, even if it has to be processed in queue. The voice communication, for instance, in the congestion mode is not possible to carry out at all. Also, the length limitation of Twitter messages assures that only the most relevant information will be sent, making possible to have less distraction in noisy and dynamic situations.

⁶ Depending on the situation, the Internet connection could be still available through the mobile phone networks, at least for a short time period after the general power loss. Also, the cell towers are usually backed with the batteries, solar systems or diesel power aggregates for emergency situations.

⁷ At that time 1-2% downtime even in normal conditions.

⁸ Like SMS message.

Up to today, Twitter has been actively adopted by users and utilized in many areas. The investigation of its usages and potentials has been conducted by many authors worldwide.

Twitter has been used in emergency situations like water floods or earthquakes [NATHAZ10, TADOPT09 and SOM10], in investment and stock market area [TWTRADE10]. It is used even to get or suggest health related information [SFLU09], while its malicious operation can be traced to spreading and controlling malware and botnets [SOOD11, KOOFACE10].

Further applications and usage of Twitter are discussed in more detail in Section 3.

2.3 Twitter and its Applications

There is a growing amount of research directed toward the exploration of possible uses of Twitter in diverse applications. The topics range from the descriptive analysis of Twitter usage patterns, user behavior, motives and expectations, to the usage in narrow fields, in the term of harnessing collective intelligence or determination of the collective mood. The following provides details of relevant and representative research from the field.

2.3.1 Usage Patterns and Motivation of Twitter Users

Microblogging services revolutionized the way information is provided and consumed in an online world. Compared to blogging in the classic sense, microblogging stimulates less but more frequent updates from the users. The limitation in message length, which numbers 140 characters in Twitter, has as effect two main consequences.

At the first, it lowers the access barrier for the users. As in the classic blogs it is often required or expected to write a (comparatively) longer post, the users are not always in the position or motivated to invest the effort and the time necessary. Writing a blog post may be also related to the writing style and general structure of the work, which may reflect on the user's perception and reputation in the community. However in the microblogging, structure, style and grammatical/lexical consistency are in the background – due to the post size they are next to unimportant. Informal nature of microblog moves focuses more on the information provided and its value.

The next noteworthy aspect of microblogging related to the message length is its omnipresence – both in terms of production and consumption of microblogging. As the message is short, it can be sent from a wider range of devices more easily. It is easier to send or read short status updates or interesting information from the cell phone⁹, than to write a (lengthier) blog post on the small keyboard or read it using the small display screen. For the preparation of the blog post it is sometimes required from the poster to search for references and include them in the message. This activity requires some effort both on the standard desktop or laptop computers. However, it requires slightly more effort on smaller devices like PDAs, cell phones or tablets.

The second consequence of the length of Twitter message relates to the frequency of update. While the most active bloggers could send new posts once in a few days, in the standard microblog usage average user may post its updates even several times a day.

In order to get more detailed insight into user behavior and habits in social networks, Microsoft conducted research in early 2011. They published the infographic based on their findings [MSFT11]. According to it, 50% of Twitter users access the service using the mobile interface. More than 33% of Facebook users access their service using the mobile device, while for other social networks, the number of mobile users flows around 30%. This fact emphasizes stronger and wider user adoption and presence of Twitter compared to other OSNs. The nature of microblogging based on the paradigm of short messages, frequent updates and pure information surely contributes to this number.

In regards to user intention, it is interesting to note that, since its beginning, Twitter officially prompted users to share the answer on the question “*What are you doing?*”. Since then, many users have found other creative ways of Twitter’s usage. They shared their personal feelings, statements, comments. The companies have been experimenting with different methods of approaching their customers and markets using Twitter. As a result of user adoption and wide proliferation of Twitter service, a shift from user, personal centric view has been made towards a pulsing, permanently active and unique information network. As a reflection of this change, later in 2009 Twitter altered the original question to “*What is happening?*” [TWEET09].

The study of Java et al. investigated the user intentions on Twitter. Their study found that the main types of user intentions on Twitter include daily chatter, conversations, sharing

⁹ Usually, the SMS (Short Message Service) message length is up to 160 characters. This is one of the primary functionalities of the cell phone, therefore sending a microblog post may be perceived similar to sending a SMS message based on the effort required from the user.

information and reporting news [JAVA07]. Furthermore, they found that the most users of Twitter may be allocated to one of the following categories: information source, friend and information seeker. Their findings may suggest that, since its early beginning, the users have identified more possible usages and potentials of Twitter that could exceed original intentions of its inventors. Similarly, the research based on 100,000 Twitter users conducted in February 2008 [CHIRPS09] identified three broad groups of users based on the connections they had with others. The authors characterized them as broadcasters, acquaintances, while the third group of users having significantly more followees than followers has been identified as *miscreants* or *evangelists*.

The other study conducted in 2009 by Heil and Piskorski reveals intriguing patterns of Twitter users [HEILPISK09]. They found that men have 15% more followers than women. According to the same research, men also have more reciprocated relationships. Also, there is nearly double probability that man will follow man than woman (65% versus 35%).

The other interesting finding sets Twitter somehow different than other social networks. In the other online social networks, the most of activity centers on women, where men follow women they do and do not know, and women follow other women they know. However, on Twitter both men and women find the content produced by men more compelling. Explanations for this finding may be found in the fact that, in Twitter, there is no additional content provided, as photographs, detailed biographies etc, like it is in some other online social networks. Also, Twitter focuses more on the activity of the information sharing, while in some other networks even the existence of the personal or social relationship matters.

The next characteristic of Twitter usage pattern compared to other online social networks may be found in level of users' contributions. A typical Twitter user contributes relatively rarely. There is evidence showing that 10% of Twitter users contribute to 90% of its traffic. For comparison, in Wikipedia top 15% of users account for 90% of Wikipedia's edits. It is already clear that Wikipedia is not primarily a communications tool. Based on that fact Twitter could be considered more like one-to-many, mostly one way publishing service than two-way peer-to-peer network.

There are also other differences, which distinguish Twitter users from other Internet users. Based on research from Lenhart and Fox published in 2009 [TWSTAT09], Twitter users are more likely than others to connect to Internet wirelessly. The 40% of the Twitter users utilized their

cell phones to connect to Internet, while only 24% of other Internet users used the same mean to access the Internet resources.

Moreover, it is more likely that the average Twitter user consumes news on mobile devices. That implies higher rate of reading newspapers online¹⁰ and more than double rate of users reading a newspaper on smartphones or cell phones¹¹.

Although it may be guessed that among online social networks Twitter covers the segment of the younger user population, according to this report it is not the case. Concretely, with a median age of user of 31 years, Twitter is ranked above Facebook (26) and MySpace (27) and below LinkedIn (40). The authors concluded that Twitter users engage with news and own technology at the same rates as other Internet users, but the way they are using technology “*reveals their affinity for mobile, untethered and social opportunities for interaction*”.

2.3.2 Safety Critical Issues

Because of its ubiquitous nature, Twitter is used not only to report what the person is doing at the moment, or how it feels, but also to spread information about events. The unexpected events, like earthquakes, threatening weather conditions, natural catastrophes and disasters may induce exceptionally strong personal feelings like shock, upset, or fear, stimulating users to share that information with online community. As part of their postings related to such events, users may provide additional information too. Analyzed on a large scale, those postings may provide useful insight, which can contribute to establishing better *situational awareness* (SA).

SA may be referred as a state of idealized understanding of what is happening in the environment consisting of many actors. The definition from Sarter and Woods says it is “*based on the integration of knowledge resulting from recurrent situation assessments*” [SARWOOD91].

That definition may point out to the challenges which should be overcome in the dynamic and noisy environments like Twitter. Although most of the literature covers situational awareness in the context of military and aviation operations, it has been researched in other domains like weather, emergency response, transportation or civil engineering. [PARAMO96, OLOUFA03].

¹⁰ 76% versus 60% are less likely to read printed newspaper (52% versus 65%) than other Internet users

¹¹ 17% of Twitter users read newspapers on smartphones, compared to 7% of other Internet users.

Another research field of relevance in this area in the literature is often described as *crisis informatics*. This discipline addresses social and technological concerns in emergency and crisis response, aiming for socially and behaviorally-informed development of information and communication technology for such situations. With the newer developments introduced, the process of disaster research is being enriched with on-line investigation components, including also *crisis communication*.

In the study related to command and control in battlefield operations, the work of Sonnenwald and Pierce [SPDGROUP00] emphasizes that situational awareness addresses individuals who “*must work together to collect, analyze, synthesize and disseminate information*”. The source for this statement can be found in the fact that diversity and complexity in the battlefield increase on such way, that it is not possible for a single individual to acquire and assess diverse and rapidly expanding information sources and react in a timely manner.

The similar flow of mixed and diverse flow of information we may find in online social networks. Therefore, the research in the theory of situation awareness in general or particular fields may give useful insights or directives in how useful information could be recognized and retrieved in computer mediated group communication, including individuals, groups or communities.

In order to be able to measure and evaluate information and its usefulness provided in Twitter, Mills et al [TWRESP09] investigated Twitter usage in various emergency situations. Based on their observations, they analyzed information gain and tradeoffs related with Twitter usage in emergency communications field. This specific measure they name *information reliability and quality*.

The general finding shows that Twitter is essential as a medium for information retrieval, especially during the first hour. Within 24 hours, mainstream media arrives to the average level of information quality on the Twitter network. Furthermore, authors inferred that after one week mainstream media and specialized information outlets provide higher chance to find critical information. Their finding is graphically approximated in Figure 1.

Although authors refer to this relation as a rough estimate, additionally they noticed that with Twitter there is potentially more information than with any other information network, but the variability in quality and sheer quantity may make it difficult to recognize, retrieve and manage.

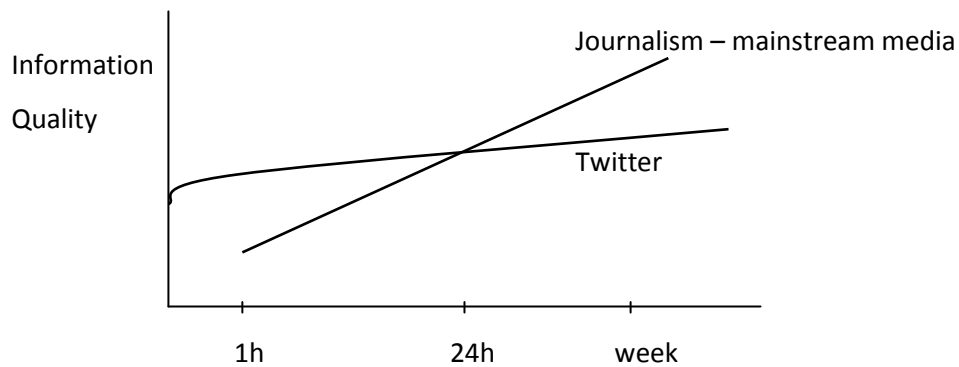


Figure 1: Information quality comparison – Twitter and mainstream media

Practical application of Twitter in real emergency scenarios has been studied in this context in many occasions. Vieweg et al in their work [NATHAZ10] investigated two disaster events in the US: the Red River floods and the Oklahoma Grassfires, both of which occurred in the spring of 2009.

Red River Floods occurred in spring 2009, in the Winnipeg community, in North Dakota. Previously, National Emergency Center issued warnings and flood predictions for Fargo community, located upstream. The floods in the upper stream have been prevented due to previously built dikes and massive sandbagging. However, in the downstream near the Winnipeg the ice jams blocked the regular flow of Red River and prevented the opening of the Winnipeg Floodway¹², which caused several flood threats for Winnipeg residents culminating on April 8, where the floods in some areas stayed for weeks afterwards.

Oklahoma Grassfires occurred also in spring 2009. High winds and dry conditions contributed the fueling of numerous grass fires in central and southern Oklahoma and parts of northern Texas. During April 9th and through mid-morning of April 10th there has been immediate fire threat in this region. As a consequence, many roads were closed and residents evacuated as firefighters tried to control rapid spread of fire.

In their work, Vieweg et al used Twitter API interface to get the tweets from a 51-day window for Red River floods and six-day data window for the Oklahoma grassfires – in both cases for the

¹² A man made channel to divert excess waters around the city

period for which the threat has been identifiable and active. Based on selected keywords, data collecting resulted with 13,153 and 6,674 tweets, generated by 4,983 and 3,852 unique authors, respectively. On those sets later the filtering technique has been applied, in order to keep only relevant tweet data and remove unrelated posts. The filtering has been done in two stages – first keeping only the user data streams that contained more than three tweets containing the search terms. Then, those tweets have been classified in two groups, on-topic and off-topic, based on the level they mention emergency case. Finally, the resulting on-topic tweets were further investigated and locations of their authors have been manually extracted and reviewed.

The next step included feature-labeling in resulting dataset. The features related to geo-location¹³, location referencing¹⁴ and situational updates¹⁵ have been referenced using automated and manual extraction. In the analysis of these features authors identified *High Yield Twitterers*. This term has been assigned to users who cautiously constructed their tweets to report as much relevant information as possible within allocated space. The authors suggested that this category of users is somehow aware of their public role, or their perception of the public role, and design content-rich, deeply informative tweets to be read by a larger audience intentionally.

The next interesting finding of this study is related to re-tweeted information. *Retweeting* is a form of the convention among Twitter users, a passing of previously broadcasted tweet. The tweets in question are considered especially noteworthy and interesting by the users forwarding them. According to the results gathered, the tweets containing geo-locations and situational updates are more likely to be retweeted, indicating a preference of other Twitterers.

Markedness is the next phenomenon noticed in the set, which in this case refers to generalization of some terms like places, landmarks or items¹⁶. Those entities are not referred by their name but according to the category they belong or using other way, which is supposed to be taken for granted by other users. This trend may be of importance in information

¹³ Clearly identifiable information that includes street addresses, intersections, city names, county names, highways and place names like schools, landmarks etc.

¹⁴ Information that uses one place as a reference for another; or mention of a location using a landmark

¹⁵ Identified and organized into several categories, like *warning, preparatory activity, flood level, weather, damage reports, road conditions* etc.

¹⁶ For example we may use following sentence: “*The water level of the river is increased rapidly*”. Here, the author is referring to one particular river in its neighborhood. Although it may be self-evident for human reader, computer processing of such sentence requires additional activity in the term of reasoning.

extraction techniques as it affects the ability of the system to examine and understand analyzed data correctly.

As a result of this research, authors proposed an outlined set of relevant microblog-enhanced situational features which may be of importance in further emergency data mining. These features are categorized into the groups, including subcategories for particular cases.

In other work from Sakaki et al [SOM10], the authors focused on real-time detection of earthquakes in Japan. Their solution is considered as an innovative social based approach in early earthquake detection. The basis assumption of this work is centered on the Twitter user which represents *social sensor*, while each tweet posted is *sensory information*. However, these sensors are of a vast variety and pose many different characteristics. They may be inoperable, malfunctioning; some of them may be unusually active or very noisy. The basic object of research is an *event*, which represents arbitrary classification of a space-time region. An event may be in relation with actively participating agents, products, passive factors or a location in a space-time. The properties outlining the earthquake events are their large scale nature, their ability to influence people's daily life and their both spatial and temporal features. There are also other events, like typhoons, traffic jams, storms, hurricanes etc, which may pose the features outlined. The proper rules and techniques to distinguish these event types are crucial in the analysis.

In work from Sakaki et al three main groups of features of each tweet were identified:

- Statistical features: including number of words in each tweet, position of the query word inside a tweet
- Keyword features: the words present in a tweet
- Word context features: the words within the tweet related to the query word, like previous or the following word

These features are used to determine which tweets are referring actual events (positive) and which are referring other events (negative), as ones happened in the past. Based on the research results, keyword and word context features did not contribute significantly to the classification results. This fact may correlate with the finding that users usually tend to send shorter tweets when they are surprised. On the other side, just position of the words in relation to the query word may not be enough for the system to bring decision. The authors identified

the cases where it was difficult even for the human to judge whether a tweet is reporting an actual earthquake or not.

The performances of the proposed system have been evaluated on 49,314 tweets retrieved during one month. After classification, 6,291 positive tweets by 4,218 users were identified. The rate of identifying the earthquakes classified as 3 or more at JMA seismic intensity scale was 96%. The value of this system in the sense of the loss prevention is its speed, e.g. ability to deliver information timely. Compared to the system already used by JMA to deliver announcements of the expected seismic activities and expected arrival times, the proposed solution is able to deliver information using e-mail in the time less than a minute. JMA's actual system using TV and radio broadcasts usually takes 6 minutes to deliver information report. In many cases, this presents unacceptable delay, as the earthquake waves propagate with the speed of 3-7 km/s.

Interesting research has been conducted by Hughes et al [TADOPT09]. They investigated Twitter usage during mass convergence events, namely two emergency¹⁷ and two national security related¹⁸. The four events happened in time range from 21st Aug to 14th September 2008. Similarly as for other described works, the authors used Twitter Search API and periodically extracted messages containing highly narrowed search terms (containing only names of hurricanes or conferences, respectively).

After sampling of the search results and initial analysis, authors noticed a correlation between the number of messages related to the event and its impact. For instance, Hurricane Ike has caused damage financially estimated at \$27 billion, while Hurricane Gustav caused damage in the range of \$4-14 billion. Total number of tweets sampled follows this pattern in relation to the event. Similar case is also for both conventions organized.

The next characteristics related to usage of Twitter in such occasions can be found in comparison of the number of tweet replies in standard Twitter traffic and tweet replies in such events. Authors found a strong difference in this sense. While during mentioned events the number of reply tweets counted between 5-7%, in general sample of Twitter traffic reply tweets accounted for more than 20%. The authors hypothesized that one of reasons for such difference may be a decision of users not to direct the traffic toward one particular user but rather to the whole group. Additionally, the authors have found that tweets tend to contain URLs more often

¹⁷ Hurricane Gustav Hurricane Ike

¹⁸ Democratic National Convention and Republican National Convention

in case of mass events than in normal traffic. In the case of mass events, some 36-52% of tweets contained URLs, while in the sample of standard Twitter traffic URLs have been contained in around 25% of cases.

Another compelling finding of this research is related to user adoption of Twitter. The authors compared Twitter adoption¹⁹ level of new users during mass events investigated, and new Twitter users based on a random sample. The results suggest that user adoption rate is higher for users who began using Twitter as a part of important situation related to direct experience and usefulness of the service.

2.3.3 Markets, Products and Investments

Considered from the point of regular fluctuations and their volatility, financial markets and investment products are surely one of areas where analysis of microblogging messages may be applied. Psychology of consumers and traders with its unquantifiable and hard to understand or explain characteristics is often overlooked when it comes to general trading forecasting. In this sense, Arthur et al [ASSET96] identified two main views of understanding, represented from academic theorists and market traders.

The first group tends to estimate the future value of assets rationally, based on all market information available. From the point of efficient-market hypothesis, it is believed that there are no opportunities left for speculative consistent profit, following that temporary price shocks are reflecting rational changes in assets' valuations rather than sudden shifts in investor sentiment. From the other side, market traders believe that technical trading²⁰ may be profitable and that behavior of traders can be affected and driven by herd effects, unrelated to market news. This can be often identified as an effect or representation of "*market psychology*", supported by areas like behavioral economics [ASSET96].

Therefore, as microblogging messages may be a result of user's sentiment and rational or irrational expectations, their content may be considered as a valuable source in the analysis of market trading and forecasting. In some past research, message boards on Internet have been analyzed for valuable financial information. There have been reported correlations between

¹⁹ In the term of continuous service usage for over of 17 weeks.

²⁰ Prediction of future asset valuation based on past data, like trading price and volume.

message volume and trade activity and volatility. Antweiler and Frank [MSGBRD04] have shown that talk in internet message boards is not just noise, but can be a source of financially relevant information and furthermore could be used as a valuable resource in studies of events and insider trading. Other research suggests that message boards as a communication instrument introduced changes in pricing behavior. Empirical evidence found by Jones [JONES06] shows a significant increase in daily trading volumes, lower returns and higher volatility after a company's message boards was established. This may be a result both in cases when the new investors were drawn to the market, or existing investors were stimulated to trade more frequently.

However, in comparison with Internet message boards, microblogging has several additional features which may be considered worthy for distinct consideration. First, Twitter broadcasts messages in real time. That means the information can be retrieved instantly. In order to retrieve actual information from the message boards, they should be crawled frequently, which may introduce problems on a large scale.

Second, Twitter aggregates information, there is only one place to search and retrieve information from. Unlike Twitter, there are many message boards, in different locations, based on different software etc which makes retrieval process more challenging. Additionally, it affects posting behavior of users as they need to enter different message boards periodically, send new posts or take a part in discussions.

The third difference may be found in rating of users and their influence. In the case of message boards, it may be hard to identify the same user posting on multiple message boards under different nick names. On Twitter, the average user will usually post from one account. In Twitter, the value of the author and the posted information may be particularly estimated through the number of retweets, followers or followees – the dimensions which do not have a counterpart in the case of message boards. Furthermore, availability of such features makes it possible to introduce measure and research on information diffusion. Based on these arguments, we may conclude that, compared to message boards, analysis of microblogs like Twitter may introduce some additional advantages.

One of the very first works investigating microblogs for an impact on financial markets is prepared by Sprenger and Welpé in December 2010. Their research concentrated around two basic questions:

- 1) Whether and to what extent the information content of stock microblogs reflects financial market developments, and
- 2) Whether microblogging forums provide an efficient mechanism to weigh and aggregate information.

The first question implied a research on the possibility to predict returns based on tweet sentiment, whether message volume is related to returns, trading volume or volatility, and third, what is the correlation between the level of disagreement among messages and volatility or trading volume of assets. The second question directs the research toward investigating whether the quality of investment advice is in relation with level of mentions, the rate of retweets or author's followership.

In this work, authors collected 249,533 stock related microblogging messages containing the dollar-tagged ticker symbol of an S&P 100 company, in the period from January 1st to June 30th 2010. Based on the sentiment (bullishness) presented, the tweets have been classified as buy, hold or sell signals, based on Naïve Bayesian text classification.

The authors noticed significant spikes of message volume just before the opening of the markets. Additionally, the fact that the majority of tweets were posted during the trading hours provides the evidence that microblogging in this case is used for real time communication. Furthermore, authors observed strong correlation between sentiment²¹ and returns, and from other side, slightly weaker correlations between sentiment and abnormal returns.

The next relation found in this research is concerning the relationship between message volume and trading volume. Interpreted as elastic measure, they found that 1% in an increase of message volume produces 10% increase in trading volume; however, their finding suggests also that returns cannot be explained with message volume. Finally, in this work authors observed the increase of volatility as the message volume rises.

The further research in this work from Sprenger and Welpé is related to *information diffusion*. They tried to examine whether individual messages with high quality information are weighted more heavily and spread through retweets. As an addition to that, authors wanted to investigate whether advisors with higher potential can be identified and whether these users receive greater attention in the community.

²¹ Sometimes referred as a *bullishness*, referring to trading recommendation identified in message

Although it could be concluded intuitively that retweeted message may contain information of more value, the empirical findings of authors did not prove that. Actually, the difference of quality of information between retweets and non-retweets is statistically insignificant. However, there may be reasons why significance is marginal: the retweets are often modified original messages with additional comment of retweeter, which can alter the original tweet sentiment and thus the quality of the information. Additionally, the messages can be retweeted after longer delay, becoming irrelevant for the day of the investigation. Anyways, the authors dropped the initial hypothesis and focused further on the identification of more than average users. There, the evidence has been found that users who provide higher quality investment advices are retweeted more frequently; they also have larger followership.

Other research, conducted by Vincent and Armstrong in the area of finance and automated trading [BPOINT10] analyzed Twitter buzz with regards to Forex²² trading. Namely, timely identification of points of change in financial time series in the trading of foreign exchanges is especially significant for successful trade. As the currency exchange rates fluctuate very often, the prediction of the amount and direction of change should be done frequently and in a timely manner.

In this work, authors analyzed a time span of 5-months, from the October 2009 to March 2010. During that time, they made around 100,000 observations using the service Twitscoop²³. In the observations, new word instances for a particular period have been identified. In the case of two words or more, Twitter Alert has been triggered and that represented a possible breakpoint in the series. These alerts have been used on previously developed genetic algorithm for automated currency trade between USD and EUR. Using this approach authors have been able to determine a correlation between buzz found on Twitter and USD/EUR exchange rate. The correlation is expressed as a relationship between the time to react to each alert and the performance of algorithmic traders. As the relationship found takes a form of wave²⁴, authors named it "Twitter Wave". It suggests that correlation is most marked in the period of 4 and 6 minutes before the break-point in the trading occurred.

Another interesting finding with regard to Twitter usage is related to brand mentioning in microblogs. Jansen et al. [WMOUTH09] analyzed microblog entries in Twitter posted in the

²² To be explained

²³ Description follows

²⁴ Profit margin in function of time, where the reference point in time is issuing of Twitter Alert

period of 13 weeks. These entries have been extracted based on mentioning of 50 previously selected brands, resulting with more than 150,000 tweets. Authors have found that 19% of the entries contained brand mentions. Of those, about 20% of posts contained some expression of sentiment related to the brand. In their dataset, more than 50% of sentiment expressions were positive, while 33% were critical to the company or product. Their finding suggests that a significant amount of the Twitter communication relates to the products or brands, which further indicates that the microblogging medium poses a competitive place for organizations, especially in the fields of marketing, customer relationship management or business intelligence mining.

As the microblogging world is constantly rising and expanding its presence, counting now with hundreds of millions of users worldwide, it has potential and should become an integrated part of overall marketing strategies of the companies. Using microblogs and their ubiquitous characteristics²⁵ organizations are able to shorten the emotional distance to customers and engage in more direct, two-way communication. This way, it could spare them many inefficient advertising activities and give an opportunity to get more direct, and what is important, real-time feedback from customers. However, that real-time component makes a microblogging world very dynamic, pulsing and fluctuating medium, requiring constant activity and attention invested from the organizational side.

Also, in the previously mentioned research it has been discovered that in 80% of the tweets containing brand mention, no sentiment expression has been found. It suggests that users were seeking for the information, asking questions or answering questions of the other users. That place may be an excellent source of business intelligence for companies. For instance, instead of (or additionally to) organizing the costly, complicated and relatively inefficient customer surveys or analysis, the companies may additionally engage in microblog arena. There they can try to find new product opportunities or ways to improve their products and satisfy customer needs better. Using microblogs, companies may push information to customers and get real-time feedback. They can also control and estimate how pushed information is further valued and perceived.

²⁵ The microblogs are updated frequently from the wide range of devices and the communication (update) is real-time based

2.3.4 Disinformation on Twitter

Due to its architecture and properties, Twitter is especially suitable for use in disinformation operations.

The limited length of the messages tends to affect inclusion of the source or reference which may prove information posted is correct. Also, the message size limitation may induce the poster to produce a simplification of information.

This simplification may influence and distort its perception and understanding, thus transforming it to disinformation. Therefore, in the Twitter, disinformation may be posted and forwarded both intentionally and unintentionally.

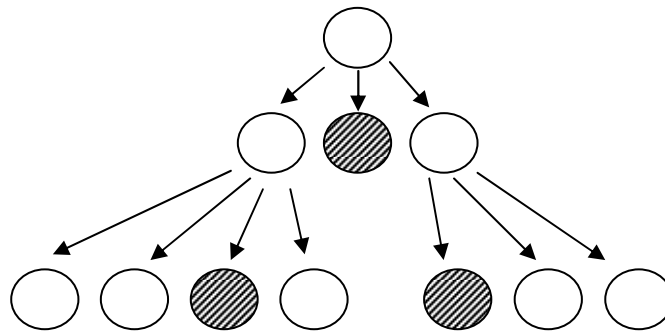


Figure 2: The propagation of disinformation in Twitter network

The information in Twitter is propagating through retweets. The rate of message propagation is proportional to the product of probability of each user retweeting the message and the number of users receiving the message. There are several factors playing the role in user decision to retweet the message. Among the factors influencing user decision to retweet identified by Boyd et al. [TWRTW10], particularly interesting may be the intention to amplify or spread tweets to a new audience, to inform audiences and validate others' thoughts. This way the particular attention to a message and its content is drawn.

Figure 2 represents a model of information propagation in the Twitter network. Being broadcasted from the source node, the information is further propagated to followers of each node by voluntary retweeting it. The shaded nodes represent the disbelieving users, e.g. ones who distrust information obtained and, therefore, ignore it. The figure assumes that retweeting rate is 100% among believers, and 33% rate of disbelief.

From the figure, the property characteristic for Twitter is observable: the users in the network are generally not aware of the disbelieving ratio among other users, especially ones which are not in their network (followers-followees).

Unlike in IRC, internet forums or email mailing lists, there is no mechanism in Twitter for sibling nodes to communicate, unless they already have established direct relationship. If a user in Twitter tweets something false and one of users in its network refutes this information, other followers of tweeting user are usually unaware of that act. This way, disbelieving ratio (globally) expressed to particular information (message) do not affect its valuation and retweeting among users – it produces only local effects. These characteristics may stimulate malicious users to optimize particular message rather for maximum chance to be retweeted than to be universally believable.

In a Twitter study from Haewoon et al [TWNEWS10] researchers found that any retweeted tweet on average is to reach 1,000 users, no matter what the number of followers is of the original tweet. This is an interesting finding, as it gives each individual user the power to spread information (or disinformation) broadly.

Interesting example of disinformation spreading in Twitter may be found in Swine Flu (H1N1) outbreak in 2009. In the period from April 20th to April 24th the percentage of tweets referring H1N1 rose from nothing to 0.2% of all messages in circulation. On April 25th rough 2% of all tweets were related to Swine Flu. A significant part of this traffic consisted of misinformation related to H1N1 transmission vectors, about the actual spread of the outbreak and the speculation of its source [SFLU09].

While there is no evidence to support the claim that some or all of that misinformation have been seeded purposely, based on previous analysis it can be understood that, for the external parties, it would not be hard to affect the information flow.

How it may be easy to communicate disinformation on Twitter based on false identity may be observed from the example of Janet, a Twitter user pretending to be Exxon employee [JANET08]. In August 2008 “Janet” took part in Twitter conversations pretending to be an Exxon’s member of staff. She has been answering questions and giving explanations, like explaining the direction of the company or where philanthropy resources are being spent.

It took three days since the first tweet in the name of Exxon Mobil Corp. appeared to draw the attention of mainstream internet media and analysts. At that time, it was intriguing for general public to meet a company engaging with its customers via Twitter channel.

From this issue, it can be observed that lack of identity confirmation and deficiency or unreliability of the options for the community to confirm identity may present one of the obstacles in establishing communication based on trust. Also, unaware users are at higher risk from being misled by information which originates from unconfirmed or untrusty source or which reference is missing.

3 Software Vulnerabilities

Errors are an inevitable companion of the software development process, or generally of any development process which is complex, layered, distributed, domain-crossing and which involves a number of subjects with different backgrounds, goals and interests.

As the software gets increasingly complex and deeper integrated into business processes and everyday life, the significance and possible dangers of the impacts resulting from the errors in the complete process of the software development become more and more accented.

This chapter introduces the reader with the software vulnerability model, related terms and problems. It presents some of the obstacles in the process of defining and distinguishing the concept. The taxonomy of the software vulnerabilities is the particular topic of importance for this work, therefore, the brief overview, the history and developments of this field are provided.

Lastly, the reader is introduced with the initiatives from Mitre Corporation defining the systems for Common Vulnerabilities and Exposures (CVE) and Common Weaknesses Enumeration (CWE), the products of which are further applied in this work.

3.1 Definition

ISO/IEC 27005:2008 defines the vulnerability as *a weakness in an asset or group of assets. An asset's weakness could allow it to be exploited and harmed by one or more treats*. The same standard treats asset as *any tangible or intangible thing that has value to an organization*. Threat is further considered as a potential event which may turn into an actual event, causing an unwanted incident with additional potential to harm an organization or system [ISO27005].

This standard identifies vulnerabilities in one of the following areas:

- Organization
- Processes and procedures
- Management routines
- Personnel
- Physical environment
- Information system configuration
- Hardware, software or communications equipment
- Dependence on external parties

ISO's definition is broad in the scope, considering the existence of flaws in the environment, working systems or procedures extrinsic to the software system. However, the scope of investigation of software vulnerabilities requires more refined and precise definition.

The National Institute of Standards and Technology (NIST) gives the following general definition which may be easier applied on the software system. The publication from NIST named *SP 800-30* as a vulnerability identifies *a flaw or weakness in system security procedures, design, implementation or internal controls that could be exercised (accidentally or intentionally) and result in security breach or a violation of the system's security policy* [NISTSP02]. The significance of this definition in the scope of software systems is in the fact that it identifies the potential of vulnerabilities to be a result of a flaw in different software development processes or cycles, like design or implementation.

Although they have been extensively covered in the literature, there is no widely accepted definition of software vulnerabilities. One of the contributions towards a better understanding of the field has been introduced in the dissertation of Ivan V. Krsul. In his work Krsul analyzed the emergence of software vulnerabilities and identified two fundamental underlying concepts. The importance of his work is in the fact that he, similarly to NIST's definition, derives different areas of software development in which software vulnerability may originate: specification, development or configuration [KRSUL98].

The first concept identified by Krsul relates to terms *error*, *fault* and *failure* [KRSUL98]. They reflect software vulnerability from its origin, or cause, to its manifestation and induced consequences.

IEEE Standard Glossary of Software Engineering Terminology gives a more refined definition of the term *error* and connects it to the term *mistake*. One of possible distinctions between them considers *mistake* as an event of human action that produces incorrect results. That could be, for example, an incorrect action of a computer programmer or operator. Furthermore, the same source considers the *error* as *the difference between a computed, observed or measured value or condition and the true, specified or theoretically correct value or condition* [IEEE90]. In this sense, the *fault* is further identified as a manifestation of an initial mistake, whereas the *failure* corresponds to the result of the fault and symbolizes system's inability to perform its functions.

The second concept identified in the work of Ivan V. Krsul is *computer policy*. Software vulnerabilities violate or provide the mechanism for disobeying the rules and practices defined under computer policy. Krsul used the work from Garfinkel and Spafford [PRACUNIX96] as a starting point in the process of defining a computer policy. They state that the policy *is used to define what is considered valuable and specifies what steps should be taken to safeguard those assets*.

The additional term, *security policy*, is often related to the computer policy either as the same concept or its refinement. Landwehr mentions the security policy as a set of rules defined to meet particular goals [LANDWEHR01]. These goals may be classified under different categories, where the most prominent of them include confidentiality, integrity, availability, accountability, authentication and non-repudiation [LANDWEHR01, LAMPSON04].

For further considerations it should be noted that the term vulnerability has a wider scope than software vulnerability, as shown in the previous definitions [ISO27005, NISTSP02]. However, as this work focuses on the software vulnerabilities alone, the terms vulnerability and software vulnerability are equated here for the practical purposes. The same applies to the flaw, which in the scope of this work corresponds to the software flaw only.

Now that the basic underlying concepts are presented, the term software vulnerability may be further elaborated. The Organization of Internet Safety (OIS) formulated the common definition of this term:

Security vulnerability is a flaw within a software product that can cause it to work contrary to its documented design and can be exploited to cause the system to violate its documented security policy. [OIS04]

Information Assurance Analysis Center (IATAC) provides more comprehensive definition:

A vulnerability is an attribute or characteristic of a component that can be exploited by either an external or internal agent (hacker or malicious insider) to violate a security policy of (narrow definition) or cause a deleterious result in (broad definition) either the component itself, and/or the system or infrastructure of which it is a part [IATAC11].

OIS's definition expects that *software flaw* is a necessary condition for software vulnerability to exist.

The important difference between flaw and vulnerability is determined by the level of exploitability of the flaw. Flaw stands for a potential vulnerability which may not be manifested for a number of reasons. Consequently, the software vulnerabilities form a subset of software flaws conforming to conditions or expectations in the real-world exploitability. For instance, software flaw may be available in the section of the program which is not easily reachable by a potential attacker. Additionally, it may require certain privileges or settings related to its environment, users or other connected systems to become exploitable.

Culp from Microsoft formulates the prerequisite for the flaw to be considered as vulnerability. It is in essence infeasible to prevent violation of the security policy even when the product containing vulnerability is properly used [CULP00]. For instance, web application may contain software flaw which is exposed to visitors with administrative privileges. As in normal operation administrative privileges are given to the trusted users only and standard users possess restricted privileges, that flaw should not be considered as vulnerability.

One captivating view on software vulnerabilities is presented by Engle et al [TREE06]. They consider vulnerability as a set of *state transitions* which take the system from the set of allowed states to disallowed state. Disallowed state is known as a *vulnerable state*. It has a set of attributes purposely identified as characteristics. The task of security policy is, therefore, to separate system states into allowed and disallowed ones. This proposal has been further extended and examined in [VCMOD08].

The definitions and views shown above indicate ambiguity and potential obstacles in the process of gaining a common understanding and definition of the term software vulnerability. However, in the scope of this work, these ambiguities are of a less relevance. Therefore, they will not be discussed in detail. Rather, their presence will later show how, depending on the view, the different classifications schemes can be developed and how the view on software vulnerability may influence further process of its investigation and analysis.

3.2 Software Vulnerability Taxonomies

Many subjects take part in the collection, analysis and dissemination of Information about software vulnerabilities, including the following:

- Various government agencies and public bodies
- Privately-owned or independent research and academic institutions
- Commercial software vendors
- Commercial agencies and other entities specialized in software vulnerabilities
- Independent individuals, including black and white hats²⁶

Each of these subjects has its own motivation and purpose for working on security vulnerabilities. They also have different views on vulnerabilities, their origin, description, classification, fixes, potential impacts etc. Their classifications may differ or overlap not only in the descriptions and related perception of vulnerability and its consequences, but also in the metadata structure or in the procedures used for the information retrieval, processing and exchange.

In order to make the organization, exchange and analysis of security related information feasible and comparable between those subjects, and further to facilitate the public utility of the retrieved information, some common mechanisms and technologies with unified and harmonized structure should be used. They could include classification systems for

²⁶ White hat is used as a description of experts or hackers who identify a security weakness, but do not take malicious advantages of its discovery. Contrary, black hats are individuals using vulnerability for malicious purposes.

vulnerabilities, but also the procedures and guidelines for the vulnerability assessment, as well as appropriate formats for storage and exchange of information.

In the last decades many authors approached the topic of software vulnerability classification. Many of their proposals are in the form of taxonomies. The taxonomy corresponds to a subject-based classification that arranges the terms in the controlled vocabulary into a hierarchy. It allows related terms to be coupled together and categorized in ways that make it easier to select the appropriate term to use, whether for the searching or to describe an object [ONT04].

In the software security domain, a taxonomy-like organization allows vulnerabilities to be uniquely identified. One of the first relevant works related to software vulnerability taxonomies appeared in 1970s. At that time, Abbott et al focused on operating systems, developing taxonomy based on seven root classes [RISOS76]. These classes are depicted in Figure 3.

Later, Landwehr et al proposed a taxonomy which classifies each flaw according to its *genesis*, *time of introduction* and *location* in the vulnerable system [NRL93]. The rationale for such effort has been found in the observation that the history of computer failures is relatively undocumented.

Their intention was to help system designers and security analysts in the process of increasing level of the understanding and overview of security flaws. They argued that such classification could provide a detailed insight into the life cycle of software failures. That way, the user could benefit from the information like which parts of the system, what time points and which phases in the software development process introduced what kind of flaws.

Three levels of the taxonomy by Landwehr et al can be summarized as follows:

First level: the *genesis*. The flaws can be produced intentionally or unintentionally. Intentional ones can be a result of malicious or non-malicious motivation. The group of malicious flaws includes Trojan horses, trapdoors and time bombs, while non-malicious group consists of covert channels and others.

From the other side, the inadvertent group of the flaws has six categories:

- Validation errors
- Domain errors
- Serialization and aliasing errors

- ❑ Errors due to inadequate identification/authentication
- ❑ Violation of boundary conditions
- ❑ Other logic errors

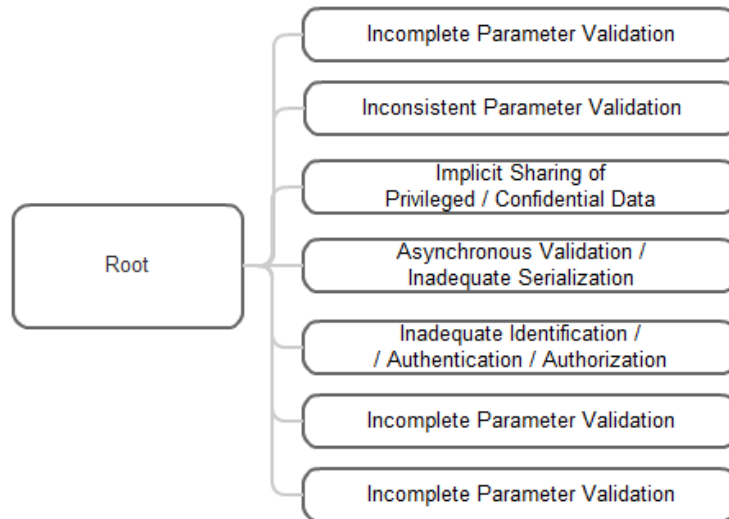


Figure 3: Taxonomy of software vulnerabilities in operating systems [RISOS76]

The second level in this taxonomy is determined by *the time of introduction* of flaws in the software life cycle. These include software development phases such as development (requirements - specifications - design, source or object code), maintenance or operation.

In the last, the third level of the taxonomy, the location of the software flaws determines the classification of the software flaws. As the possible source of the flaw, this classification includes not only the software, but also the hardware. Branch devoted to the software as the source place of the flaw contains subcategories related to the flaws occurring in the operating system components and support software.

Landwehr et al identified the following components of the operating system as potential flaw locations:

- ❑ System initialization
- ❑ Memory management
- ❑ Process management/scheduling

- ❑ Device management, including I/O and networking
- ❑ File management
- ❑ Identification/Authentication

The support software flaws branch further in the privileged and unprivileged utilities.

The taxonomy presented by Landwehr et al. is not only theoretical construct – in appendix authors provided 50 example vulnerabilities which have been used during the construction of the taxonomy.

Howard and Longstaff pointed out that this taxonomy do not address some of the issues previous taxonomies had, like the ambiguity of classification procedures or the presence of several “other” categories, which implies the incompleteness of the taxonomy [HL98]. Aslam noticed that Landwehr’s taxonomy can be difficult to implement in reality, because it requires access to the source code of the software and additional information related to the programming environment.

In addition, the decision about a time of flaw introduction in the software system requires additional knowledge like the project documentation or progress reports. These are difficult to obtain publicly and require an additional effort to be examined [ASLAM95].

In his thesis [ASLAM95], Aslam proposed a domain-specific taxonomy for vulnerabilities in UNIX operating systems, with the primary motivation to develop a system which supports the unambiguous classification of the flaws into distinct categories.

Aslam collected security flaws from different sources, including CERT advisories, flaws published on mailing lists and literature survey. His approach was a bit different, as he focused on analysis of software faults. In order to address issues found in other proposals, like ambiguity resulting in the categorization of flaws in more than one category, he specified selection criteria for each fault in category.

The main categories proposed in his work [ASLAM95] are based around the following faults:

- ❑ coding faults
- ❑ operation faults
- ❑ environment faults

However, some authors²⁷ deemed that approach as still incomplete, failing to define the classification schema that could discover unique categories of vulnerabilities.

Instead, they proposed the solution based on a different viewpoint. Instead of the hierarchical taxonomy, their proposal uses attribute-value pairs to provide a multidimensional view of vulnerabilities. Those pairs are present in the form where an *object* has an *attribute* with a *value*. Once object has an attribute with a defined value, it becomes a *property* of the object. In this system, as an addition to vulnerabilities and exploits, they identify *mitigations*²⁸, which also stand in relationships with vulnerabilities through the attributes [CMSEI05].

In the domain of the software vulnerabilities, possible object-attribute combinations should be limited only to the relevant ones. The object's attributes correspond to the security flaws that may or may not lead to the vulnerability of a system. The Figure 4: Example exploit [CMSEI05] shows Unified Modeling Language (UML) activity diagram of the exemplar exploit. The program from this example illustrates contains several software flaws. Each activity described is tagged with an attribute-value pair (dashed boxes) that represents required preconditions for the exploit to succeed. Additional descriptive attribute of the program is its usage of a memory manager that uses boundary tags, such as `dmalloc`²⁹ [CMSEI05].

The significance of this approach is in the fact that it offers the universal framework, which usage can be based on different aspects. It can enable users of different roles to take the advantage of the common description and classification system, each in their own domain. Attributed code segments can be automatically analyzed against available classifications and descriptions. This could lead to the discovery of the information about their vulnerability to the known exploit classes. It enables and eases the process of automatic assessment of threats posed by vulnerabilities and mitigations techniques or strategies available for them.

Furthermore, the solution proposed forms the base for a better understanding of characteristics of the vulnerabilities and their correlation with the incidents, exploits and artifacts. This could lead to the development of the predictive model that can anticipate threats with a high level of significance automatically.

²⁷ Including Seacord and Householder

²⁸ Mitigation is referred as a set of methods, techniques, processes, tools or runtime libraries that are able to prevent or limit exploits against vulnerabilities. Alternatively, they are referred as countermeasures or avoidance strategies. [CMSEI05]

²⁹ Doug Lea's implementation of standard C routine `malloc`

Based on the previous research presented in this section, it can be noticed that different approaches have been applied to the problem of creation of the taxonomy of software vulnerabilities. In his overview, Krsul discussed 17 approaches to the taxonomy and classification. In his thesis, he disputed some of the taxonomies known in the literature, including ones mentioned in this work [NRL93, ASLAM95]. He argued that they even do not satisfy necessary conditions to be considered as taxonomies [KRSUL98].

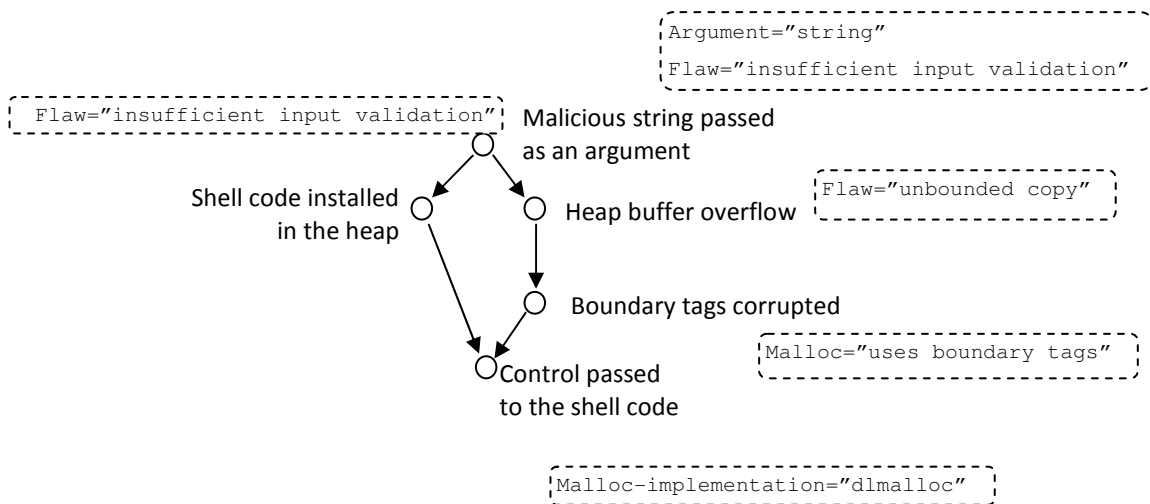


Figure 4: Example exploit [CMSEI05]

In order to develop the guidelines for their proposal for the software vulnerability taxonomy, Howard and Longstaff compiled a list of characteristics of satisfactory taxonomies. According to them, the classification categories should follow the following guidelines [HL98]:

- Mutually exclusive: the categories should not overlap
- Exhaustive: the categories should include all possibilities
- Unambiguous: clear and precise, classification should be certain and unambiguous
- Repeatable: repeated applications lead to the same classification, regardless of the classifying person
- Accepted: categories should be logical and intuitive in order to facilitate a general approval
- Useful: could be used to gain the insight in the field of inquiry

With this proposal they tried to summarize the results and describe the insight gained from the previous work in the field. Although some authors noticed and further adopted their suggestion (e.g. [SVCONS05, SITANAL05, THONO10]), others have criticized their work as not particularly useful in the practice [OZMENT07].

In his thesis, Ozment ranges taxonomies from the most comprehensive to less formal ones, noticing that most of the existing taxonomies tend to be on the less formal, weaker side of the scale, being useful in the practice mostly for a narrow analysis or a task [OZMENT07].

The peers in the field reached the relatively common observation about the fact that existing taxonomies do not provide scheme which can be widely used in the practice as a general and universal tool. Instead, they observed that the simple classification and enumeration schemes are used frequently in the practice.

3.3 Initiatives of Mitre Corporation

A decade ago there has not been a commonly adopted naming convention and classification for defining and describing security problems in the software. Instead, many subjects used or tried to promote their schematics and methods for identification and notation of software security problems. That situation resulted in instable, incompatible and incoherent naming schemes and approaches.

The implementation of the processes from the domains of information exchange and cooperation has been embarrassed and complicated for vendors, users, security analysts and other parties involved in the tracking and analyzing of software vulnerabilities.

In January 1999, at the 2nd Workshop on Research with Security Vulnerability Databases at Purdue University, MITRE Corporation presented an approach and proposal for a common enumeration of software vulnerabilities. In beginning conceptualized as a simple mechanism for linking vulnerability-related databases, through the time CVE (Common Vulnerabilities and Exposures) progressed toward the most comprehensive list of software security vulnerabilities available.

Initial motivation of the CVE initiative was to facilitate adoption of the CVE list as a common mechanism for referring to vulnerabilities and exposures.

This initiative targeted five main areas of activity:

- Unique naming of every publicly known security vulnerability and exposure
- Injection of CVE names into security and vendor advisories
- Establishment of CVE usage as a common practice in computer security area
- Promoting CVE usage through public policy guidelines, methodology requirements, requirements for new capabilities, training, usage and best practice suggestions
- Motivating commercial suppliers to use CVE names in their update and fixing mechanisms

Today, CVE is widely adopted, gradually approaching the goal of uniquely naming every publicly known security relevant software problem. Its names are normally included in many security relevant bulletins and sources, including CERT/CC, Microsoft, Red Hat and 73 other organizations. Additionally, there are more than 100 organizations participating worldwide in the process of making their products, services and offerings CVE compatible³⁰.

CVE is extensively applied in the fields of vulnerability management, patch management, vulnerability alerting and intrusion detection. It is integrated in the US-CERT (United States Computer Emergency Readiness Team) bulletins, used by SANS Institute and approved by NVD (National Vulnerability Database). NVD is the United States government repository of standards based vulnerability data, which is completely based upon the CVE and represents its refined superset. NVD provides vulnerability data feeds implemented according to the CVE standard, augmented with the additional analysis information and search database.

The main concept the CVE list is based upon is represented through the CVE Identifier³¹. It is unique, common identifier for publicly known information security vulnerabilities. In order to be accepted in the CVE list and therefore recognized as a CVE *entry*, each potential security vulnerability or exposure is initially assigned a *candidate* status and appropriate candidate number. In the further process, the CVE Editorial Board decides whether it should be designated official CVE entry status. If the candidate is accepted, its status is updated to *entry* on the CVE list. CVE list is organized and maintained by the CVE Editorial Board.

³⁰ As of September 2011, based on report from <http://cve.mitre.org/compatible/index.html>

³¹ Also called CVE name, CVE number, CVE-ID or just CVE

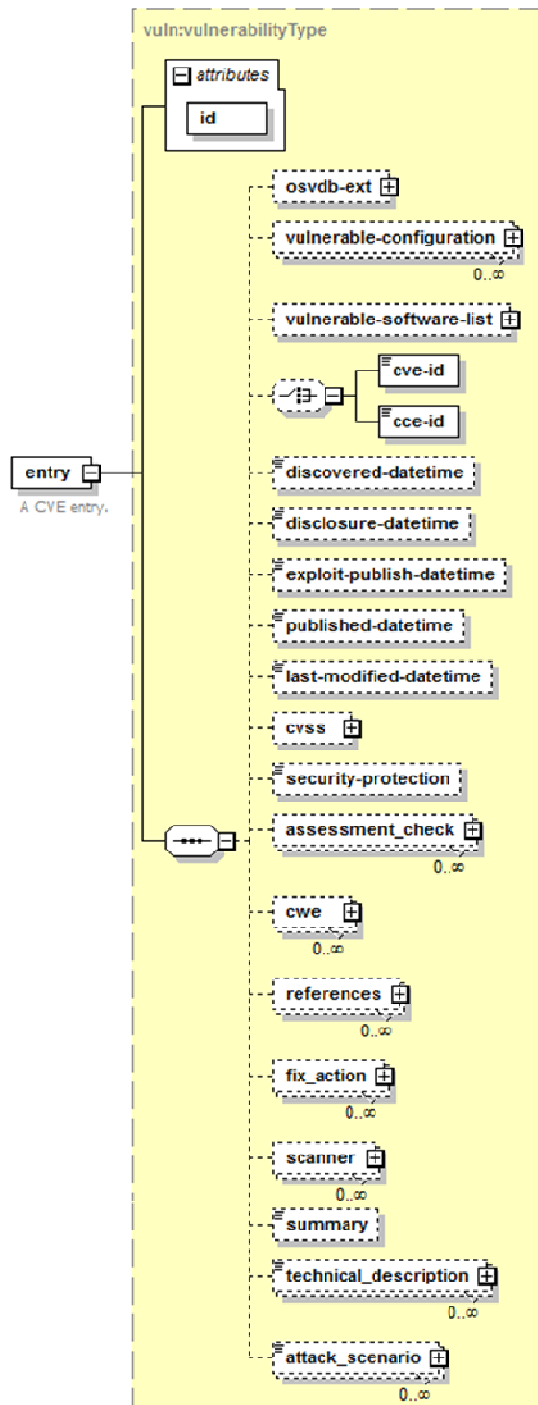


Figure 5: CVE entry schema definition (version 2.0)

Its members are, among the others, the security-related organizations including commercial security tool vendors, members of academia, research institutions, government agencies and other subjects. The Mitre Corporation moderates and provides guidance in the complete

process [CVEED11]. Figure 5 provides the general overview of the CVE data provided by NVD (top level, part related to vulnerability entry).

From this diagram, it can be noticed that the most of the elements are optional. In practice, their inclusion depends on the several factors. Usually, as the CVE list includes vulnerabilities from many different sources, the reports from those sources may vary in extent. In some cases, the evaluation of some aspects of vulnerability (or exposure) may not be applicable for particular vulnerability, or, the subject reporting vulnerability may not execute complete set of evaluations.

There may be other reasons why some entries do not expose all elements available by the schema definition. However, the academic and commercial community is working on improvements and extensions of the CVE list and related resources constantly. They are also working on the further promotion of the standards and training. As the consequence, the completeness and the scope of the vulnerability information submitted is increasing with the time.

The elements illustrated in Figure 5 carry in the most cases self-explanatory names. According to the scope of this work, the most relevant fields from this schema and related standards will be discussed bellow. Figure 6 describes the elements *vulnerable-configuration* and *vulnerable-software-list* from Figure 5. These elements refer to the values from the CPE list (Common Platform Enumeration), helping us to identify the name of the product affected.

CPE is another initiative of Mitre Corporation, with the goal to establish a naming scheme for information technology systems, platforms and packages. In the context of CVE, CPE is used as a source of information about products affected by the vulnerabilities. The more details about CPE are provided in Chapter 4.

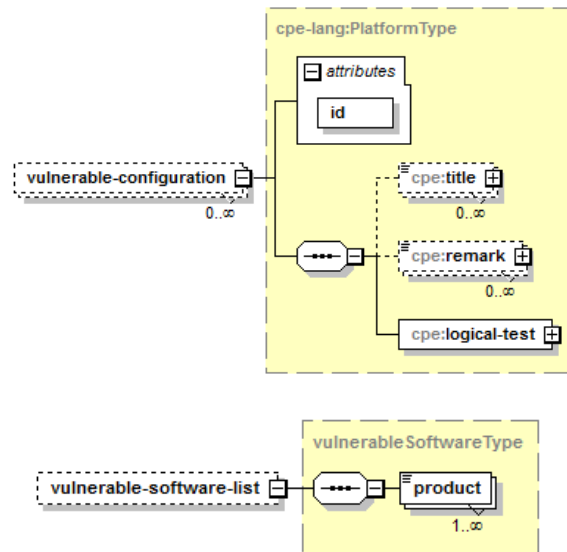


Figure 6: Definitions of elements in CVE schema (version 2.0)

The element from Figure 7, *cvssImpactType*, refers to the CVSS (Common Vulnerability Scoring System), which role is to describe the characteristics and impacts of the vulnerability through scores. CVSS is developed by the CVSS Special Interest Group (CVSS-SIG) under supervision of Forum for Incident Response and Security Teams (FIRST). Its actual version is 2.0, published in 2007.

From the diagram presented in the Figure 7 (bottom part), it can be observed that among the three groups, the *base_metrics* is the only compulsory element to be provided.

Three metric groups describe the following [CVSS07]:

- **Base:** describes the intrinsic and base characteristics of a vulnerability that do not change over the time
- **Temporal:** describes the characteristics of a vulnerability that change over the time but not among user environments
- **Environmental:** describes the vulnerability from the aspect of the relevancy of particular user's environment

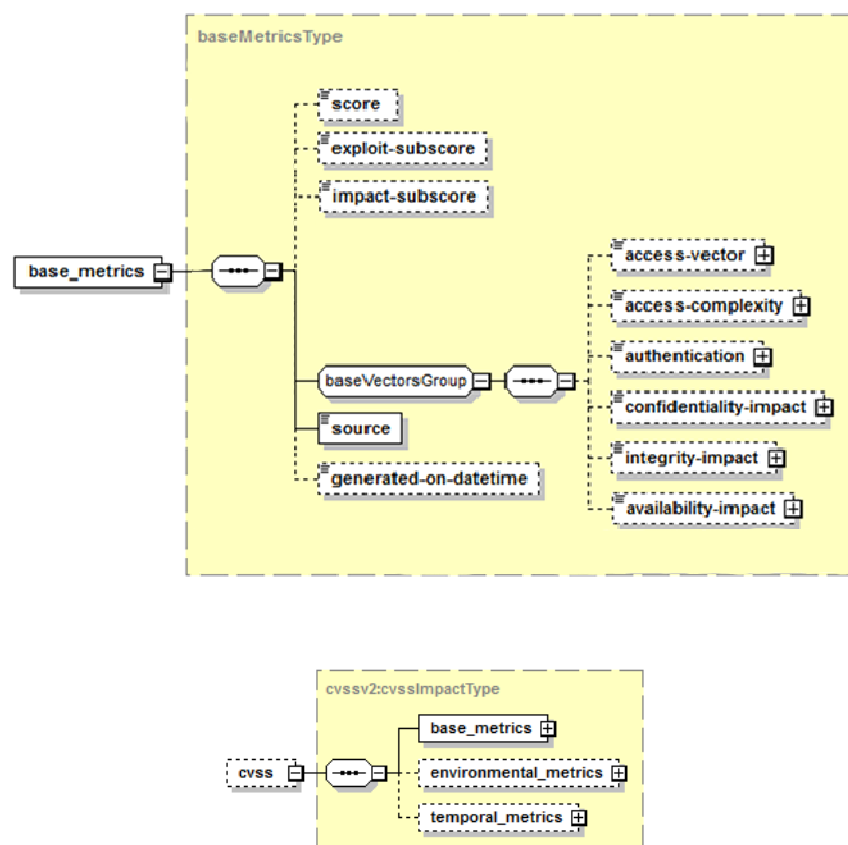


Figure 7: Expanded element *baseMetricsType* from the CVSS scheme

The base group is further expanded and illustrated in the upper part of the Figure 7.

The base metrics group builds upon the cumulative severity score³². The bases for its calculation are the subscore sets related to the following metrics: *Access Vector*, *Access Complexity*, *Authentication*, *Confidentiality Impact*, *Integrity Impact* and *Availability Impact*.

After the base group metrics are designated, the cumulative score is calculated in the range from 0 to 10. The scoring process produces the vector in the form of text string that contains the values assigned to each metric and that should be always shown together with the score. The base and other equations for the calculation of the score are defined by CVSS Guide [CVSS07]. The calculation of temporal and environmental metrics is not required, but may be included if desired.

³² Calculation is done based on CVSS criteria

The next important information CVE entry provides is the relation to CWE identifier (Common Weakness Enumeration). This relation points to the particular CWE identifier, which then provides the information about the classification and type of the software vulnerability.

CWE is a result of community initiative, targeted to communities of software developers and software security analysts. It is a formal list of common software weaknesses that can occur in the software architecture, design, code or implementation, which can lead to exploitable security vulnerabilities. It has been designed with two objectives on the mind. The first is to help shaping and maturing of the code security industry. The second one is to strengthen software assurance capabilities of the organizations involved in the reviewing process of the software systems, both in the cases of software acquisition or development [CWE07].

CWE organizes weaknesses³³ in a hierarchical structure, where each weakness is given a unique identifier (CWE-ID). The hierarchical structure follows the different abstraction levels of classes, where the higher level represents more abstract, and the deeper level more concrete concept of weakness. The elements in this system are organized in four main types: Category, Compound Element, View and Weakness, each designated with a unique id [CWEFAQ11].

In its version 2.0, the CWE list contains 886 entries, defined through 693 weaknesses, 157 views, 27 categories and 9 compound elements. The category in CWE list is defined through a collection of weaknesses sharing a common attributes. These attributes may be based on a number of concepts, including environment (like J2EE, .NET), functional area (like authorization, encryption) or resources (like credentials management or certificate issues) [CWEFAQ11].

The compound elements in the CWE categorization are built from meaningful aggregations of several weaknesses (composite or chain). Example for such element may be CWE-692, which describes blacklist based protection mechanism intended to defend against XSS attacks, which fails due to the incompleteness of the related blacklist.

Views in the CWE list represent predefined perspectives used to look at the CWE weaknesses. One example for such approach is *CWE-700: Seven Pernicious Kingdoms*, which organizes weaknesses using hierarchical structure that is similar to taxonomy presented by Tsipenyuk et al [SKING05].

³³ A term *weakness* in this case can be referred to *software flaw*, as explained in the chapter 3.1. Therefore, it represents a *potential vulnerability*.

Finally, the weaknesses may be classified in three sub-classifications: *class*, *base* and a *variant*. *Class* is the most abstract type of weakness, like CWE-311, which describes failure to encrypt sensitive data. *Base* is a more specific type of weakness that is mostly independent of a specific resource or technology. The example of it is CWE-364 (signal handler that introduces a race condition). *Variant* is a weakness specific to a particular resource, technology or context. Examples for a variant may be CWE-107 (unused validation form in Struts) or CWE-563 (unused variable).

The information contained in the CWE entry includes the following [CVEFAQ07]:

- CWE identifier number and name of the weakness type
- Description of the type
- Alternate terms for the weakness
- Description of the behavior of the weakness
- Description of the exploit of the weakness
- Likelihood of exploit of the weakness
- Description of the consequences of the exploit
- Potential mitigations
- Node relationship information
- Source taxonomies
- Code samples for the languages or architectures
- CVE identifier numbers of related vulnerabilities
- References

The other information provided in the CWE entry for the type weakness is illustrated in the Appendix 2. Due to the space considerations and the relevance, illustrations of other types of definitions are not included in this document. The Appendix 1 provides the example of CWE entry. This example contains the subset of information defined by the CWE vocabulary. It is presented in the format suitable for the paper-based representation, reflecting the information available and not its complex underlying structure and organization.

4 Software Framework

This chapter deals with the software framework prepared as a part of implementation phase of this thesis. It describes the framework and provides information about its main components, features, and related technology. The second part of this chapter describes the external data sources and explains how they are used in the framework. Finally, the third section presents the processing flow done with the software framework.

4.1 Rationale behind the Idea

The number of the users of social networks increases rapidly. Although the platforms of some of the most popular social networks are based on sharing of personal information and general entertainment, the potential of the networks is increasingly being recognized in the other areas too. The important fact about online social networks is that they facilitate and simplify the information sharing process, what surely contributes to their popularity.

It is not only that they encourage the sharing of information, but their approach also includes other prerequisites for successful knowledge sharing. For instance, the practical inclusion of three solutions to the knowledge sharing dilemma [CABRERA2002] plays important role in the willingness of the users to share the information. That readiness builds an important block in the foundation of the global information space. Today, it eases and facilitates access to the information which was expensive or unavailable before. This creates new ideas and opportunities in the research fields.

The idea behind this work is to gather the software vulnerability related data in the Twitter information stream, and then to analyze it, asses its potential and determine its properties and trends involved. For such purpose, the practical software framework was implemented.

The first task of the software framework is to gather relevant information from Twitter and other external data sources, convert it to an appropriate representation and prepare for further processing.

The processing of that information is the next task of the framework. In this task the information on Twitter, basically the short messages broadcasted by the users, is processed and augmented with the information from the other data sources. These data sources are related to the general software and software vulnerability domain, with the possibility to add information from the other domains or sources, if necessary.

The third important task of the software framework is to enable the export of the preprocessed and augmented information both in the form of visual diagrams, or in the other format appropriate for analysis in the other tools. The general description of the software framework, data sources and the processing flow is further disseminated and presented in the next sections.

4.2 Framework Description

The greatest part of the software implementation used in this work is based on Java 6 compatible code, developed using SpringSource Tool Suite IDE. Its graphic description is provided in Figure 8.

The framework itself consists of several loosely coupled components named *TweetCatcher*, *FeatureCatcher* and *Presenter*. Furthermore, the framework depends on several external tools, API services and data. On Figure 8 they are represented in three layers. The first layer consists of the software libraries used to extend or enrich functionality of the core. The most important of them are *Twitter4j*, *Jena*, *LingPipe* and *HighCharts*, depicted in the figure in the second layer.

The following layer displays the external sources of data. The first source is related to the services provided as an application interface and remote endpoint for queries based on different technologies, like REST or SPARQL. *Twitter Streaming API*, *DBPedia* and *FreshMeat* are providing this type of access to their data.

The second sources of the data are CVE, OSVDB, CWE and CPE. These are sources providing their data publicly, but in the form of downloadable file which is periodically refreshed with new records.

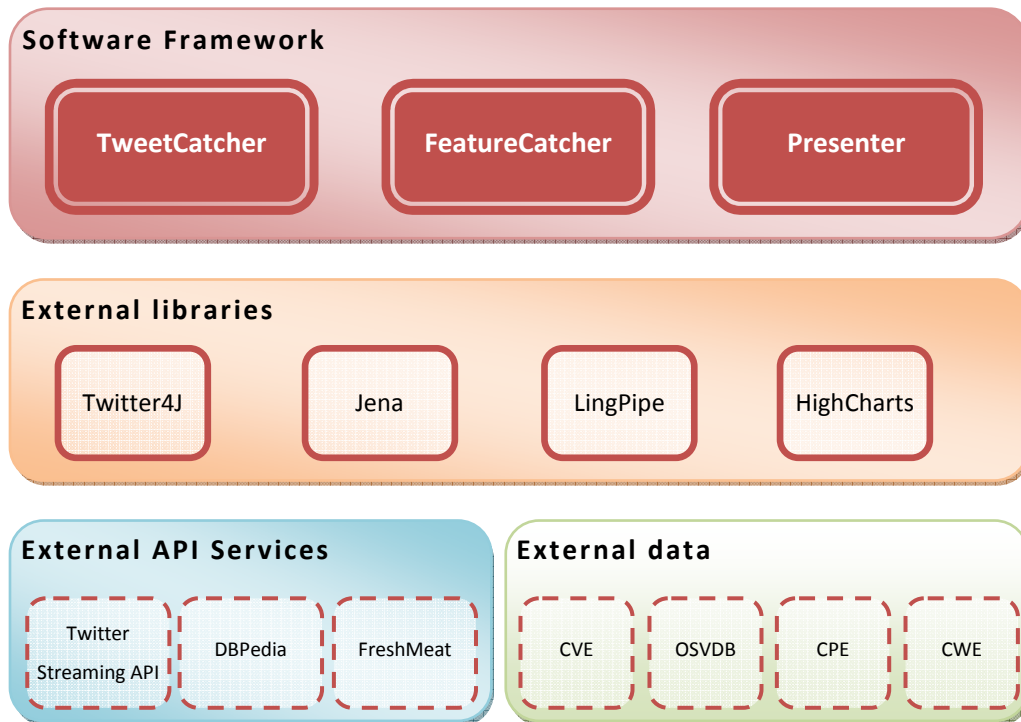


Figure 8: Graphical overview of the software framework

The rest of the current section focuses on the brief descriptions of the most important external libraries, services or data sources used in this work, as shown in the Figure 8. The following sections provide detailed descriptions of the components depicted in figure.

Twitter4J is an unofficial Java library for the Twitter API in general. It provides extensive coverage of Twitter API functionality through pure Java code available on any Java platform version 1.4.2 or later. Its support for Twitter API is present in the following areas:

- Timeline Resources
- Tweets Resources
- User Resources
- Trends Resources
- Local Trends Resources
- List Resources
- List Members Resources
- List Subscriber Resources
- Friendship Resources
- Friends and Followers Resources
- Account Resources
- Favorite Resources
- Notification Resources
- Block Resources
- Spam Reporting Resources
- Saved Searches Resources
- Geo Resources
- Legal Resources
- Help Resource
- Streamed Tweets Resource
- Search API Resource

Twitter4J provides a thread-safe interface to Twitter Streaming API. In this work, *TwitterCatcher* component uses Twitter4J to access Twitter Streaming API and its services.

Jena is a semantic web framework for Java based on Semantic Web Recommendations. It provides API for working with RDF, RDFS and OWL³⁴. Additionally, it supports SPARQL³⁵ and includes a rule-based inference engine.

In the context of this thesis, Jena is mostly used as a tool to represent and store resources. An overview of the Jena architecture is depicted in Figure 9.

Jena's I/O subsystem supports several stores, including in-memory store, SDB (SQL Data Base) and TDB (Tuple Data Base). SDB is a SQL-backed storage subsystem, designed specifically to support SPARQL. Its current version supports at least 8 popular RDBMS³⁶, including Microsoft SQL Server 2005, Oracle 10gR2, MySQL 5 and PostgreSQL 8. TDB is a native Java store with the focus on scalability and support for large stores. In a typical usage scenario it also tends to be faster than SDB.

In Jena, inference over several models can be provided automatically, thus making that process transparent for the user. Importing and exporting data is possible using RDF/XML, N-triple, N3 and Turtle³⁷ languages.

In the second level depicted on the Figure 9, Jena provides ontology and reasoning capabilities. Ontologies can be loaded from URI's and applied to existing models to infer new knowledge based on rules defined by the ontology model.

ARQ is a query engine for Jena which supports the SPARQL RDF query language. It aims to SPARQL conformity and provides support for TDB and SDB storage engines in Jena. The ARQ's capabilities are further extended by Joseki, which is a Jena's HTTP engine with integrated support for SPARQL protocol and HTTP (GET and POST) operations.

³⁴ RDF is a standard model for data interchange on the Web. RDFS is used to describe structure of the data (model), while OWL describes semantic relationships. Complete explanations and recommendations of these technologies can be found on W3C web site.

³⁵ SPARQL is a query language for RDF designed by the W3C RDF Data Access Working Group.

³⁶ Relational Database Management System

³⁷ Non-XML based serialization

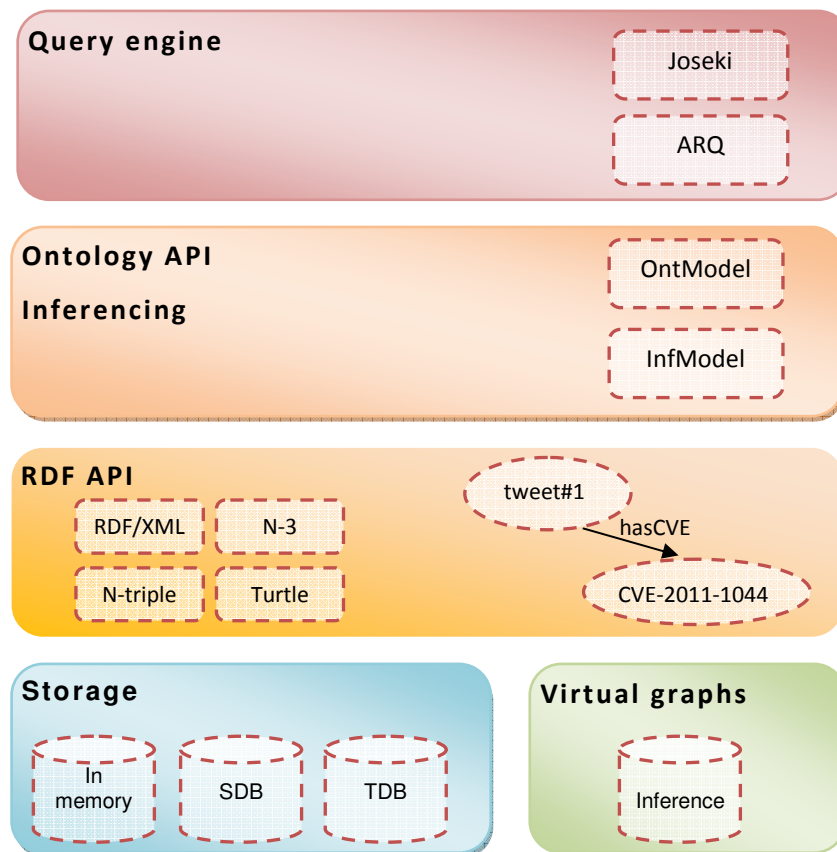


Figure 9: Jena framework overview

Twitter Streaming API is a service that allows a high-throughput and near real-time access to public and protected Twitter data. It represents a successor of the Twitter REST API, addressing some of its issues related to the latency, complexity and general costly usage.

The development of the Twitter Streaming API has been motivated with the following goals and improvements [JKALUC10]:

- the delivery of other event types as well as delivery of the messages has to be provided
- the messages (and other events) should be delivered with the lower latency
- the system should distribute full range of data, accurately, without limitations or improvisations based on rate limiting, parallel fetching of messages and similar
- large scale integrations with other services should be as easy and flexible as possible
- the support for easier integration for the developers should be provided

Twitter Streaming API consists of three main services. They target different usage scenarios and user segments. The following snippet provides a brief description of these:

- Streaming API

This service provides public statuses from all users. The statuses can be provided based on filtering criteria only. For instance, the filtering can be applied based on keyword, specific usernames or locations.

- User Streams

This service provides nearly all data necessary to update user's display. The service can be initiated through REST API and then transitioned to Streaming for subsequent reads. The service is used for a small number of user accounts – typical scenario is a Twitter client application.

- Site Streams

This service provides multiplexing of multiple User Streams. For this type of service prerequisite is received authorization by the users. Primary use case may be a web site providing service integration for many users.

Twitter Streaming API is used in this work to receive the messages from Twitter users. The messages have been filtered by keyword criteria. The next section provides the details about the keywords used.

LingPipe is a toolkit for text processing using computational linguistics. It is designed to be efficient, scalable, reusable and robust. It is thread-safe, character-encoding sensitive and supports online and customized training. Additionally, LingPipe allows usage of multi-lingual, multi-domain and multi-genre models [LP11].

The LingPipe framework provides interfaces to many application scenarios, including topic classification, named entity recognition, part-of-speech tagging, sentence detection, detection of interesting phrases and new terms as well as word sense disambiguation.

HighCharts is a charting library written in pure JavaScript. It offers an easy way to integrate the interactive charts in web applications. The wide range of charts is supported, including line, spline, area, areaspline, column, bar, pie and scatter charts [HC11].

Depending on user needs, the charts can be customized, providing additional functionality such as multiple axes and axis supporting display of dates and times. The zooming of data is possible, both on X and Y axis. Additionally, a chart can display a tooltip text on hovering, containing the

information on each data point or series. The text labels can be rotated in any direction. The users can print or export charts to several popular image formats, as well as PDF and SVG files. The data can be loaded internally, from the code, or from the external service, using callback functions [HC11].

In this work HighCharts is used as a tool for visualization of trends on Twitter.

DBPedia is a wide community effort to extract structured information from Wikipedia and make it available to the public. DBPedia strives to be a knowledge base covering many domains, making knowledge accessible under the terms of the Creative Commons Attribution-ShareAlike 3.0 Licence and GNU Free Documentation Licence [CCS11, GNUFD11].

As of the time of this writing, DBpedia describes more than 3.64 million things, including 416,000 persons, 526,000 places, 169,000 organizations and many other entities. It automatically evolves as Wikipedia changes and provides full multilingual access [DBP11].

In the context of this work, DBpedia is used in the process of selecting keywords, which have been later applied to the Twitter Streaming API as selection criteria for incoming messages.

Some of the resources leveraged are based on National Vulnerability Database (NVD). The first of them is **CVE**, which stands for **Common Vulnerabilities and Exposures**. It is an international, community based effort supported by the government, industry and academic initiatives with the goal to facilitate identification, finding and fixing of software vulnerabilities. The details about CVE have been already provided in Section 3 in Chapter 3.

CPE (Common Platform Enumeration) is another initiative from Mitre used in this work. It is a naming scheme for information technology systems, platforms and packages. In this work it is used as a source of data related to software products and vendors.

CPE defines a naming syntax and conventions for constructing CPE names. CPE names are represented by a URI (Universal Resource Identifier), providing information about the platform, vendor, product name, version, update, edition and language.

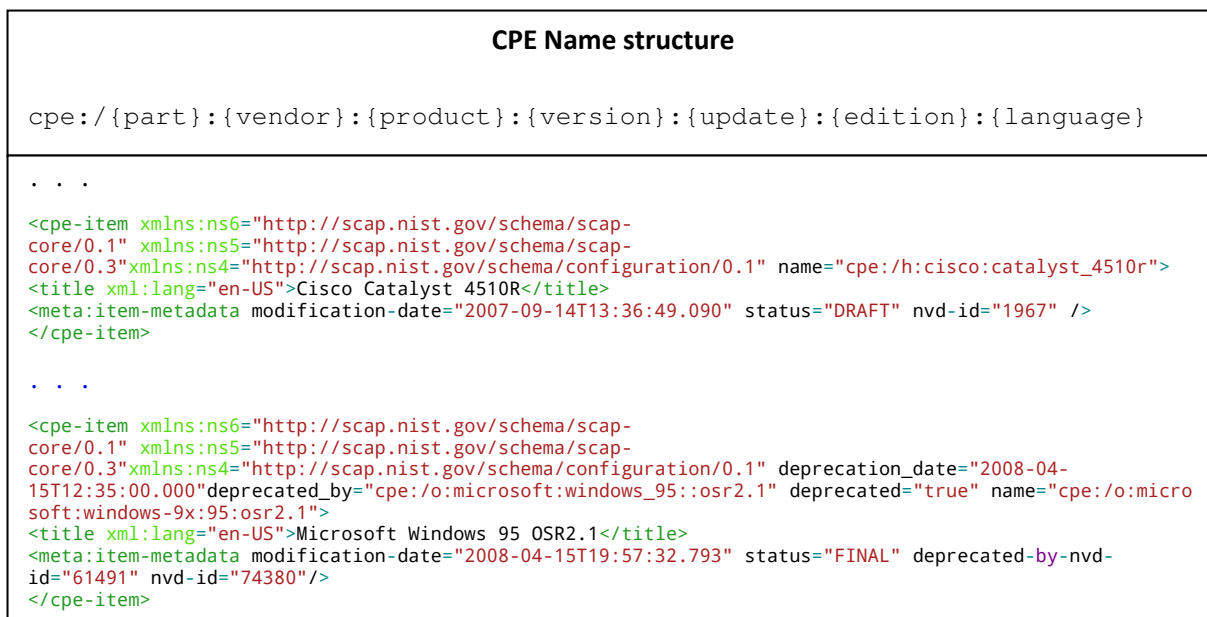


Figure 10: CPE naming structure and example entries

The CPE naming structure is represented in Figure 10. It also shows examples of names for hardware and software products. The *part* field as shown in Figure 10 refers to the type of resource, which is either hardware (h), operating system (o), or application (a). The last five components of the name are optional.

The next source of relevance is **OSVDB (Open Source Vulnerability Database)**. OSVDB (Open Source Vulnerability Database) is an independent and open source database founded in 2002. Later in 2004 Open Security Foundation (OSF) has been established in order to ensure OSVDB’s long-term viability. The goal of OSVDB is to provide accurate, current and unbiased technical information on security vulnerabilities.

OSVDB provides up to some extent similar data as CVE. Its data model of is presented in Appendix 3.

Freshmeat is the next source used in this project. Although not directly related with the software security, Freshmeat is one of the largest web indexes of Unix and cross-platform software. It focuses on open-source software and provides comprehensive information about it. This way the programming environment, license and category (tag based) of the software can be extracted. Currently, Freshmeat does not provide its database directly to the clients (in downloadable form), but it must be queried using predefined API methods.

4.2.1 TweetCatcher

TweetCatcher is the first component in the software framework. It is used to receive and store messages from the Twitter Stream API. The messages received are selected on the keyword based criterion, representing the subset of all messages related to the domain of software security.

TweetCatcher has been developed as a separate, background service, which connects to the Twitter Streaming API³⁸ and receives an active stream of Twitter messages conforming to the predefined criteria. The application stores the messages in an internal format, which is later converted to the RDF/XML representation, used by other parts of application framework.

Figure 11 contains overview of the processing workflow by the TweetCatcher component. There are three options for retrieval and storing of the messages from the Twitter Stream.

The first one covers the retrieval based on the keyword filtering. In this case, the system continuously receives the messages from any Twitter user. The received messages match with the list of predefined keywords.

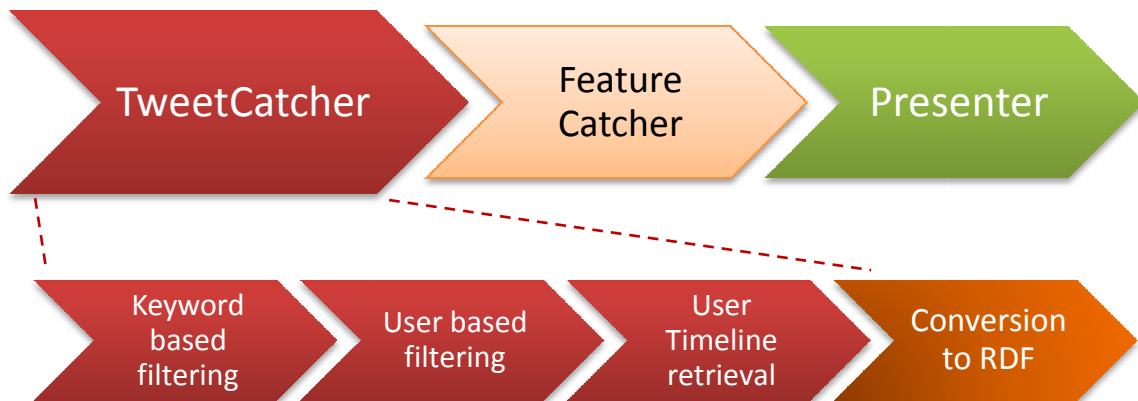


Figure 11: Retrieval of information from Twitter Stream

The second option receives the messages from the predefined set of the users, without other criteria. The third option can be considered as an extension of the second one. It enables the crawling of older posts from the user's Timeline. These posts can be then combined with the

³⁸ Using Twitter4J, which is described in the preceding section

real-time based message retrieval, in order to get archive of user's messages for an extended period.

In this work only the first option is used, while the other ones were tested. Additional part of the workflow of *TweetCatcher* is the conversion to RDF/XML. This step is performed in any case, as the background service stores the data in an internal format.

Two groups of keywords were selected for connection to the Streaming API, as presented in the Table 1. The keywords from the first group were semi-automatically selected. Initially, they have been partially derived from DBpedia by applying the SPARQL query from Figure 12 on its public endpoint. The result has been cleaned from less relevant terms and merged with manually compiled list consisting of other terms. The list of other terms included the keywords related to the public vulnerability databases (such as CVE or OSVDB) and some other terms usually occurring in software vulnerability related messages.

Based on the experience with the data retrieved from DBpedia, it should be noted that the quality of information received in this specific case is lower than expected. After detailed look at the categories in Wikipedia, which serves as a ground for DBpedia knowledge database, it can be concluded that the organization of information on Wikipedia in this case has not been optimal in the term of category classification.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?label WHERE {
  { ?category skos:broader category:Computer_security_exploits .
    ?result dcterm:subject ?category .
    ?result rdfs:label ?label .
    FILTER ( lang(?label) = "en" )
  } UNION {
    ?category2 rdf:type yago:ComputerSecurityExploits .
    ?category2 rdfs:label ?label .
    FILTER ( lang(?label) = "en" ) }
}
```

Figure 12: SPARQL query used for the extraction of security exploits

For instance, some terms have been given equal category although their concepts are not related. Therefore, for the criteria of relevance author used its own judgment based on the domain comparison. For example, the terms *Timeline of computer viruses and worms* and *Extended Copy Protection* are not related to software vulnerability or security exploit domain concepts such as *Riskware* or *SQL injection*.

Group 1		Group 2
cve 2010	correlation attack	vulnerability
cve 2011	computer fraud	vuln
cve 2009	watermarking attack	vln
buffer overflow	passive attack	vulnerable
privilege escalation	meet-in-the-middle attack	
arbitrary code execution	software attack	
heap overflow	software vuln	
dangling pointer	sw vuln	
zdi-10	software vulnerability	
zdi-11	sw vln	
zdi-09	man-in-the-middle attack	
relay attack	xsl attack	
replay attack	sql injection	
remote file inclusion	exploit bug	
zero-day	bug attack	
0day	dll injection	
integer overflow	ransomware	
xss exploit	malware	
race condition	adware	
code injection	scareware	
stack overflow	crimeware	
trojan	bug fix	
remote file inclusion	sql-injection	
session poisoning	ssi injection	
osvdb	security fix	
nvdb	security update	

Table 1: Keywords used for Twitter Streaming API

There could be isolated various types of the information domains represented in the first group of the keywords. The most of the keywords relate to security weaknesses, which often occur in the software. For instance, these are *race condition*, *integer overflow*, *xss exploit*, *stack overflow*, *privilege escalation* and others. The other keywords are general in software security domain, such as *security update*, *security fix*, *software attack*, *computer fraud*, *Trojan*, *sw vuln* and others. Finally, the keywords and their combinations, such as *cve*, *zd* and *osvdb* can be assigned to the domain of security vulnerability databases, advisories and general vulnerability markers. The keywords containing label *cve* refer to the CVE list, which is introduced in Section 3 of Chapter 3 and used extensively in some of the analyses presented in Chapter 5.

The keywords from the second group have been manually selected. They are related to the common variants or forms of word *vulnerability*. That word has been used in much wider context than of software vulnerability; however, its inclusion could provide some additional messages relating to the particular software product or software vendor. The dataset derived from the second group of the keywords has been used in analysis presented in the sections 1 and 2.

4.2.2 FeatureCatcher

This component of the application is responsible for the main processing of captured tweets. It performs three main groups of tasks:

- Feature retrieval
It handles connecting to different data sources, retrieves the information stored there, converts it and stores in the structured format which is appropriate for the further processing.
- Scanning of the tweets
It scans the tweet messages and identifies the features or entities present in them. The structured tweet description is augmented with the features recognized, providing the reference to the information or the whole extracted entity (optional).
- Processing, manipulation and filtering of tweet data
The tweet messages are further manipulated, e.g. filtered according to the different criteria groups or merged.

The tasks done by the component are described in activity flow provided in Figure 14.

FeatureCatcher takes the information from several external sources. The term feature used in this work is referred to the information provided by those sources and identified in the tweet messages.

The feature in this context represents the property of a message, which is used for its further characterization and processing. For instance, the message can have properties like the location of the user posting the message, the software products mentioned in the message, structured security vulnerability identified in the message and others, as described in the following sections. Based on the properties characterizing them, the messages can be searched, filtered, clustered, compared, grouped and arranged on any other way, according to the processing workflow. On an aggregate level, the distribution, incidence and similarities between the messages having the properties can be obtained, which is used to gather and recognize the global trends, similarities or dependencies between the features.

The sources consumed by FeatureCatcher include vulnerability database (CVE), platform descriptions (CPE) and software weaknesses types (CWE) from National Vulnerability Database (NVD) and Mitre. Other sources of information include Open source vulnerability database

(OSVDB) and Freshmeat. The method used for information ingestion varies between different sources. For example, NVD provides information in RDF/XML compatible format, while OSVDB and Freshmeat as sources require additional effort to be invested in form of active crawling of its information or conversion from MySQL database.

As previously said, the information is ingested and processed from the respective sources. In this process it is converted to adequate RDF/XML representation, which is to be elaborated in more detail later. These representations are stored in separate files and in further processing and analysis loaded as needed.

The second task, scanning and augmenting of the tweets, is done as the part of the process workflow. In this step, all the features retrieved from external information sources are loaded and merged into one model. Then, each of the tweets is being processed and its content queried against the model.

When the particular feature is found, its reference is included in the tweet message. Optionally, the complete feature (copy of information found) can be stored in the tweet description. This is done optionally, in the cases when simple processing has to be performed. Otherwise, for performance reasons, the usage of standard references is recommended.

This workflow will be described later in this chapter using a practical example. Due to the performance requirements and memory limitations on available platform, some tasks are performed separately and independently. This is also a useful option for the scenarios where only one processing is to be done, such as identification of one class of features in the tweets.

Final output of this task consists of enriched tweet database in RDF/XML form, or as internal in-memory model. This database is later analyzed and treated by other components or applications. Depending on the size of the output, it can be loaded into external RDF store supporting processing of remote queries³⁹, and then used in this or other applications.

The third task of this component is related to the adjustment of amount and structure of the information contained in the tweet database. According to the preconfigured requirements, the application filters tweets based on several criteria, including property type, time of original broadcast, minimal set of properties and others.

³⁹ Such as Joseki or Virtuoso

This component provides flexible options for message manipulation through different layers of filtering. For example, the messages can be selected by criteria of inclusion or absence of the property types. It is further possible to select only the messages which satisfy minimal set of features supported and to include those messages with the other optionally present features in the further processing.

This component also supports extension and inclusion of the other modules. As the experimental ones, similarly integrated were Apache Mahout's Frequent Pattern Mining⁴⁰ technique and retrieval of interesting phrases based on LingPipe. Due to the scope requirements these are, however, not used in this work.

4.2.3 Presenter

The third component of this application, Presenter, queries previously processed and enriched tweet databases, extracts and presents the information retrieved. Figure 13 provides a graphical representation of the features of this component.

The presenter at the first queries the data source, receives and presents the information. The data source can be local or remote. The data in the local data source is stored as a model inside the application, similarly as it is used by other components.

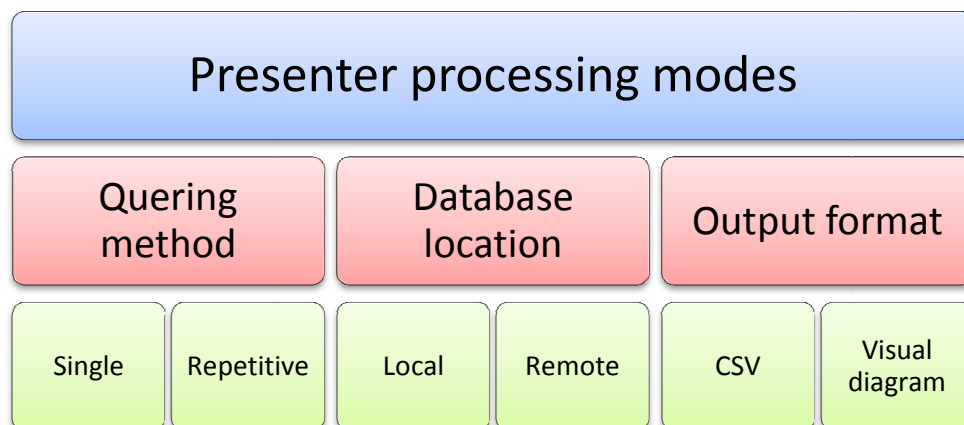


Figure 13: Supported features of Presenter component

⁴⁰ Scalable machine learning library supporting

The remote database refers to the remote system which supports SPARQL queries. This is particularly useful in the cases when the remote system provides the faster and more flexible computing platform. The other use case for this method could be a scaling of a large database into the several computing platforms or the cloud.

Finally, the querying can be done as a single or as a series of repetitive queries. The second case can be useful to get information for some time range, where each time point should be queried for the same information. In this case all responses are grouped and their representation is handled accordingly.

As for the presentation formats, supported are CSV⁴¹ and visual diagram. Diagram is created with the help of HighCharts library, described in the previous section. There is one type of the graphs supported, with the possibility to easily extend the support to the other ones. It is possible to include one or more values in the graph, which are automatically adjusted and drawn using different colors. It is possible to alter the graph interactively. For instance, the display of values can be enabled or disabled in the case where several values are present.

The examples of the graphical output are provided in Figure 24 and Figure 25, explained further in section 1. The upper right part of the image contains the links, which are used to disable or enable drawing of the particular value developments in the graph.

4.3 Processing Steps Done

As it has been mentioned in the section 4.2, the software component designed as a part of this thesis consists from the three parts: TweetCatcher, FeatureCatcher and Presenter. Section 4.2.1 gave a functional overview of the component TweetCatcher.

This component of the system has been executed as a system service from the May 15th to the September 17th 2011 in order to collect the tweet messages according to the criteria given in the Table 1.

The messages have been collected in two independent streams, based on the two groups of keywords as shown in the Table 1. The whole set of the messages resulted in 808,395 and 547,595 messages respectively for the given time period.

⁴¹ Comma Separated Variable, used for representation of tabular data

The information the TweetCatcher collects are the following⁴²:

- number of user favorites
- number of user friends
- flag, showing is the tweet message original or retweet
- number of retweets
- message source (web, phone or other)
- user (poster) id
- user name
- message status id, uniquely defined by Twitter
- message, in the form of text
- number of public lists the user is listed on
- timestamp of the tweet
- number of user followers
- number of user statuses
- user location

The collected data is stored in internal format, which is later converted to RDF/XML representation.

The second component, FeatureCatcher, is more complex in its structure and activities it performs. Figure 14 shows the workflow performed by FeatureCatcher.

It can be noticed that its workflow consists of four main activities, which are described below. These activities are a part of normal workflow and executed sequentially. However, due to the other requirements, each activity can be executed separately and independently.

For instance, the external data processing may be executed periodically in order to update the internal database of program names, vendor data, software vulnerability database and other information it provides.

Tweet augmenting can be done separately, selecting only particular data to be included in augmented (enhanced) tweet messages. Likewise, the model filtering provides the subsets of data obtained for particular purpose or environment.

⁴² For the usage in this work relevant are user id, message status id, message text and timestamp of the tweet.

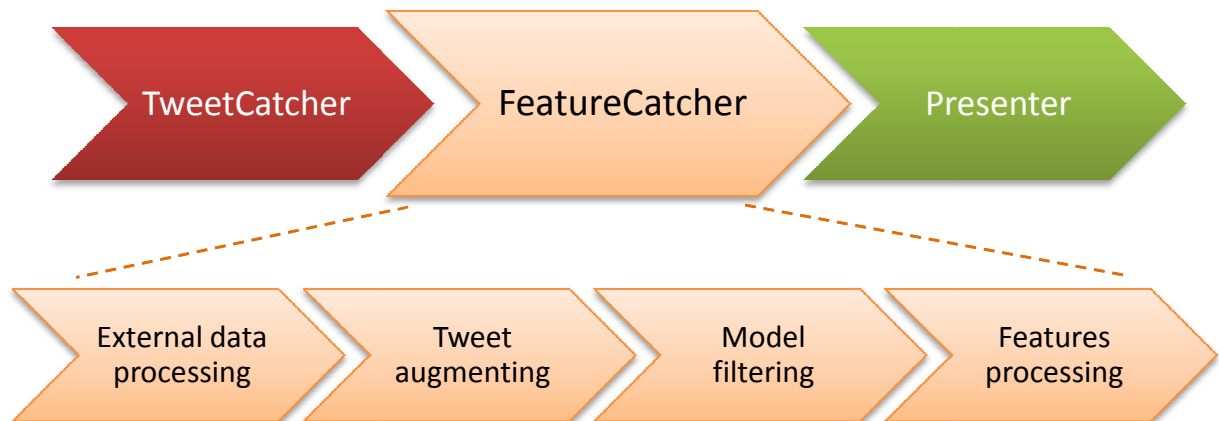


Figure 14: Activities executed by the FeatureCatcher component

External data processing is the first step in regular FeatureCatcher’s workflow. It connects to respective data sources, extracts necessary information, stores it locally and provides as a Jena⁴³ based RDF model. The data sources used are the following (already described in 4.2):

- ❑ OSVDB: this data source is used to extract product and vendor names, which are later applied in the tweet augmentation step.
- ❑ CPE: this source is also used for extraction of product and vendor names
- ❑ Freshmeat: used to get product name and product’s characteristics such as operating system, programming language, tag based category, product’s license
- ❑ CVE: used to gather information about CVE identifiers and related data
- ❑ CWE: loads definitions of supported CWE weaknesses subset

The particular information sources in this step can be omitted if necessary. The loading and conversion of the data differs among the sources.

For the external parties, OSVDB provides SQL database (SQLite and MySQL) containing vulnerability data⁴⁴. This database has to be loaded and queried accordingly in order to extract

⁴³ Java semantic web framework described in the chapter 4.2. For the sake of clarity, in further considerations the mention of term *model* refers to Jena RDF based model.

⁴⁴ Depicted in Appendix 3: OSVDB Data Model Overview.

the information. CVE and CPE provide the data in XML format, which has to be loaded, parsed according to the structure and naming conventions⁴⁵ and then finally converted into RDF.

Lastly, Freshmeat does not provide the data in a raw format. Their data is accessible in the structured format only using the public RESTful API. Due to the limitations imposed by the service provider, at the time of the data retrieval it was possible to submit up to 600 API requests per hour only. Therefore, the data gathering from the Freshmeat has been implemented as an independent service and run in the background for a couple of days.

```
<rdf:Description rdf:about="http://softvuln.at/rdf/vendor#google">
<vendor:hasProduct rdf:resource="http://softvuln.at/rdf/product#chrome"/>
<vendor:hasProduct rdf:resource="http://softvuln.at/rdf/product#earth"/>
<vendor:hasProduct rdf:resource="http://softvuln.at/rdf/product#googlebot"/>
<vendor:hasProduct rdf:resource="http://softvuln.at/rdf/product#web_toolkit"/>
<vendor:hasProduct rdf:resource="http://softvuln.at/rdf/product#desktop"/>
<vendor:hasProduct rdf:resource="http://softvuln.at/rdf/product#web_server"/>
<vendor:hasProduct rdf:resource="http://softvuln.at/rdf/product#picasa"/>
<vendor:nameNormalized>google</vendor:nameNormalized>
</rdf:Description>
```

Figure 15: Vendor details extracted from CPE: data excerpt

```
<rdf:Description rdf:about="http://softvuln.at/rdf/product#powerpoint">
<product:hasVersion rdf:resource="http://softvuln.at/rdf/version#2007"/>
<product:completeName>Microsoft PowerPoint 2000</product:completeName>
<product:completeName>Microsoft PowerPoint</product:completeName>
<product:hasVersion rdf:resource="http://softvuln.at/rdf/version#97"/>
<product:hasEntityType
  rdf:resource="http://softvuln.at/rdf/entityType#application"/>
<product:hasVersion rdf:resource="http://softvuln.at/rdf/version#2001"/>
<product:hasUpdate rdf:resource="http://softvuln.at/rdf/entityUpdate#sp3"/>
<product:hasVendor rdf:resource="http://softvuln.at/rdf/vendor#microsoft"/>
<product:name>powerpoint</product:name>
<product:completeName>Microsoft PowerPoint 2000 sp2</product:completeName>
<product:hasVersion rdf:resource="http://softvuln.at/rdf/version#2004"/>
<product:completeName>Microsoft PowerPoint 2003 sp2</product:completeName>
<product:completeName>Microsoft PowerPoint 2001</product:completeName>
<product:nameNormalized>powerpoint</product:nameNormalized>
<product:completeName>Microsoft PowerPoint 2002 sp1</product:completeName>
</rdf:Description>
```

Figure 16: Product details extracted from CPE, excerpt

At the end of the data retrieval from Freshmeat, one combined model is constructed, which is used in the later stages of the processing. Each model based on the particular source is stored

⁴⁵ Figure 10 explains naming convention of CPE data. Figure 5 provides the schema of CVE.

locally in RDF/XML compatible format. This model can be loaded in the subsequent executions of the application, or regenerated if specified by application configuration.

Figure 15 and Figure 16 illustrate the information received from CPE and represented in RDF/XML format. The former one presents the vendor entry, containing the references to the products maintained by respective vendor - in this case Google and its products like Picasa, Earth etc.

Similarly, Figure 16 presents excerpt from data extracted about Microsoft PowerPoint. This data include several variants of the full product name as well as version and/or update variants. The data also include short (and lowercase normalized) product name. These variants can be used in product name matching in tweets, with the different level of precision and correctness. Analog to the vendor entry, the product entry also includes the references to the vendor and entities like version or update.

Figure 17 presents the product entry data obtained from Freshmeat source. CPE and Freshmeat provide different types of information. Freshmeat is not so focused on formal correctness, does not provide vendor name but provides the category of the software, its license, operating system and programming language. However, the amount of data details differs among particular product entries. Additionally, Freshmeat is more concentrated on open source software. It contains the references to non-production or beta software too.

```
<rdf:Description rdf:about="http://softvuln.at/rdf/product#dovecot">
<product:hasProgLangAsMatched rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  C</product:hasProgLangAsMatched>
<product:hasTopicAsMatched rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  Communications</product:hasTopicAsMatched>
<product:hasTopicAsMatched rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  IMAP</product:hasTopicAsMatched>
<product:hasTopicAsMatched rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  POP3</product:hasTopicAsMatched>
<product:hasLicenseAsMatched rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  OSI Approved :: GNU Lesser GeneralPublic License (LGPL)</product:hasLicenseAsMatched>
<product:hasOSAsMatched rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  POSIX</product:hasOSAsMatched>
<product:nameNormalized>dovecot</product:nameNormalized>
<product:name>dovecot</product:name>
<product:completeName>Dovecot</product:completeName>
</rdf:Description>
```

Figure 17: Product details extracted from Freshmeat, excerpt

As expected, OSVDB provides similar set of the data compared to the CPE. It is because their purpose (software vulnerability analysis) is the similar and their data models and organization are in some extent comparable and compatible.

The fourth source of information, CVE list, is integrated in the model to be able to further identify tweets referring the particular software vulnerabilities. The example excerpt of this data represented in RDF/XML is provided in Figure 18. In the process of data extraction, the NLP⁴⁶ tool LingPipe is used to isolate message chunks from CVE summary.

The message chunks are generated based on the several steps, as follows:

- Sentence extraction from the CVE entry summary. Each extracted sentence is then separately processed in the further steps
- Extraction of the tokens from sentences
- Application of POS-tagging on tokens
- Based on the POS-tags produced in the previous step, the LingPipe chunker applies language model on that data and generate message chunks, which represent the units (tokens) tightly interrelated
- Only parts of the chunks with specific criteria are kept⁴⁷

In this context, Point of Speech tagging (POS)⁴⁸ refers to the activity of determining the role of the words (tokens) in the sentence (part of speech), which often depends on the context. Determined are the relationships with the adjacent words or phrases, which correspond to the grammatical role of the word. Simple forms of POS-taggers identify words such as nouns, verbs, adverbs or adjectives, depending on the structure and complexity of the language.

These chunks may be used later for the further analysis and grouping of the messages. After the data from external sources is gathered, processed and converted into suitable model for a later use, the step of the **Tweet augmenting** is performed.

In this stage, based on the selected preferences, the tweets from the input file are scanned and augmented with additional data. Data enrichment process is done by scanning the tweets and comparing them with the data loaded in the previous step.

⁴⁶ Natural Language Processing

⁴⁷ Default settings keep only tokens which POS-tag based class begins with tags NN (nouns), JJ (adjectives) and R (adverbs). Penn TreeBank POS class notation is used as a reference here.

⁴⁸ Also referred as grammatical tagging

Thus, the tweets containing particular product, vendor, CVE or other⁴⁹ mentions are given additional RDF triple in the form of the references to corresponding resources and/or in the form of literals⁵⁰.

```

<rdf:Description rdf:about="http://softvuln.at/rdf/CVE#cve-2011-1285">
<CVE:summaryChunk rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  regular-expressionfunctionality</CVE:summaryChunk>
<CVE:CVSSAccessVector rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  NETWORK</CVE:CVSSAccessVector>
<CVE:CVSSAvailabilityImpact rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  PARTIAL</CVE:CVSSAvailabilityImpact>
<CVE:CVSSConfidentialityImpactrdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  PARTIAL</CVE:CVSSConfidentialityImpact>
<CVE:id>CVE-2011-1285</CVE:id>
<CVE:summaryChunk rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  reentrancy</CVE:summaryChunk>
<CVE:CVSSScore rdf:datatype="http://www.w3.org/2001/XMLSchema#float">7.5</CVE:CVSSScore>
<CVE:CVSSIntegrityImpact rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  PARTIAL</CVE:CVSSIntegrityImpact>
<CVE:lastModifiedTime rdf:datatype="http://www.w3.org/2001/XMLSchema#long">
  1301527863470</CVE:lastModifiedTime>
<CVE:summaryChunk rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  unspecified other impact</CVE:summaryChunk>
<CVE:summary rdf:datatype="http://www.w3.org/2001/XMLSchema#string">The regular-
expression functionality in Google Chromebefore 10.0.648.127 does not properly implement
reentrancy, which allows remote attackers to cause a denial of service(memory corruption)
or possibly have unspecified other impact via unknown vectors.</CVE:summary>
<CVE:CVSSAuthentication rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  NONE</CVE:CVSSAuthentication>
<CVE:refersVulnerableProduct>
  http://softvuln.at/rdf/product#chrome</CVE:refersVulnerableProduct>
<CVE:CVSSSource rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  http://nvd.nist.gov</CVE:CVSSSource>
<CVE:summaryChunk rdf:datatype="http://www.w3.org/2001/XMLSchema#string">denial</CVE:summ
aryChunk>
<CVE:publishedTime rdf:datatype="http://www.w3.org/2001/XMLSchema#long">
  1299808880450</CVE:publishedTime>
<CVE:summaryChunk rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  unknown vectors</CVE:summaryChunk>
<CVE:CVSSAccessComplexity rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  LOW</CVE:CVSSAccessComplexity>
<CVE:summaryChunk rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  not properly</CVE:summaryChunk>
<CVE:summaryChunk rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  Google Chrome</CVE:summaryChunk>
<CVE:summaryChunk rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  service memory corruption</CVE:summaryChunk>
<CVE:summaryChunk rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  remote attackers</CVE:summaryChunk>
<CVE:summaryChunk rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  possibly</CVE:summaryChunk>
</rdf:Description>

```

Figure 18: Data extracted from CVE list, excerpt

⁴⁹ Any other entity extracted in the previous step

⁵⁰ Exact literal match, where the relation's predicate is suffixed with the term *asMatched*.

The example excerpt of augmented tweet message resource is shown in Figure 19. Depending on the settings chosen, the additional content can be directly included in the form of references to resources, as demonstrated in the Figure 19 for predicate *hasProduct*, or directly as a literals, as shown for *CVSSConfidentialityImpact* predicate.

Additionally, some data can be included only in the form of literal resources (e.g. as matched), due to the design of information source and the scope of this work.

The next step in the flow, **model filtering**, deals with the large amount of data acquired in the previous steps. During the initial tweet collection using Twitter Streaming API many messages are gathered. The process of scanning and enrichment of these messages or the preparation of large number of messages for analysis and visualization require a lot of computing resources.

The component used in this step ensures that the amount of the data engaged in the processing is optimal in terms of the scope and completeness. It can filter messages according to different criteria, ranging from the properties⁵¹ of the resources or time period chosen.

```
<tweet:hasProduct rdf:resource="http://softvuln.at/rdf/product#flash+player"/>
<tweet:hasVendor rdf:resource="http://softvuln.at/rdf/vendor#adobe"/>
<tweet:retweetCount rdf:datatype="http://www.w3.org/2001/XMLSchema#long">0</tweet:retweet
Count>
<tweet:createdAt rdf:datatype="http://www.w3.org/2001/XMLSchema#long">1308367618000</twee
t:createdAt>
<CVE:CVSSConfidentialityImpact rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  COMPLETE</CVE:CVSSConfidentialityImpact>
<tweet:userLocation rdf:datatype="http://www.w3.org/2001/XMLSchema#string"></tweet:userLo
cation>
<tweet:hasCVE rdf:resource="http://softvuln.at/rdf/CVE#cve-2011-2110"/>
<CVE:CVSSSource rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  http://nvd.nist.gov</CVE:CVSSSource>
<CVE:summaryChunk rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  Adobe Flash Player</CVE:summaryChunk>
<CVE:summaryChunk rdf:datatype="http://www.w3.org/2001/XMLSchema#string">wild</CVE:summar
yChunk>
<tweet:user rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Mert SARICA</tweet:use
r>
<tweet:userStatuses rdf:datatype="http://www.w3.org/2001/XMLSchema#long">1073</tweet:user
Statuses>
<tweet:message rdf:datatype="http://www.w3.org/2001/XMLSchema#string">CVE-2011-
2110, Adobe Flash Player zafiyeti istismarediliyor -
  http://stpmvt.com/kCvGou</tweet:message>
```

Figure 19: Example of the augmented tweet

⁵¹ Inclusion, availability, or unavailability of particular properties

The output of this processing is the model, again, which is sent to other components or simply stored locally for a later processing.

The last step in the flow, **features processing**, deals with the analysis of the tweet messages and their features and prepares the results for representation. It is based on several underlying processes which can be also applied separately and independently.

After the activity flow of FeatureCatcher is done, it is possible to use the component Presenter to export or visualize the results. Figure 13 and Section 2.3 already described this component.

Figure 20 shows the example query used to get the data from Table 4. This provides shows aggregate results. However, in some cases it is necessary to perform multiple queries, to cover the smaller considered periods (like one day resolution) in the data range. For such purpose used is the similar query, as shown in Figure 21.

```
SELECT ?vend ?name (COUNT(DISTINCT ?tw) as ?tg) WHERE {
{
  ?tw tweet:createdAt ?createdAt .
  ?tw tweet:hasProduct ?prod .
  ?prod product:hasVendor ?vend .
  ?prod product:hasSourceDB "cpe" .
  ?vend vendor:nameNormalized ?name .
} UNION {
  ?tw tweet:createdAt ?createdAt .
  ?tw tweet:hasVendor ?vend .
  ?vend vendor:nameNormalized ?name .
}
```

Figure 20: Example SPARQL query used to get data

This query is automatically produced by application in order to gather the data for a shorter time range. The application repeatedly changes the values for data range in the query and sends it to the server or performs internal lookup.

```
SELECT ?tw WHERE {
?tw tweet:createdAt ?createdAt .
FILTER(?createdAt > 1312156801000 && ?createdAt<1312502401000) .
?tw tweet:hasProduct
<http://softvuIn.at/rdf/product#wordpress_wordpress> .
}
```

Figure 21: Intermediate repetitive SPARQL query

After the querying is finished, all the data retrieved is grouped and available for export as CVE or as export to visual representation.

The visual representation is based on HighCharts library, which is described in the Section 2. The software framework produces output in the form of HTML file, which contains the JavaScript code with selected adjustments and values to graph.

Presenter component supports one graph type from the library. It is possible to display one or more values and enable or disable their inclusion in the graph interactively, as shown in Figure 24 and Figure 25.

5 Results

This chapter presents the results of the analyses which have been performed using the software framework introduced in Chapter 4 and external statistical software packages. These analyses were executed on the datasets gathered by the component *TweetCatcher*, the part of the same software framework.

The following sections present a various types of such investigations. They do not cover all possible usage scenarios of the framework, but serve to illustrate its diverse possibilities and potential usage of technologies and data sources.

The main object of the analysis is a Twitter message. Figure 22 depicts the example Twitter message after its retrieval from Twitter Stream and the further processing by the framework were applied. It depicts some parts of its structure, pointing to the further references in other datasets.

In the concrete example, after the framework is done with the processing, two objects are identified in the message: CVE entry reference (green rectangle) and the software product name (blue rectangle). These references are added to the structure of the message during the processing phase. Their purpose is to point to the external or internal objects, describing the related information. In this case, it can be noticed that the identified CVE entry refers to the specific CWE weakness type, which is further described by the elements from the *Impact Scope*, *Time of Introduction* and *Consequence Scope*. The coming section describe further details and usage of these elements.

The demonstration of the framework usage begins with comparison of share ratio of prominent software vulnerability sources in the Twitter dataset obtained. The sources and dataset are already described in Chapter 4. The next section analyzes the share ratio of products and vendors in messages. For visualization in this chapter used is the *Presenter* component of the software framework, which is already introduced in Section 2.3 in Chapter 4.

The next section is dedicated to the analysis of types of the weaknesses referred in the messages having proper CVE entry reference. The section provides the description of obtained dataset, based on several perspectives. Then, its static part – CVE list, is analyzed from the aspect of several properties and trends. The next part of that section focuses on the analysis of Twitter stream dynamic in the light of software weaknesses types.

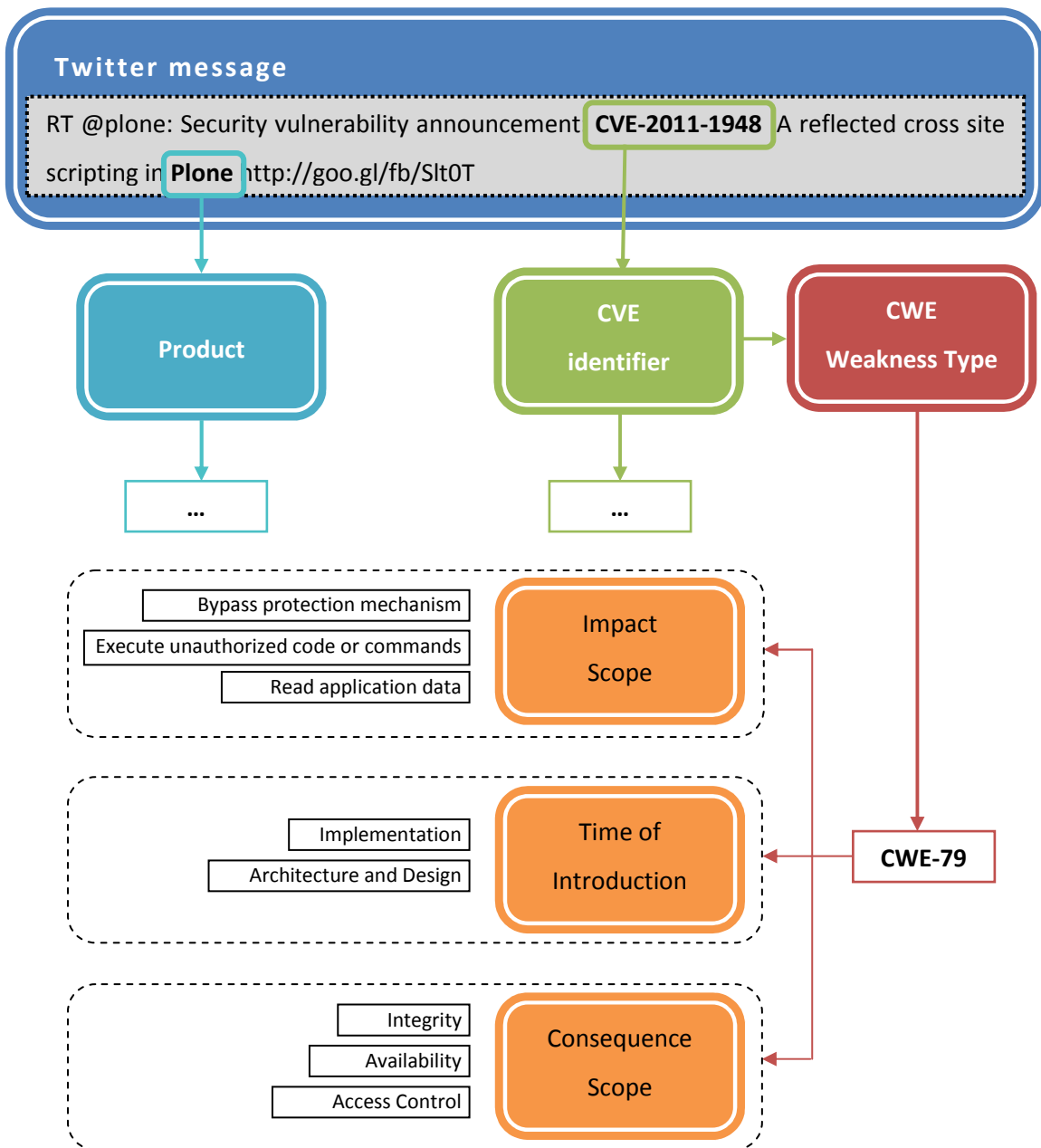


Figure 22: Example markup of Twitter message

The graphic overview on the weekly level is provided, which is further used to extract data and trends for periods and properties notably visible on the trends plot. Furthermore, the third section is concluded with the general overview of data and trends obtained.

The next part of this chapter deals with some of the inherent characteristics of the CWE weakness type (represented on Figure 22), trying to discover the preferences of Twitter users in the light of these properties on an aggregate basis.

Finally, the last analysis closes this chapter. It deals with the publication and modification time of a CVE entry. These properties are considered from the perspective of Twitter stream and CVE entries distribution in the messages. In other words, examined is the effect of CVE announcement to the user activity on Twitter.

5.1 Share Ratio of Prominent Software Vulnerability Sources

Many sources report the software vulnerabilities. They may include the companies, organizations, government establishments, or even vendors and private persons. The credibility, popularity and acceptance of those sources may differ significantly among the target population.

Figure 23 contains the share of the messages including the reference to one of the four prominent software vulnerability sources. They are:

- CVE - Common Vulnerability Enumeration
- MS - Microsoft
- ZDI - Zero Day Initiative
- SB – Security Bulletin (US CERT)

These are industry-standard subjects with different backgrounds. CVE has been mentioned earlier in this work. This is initiative from Mitre, recognized by National Vulnerability Database and supported by 73 organizations and their advisories worldwide.

Microsoft, as one of the largest software vendors maintains its own security platform and advisory. Founded by TippingPoint, ZDI stands for a program for rewarding security researchers

in their discovery and analysis of software vulnerabilities. Lastly, US CERT maintains cyber security bulletin in the form of weekly summary of new vulnerabilities.

These organizations and their publications are closely monitored worldwide by many users and organizations. The vulnerabilities published by them cannot be considered as unique and mutually exclusive in the computer security world - the organizations taking part in this process usually refer to each other. Hence, a newly discovered vulnerability may be published in diverse sources.

In the recent time, the organizations invest more resources in order to improve the consistency of their reports and to include references to the reports or lists maintained by other organizations. Mitre with its CVE Adoption process and relation to 73 security advisories is a typical example of that.

The relative significance and popularity of the vulnerability sources among the users may be partially described through the share ratio in corresponding Twitter messages.

Source	Num. of messages
CVE	2222
ZDI	1120
MS	687
SB	431

Table 2: The cumulative number of tweets representing each vulnerability data source

In this case, during the same date range⁵², Twitter Stream was monitored for the four words only: *vulnerability*, *vuln*, *vln*, *vulnerable*, as described in Table 1 from Section 2.1 in Chapter 4. The word *vulnerability* and its variations have been selected as a most common denominator for the topic of vulnerability. As this word alone may have ambiguous meaning among diverse contexts, the reference to the common security advisories and their entries has been extracted based on the formal format specific to the advisory.

Figure 23 displays the relation between those four advisory sources, through the weeks in the data range. The relation is expressed with the cumulative number of the tweets containing references to the security advisory entries. The figure shows that CVE is the most common source discussed, followed by ZDI and MS, which have varying distribution through the time.

⁵² From the May 15th to the September 17th 2011

In the each week reported the CVE related messages hold the significant portion. That fact confirms the trust and acceptance of CVE as a vulnerability source.

The varying distribution of ZDI and MS messages can be attributed to the lower and instable publication frequencies in their advisories. Microsoft maintains software products, which are quite widespread worldwide. That fact makes it an essential source worth of public attention. As many of its products are based on the similar development practices and share the common platform and code, it may happen that the discovery of one vulnerability type in some product initiates the research and possible discovery of the vulnerabilities in other similar products of the same vendor. This could partially explain the occasional periods of high reporting level, which may be the result of the same or similar software weakness found in several different products and covered by separate vulnerability disclosures.

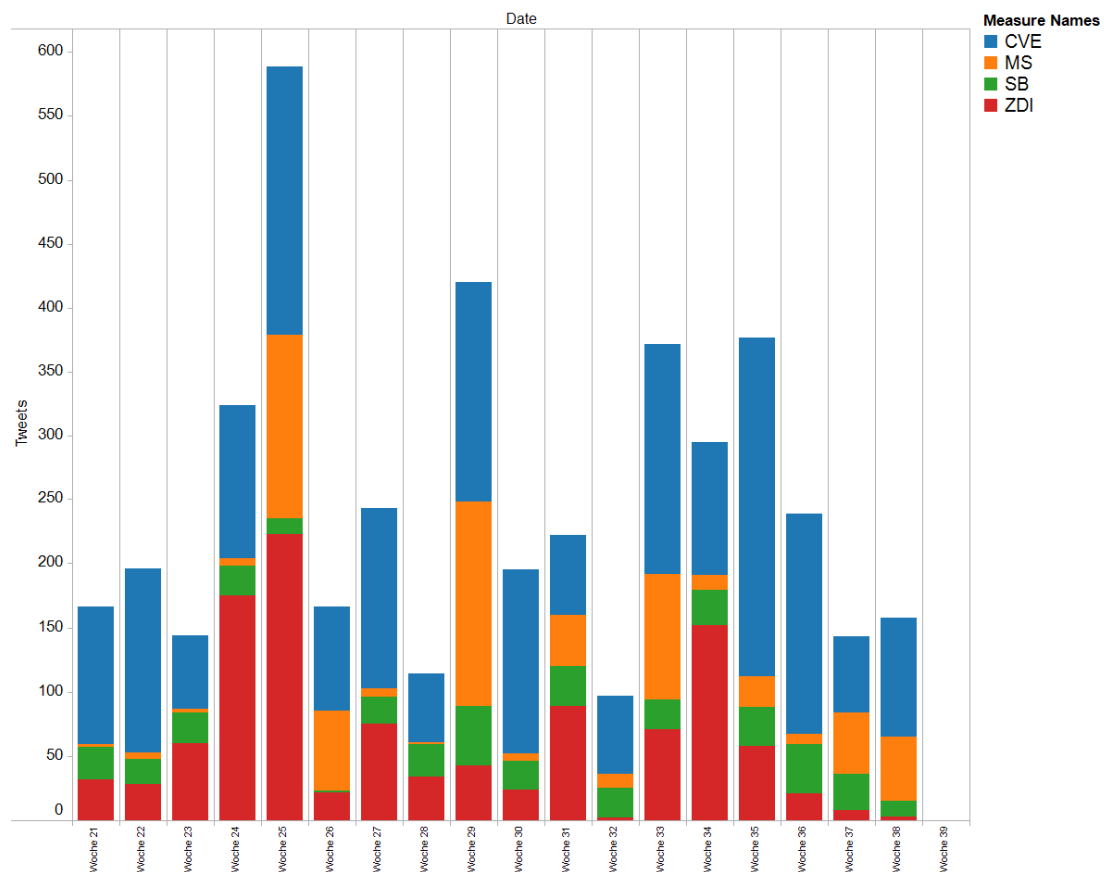


Figure 23: Ratio of tweets containing references to vulnerability sources

ZDI on the other side focuses and specializes on publishing of new vulnerabilities in a wide range of software. The backed business model is based on paying the incentives to the security

professionals for finding and describing of new vulnerabilities. They work hard with the strong vendors in the software industry to coordinate the discovery, analysis and disclosure of the software vulnerabilities. The focus on the new vulnerabilities makes it attractive an hub and source of information for computer security experts and wide user base.

US CERT's Security Bulletin is the least represented source. The security bulletins are published on a weekly basis and in many cases include references to the other sources, like CVE entries. The frequency of publication and its high level of dependences on external parties make it probably less appealing source of information.

Based on the data from Table 2 and Figure 23, it can be concluded that CVE mentions hold roughly one half of all references on an aggregate level. Also, in the sense of the weekly iterations it holds significant and stable amount of the reports. Based on that, this database is recognized and popular among the users. The second source, ZDI, holds approximately 25% of all mentions in the aggregate data. As an organization, they target new, *zero day* vulnerabilities and invest resources to find vulnerabilities with higher impact and significance in the industry. Those facts could support high variation in the frequency of ZDI mentions in the data.

Finally, it should be mentioned that CVE is the only of the four sources providing the vulnerability data in a structured format to external parties. This makes further research, analysis and integration in other tools much more accessible for other subjects. The absence of such structural model in the second case is also the reason why ZDI data could not be further investigated.

5.2 Share Ratio of Software Products and Vendors

Similarly as in the previous section, the aggregate data from the second stream based on the keywords such as *vulnerability, vuln, vln, vulnerable*⁵³ has been used to review the mentions of popular products and vendors in the period from the May 15th to the September 17th 2011. Consequently, the tweet messages have been scanned for the product or vendor names found in the local database. Previously, the local database has been populated with entries from the databases such as CPE, OSVDB and Freshmeat.

⁵³ The process is described in Section 2 of Chapter 44.2.1, while the keywords were introduced in Table 1

Vendor	Count	Vendor	Count
Google	27626 (9272)	PayPal	1568
Apple	10405	Linux	1345
Microsoft	7864	Gentoo	1119
Cisco	7243	Corel	1072
WordPress	3405	SuSe	1071
Debian	3136	Conectiva	1067
Apache	2983	Joomla	997
Facebook	2939	Sony	875
Adobe	2730	Mozilla	740
Sun	2408	Symantec	718
Skype	2347	AOL	685
PHP	2085	HP	606
Oracle	1905	IBM	497
Macromedia	1846	ISC	495
RedHat	1815	Siemens	450

Table 3: Aggregate distribution of the vendors

Table 3 lists the most commented software vendors for the considered period. Both the direct vendor mentions and products owned by the vendors were counted. For the identification of names used was the CPE database.

After initial testing, the databases from OSVDB and Freshmeat were excluded from the product and vendor recognition process. The reason for that is based on the empirical fact that OSVDB contains to a great extent the same products and vendors as CPE. In some cases the naming convention is different from the one used in CPE, producing confusion in the results.

The second and more critical reason for the exclusion is the low quality⁵⁴ of Freshmeat database. It consists of many products less popular or in immature development stage, some of them having the names based on the common English words. Therefore, its usage without additional complex processing⁵⁵ does not bring any noteworthy advantage and may produce significant noise.

Table 4 shows the most commented products for the same period as considered in the Table 3, based again on CPE product database.

⁵⁴ In the term of names of software product and this particular usage scenario

⁵⁵ *Word sense disambiguation*, used to recognize the context of the word and differentiate its status. E.g. is the function of the word to describe the product or vendor name, or it serves other function in the sentence. Example: *act* may represent vendor name or the verb in the sentence

Product	Count	Product	Count
Cisco IOS	6258	AOL Explorer	496
Google Android	5748	Mozilla	453
Apple iPhone	3965	MS Internet Explorer	423
Wordpress	3405	Oracle Applications	419
Apple iPad	2988	ISC Bind	365
Apache ⁵⁶	2890	Sybase Central	363
Skype	2347	Microsoft Outlook	318
Microsoft Windows	2210	Microsoft WINS	307
PHP	2076	Macromedia Shockwave	299
Microsoft Word	1634	Rim Blackberry	286
Macromedia Flash	1496	Microsoft Activex	256
Linux	1067	Microsoft Hotmail	240
Google Chrome	803	Apple Quicktime	237
Linux Kernel	661	PhpMyAdmin	237
Mozilla Firefox	646	Adobe Acrobat	232

Table 4: Aggregate distribution of the products

In the Table 3 it can be noticed that Google had an unusually high relative level of notions, however, from Table 4 it can be read that its direct products were mentioned on the average level. The difference between these numbers can be attributed to several factors in the general case, but the most prominent and significantly impacting one in this particular case is the noise. Concretely, the noise has been caused by the bot system posting the message related to the exploit database built around Google search⁵⁷. This bot caused 18,354 Google mentions for the same period. If that number gets excluded from the total counting, the number of Google's mentions in the dataset falls down to 9,272, as shown in the table. This way, the disproportion in data is removed to a great extent.

Before the further analysis is continued, some facts related to the data from Table 3 should be first commented.

Debian and Apache from the Table 3 in the most occasions refer the same product, Apache HTTP Server. The inclusion of both is the result of a discrepancy in the CPE database, which identifies Apache as a product of Debian. Debian is, however, the name of a Linux distribution and the organization behind it. The distribution includes binary and source packages of the applications shipped with the system, the part of which is also Apache HTTP Server. Therefore,

⁵⁶ This product refers to Apache HTTP Server. The vendor for this product identified in CPE database is Debian, however, that is not completely correct.

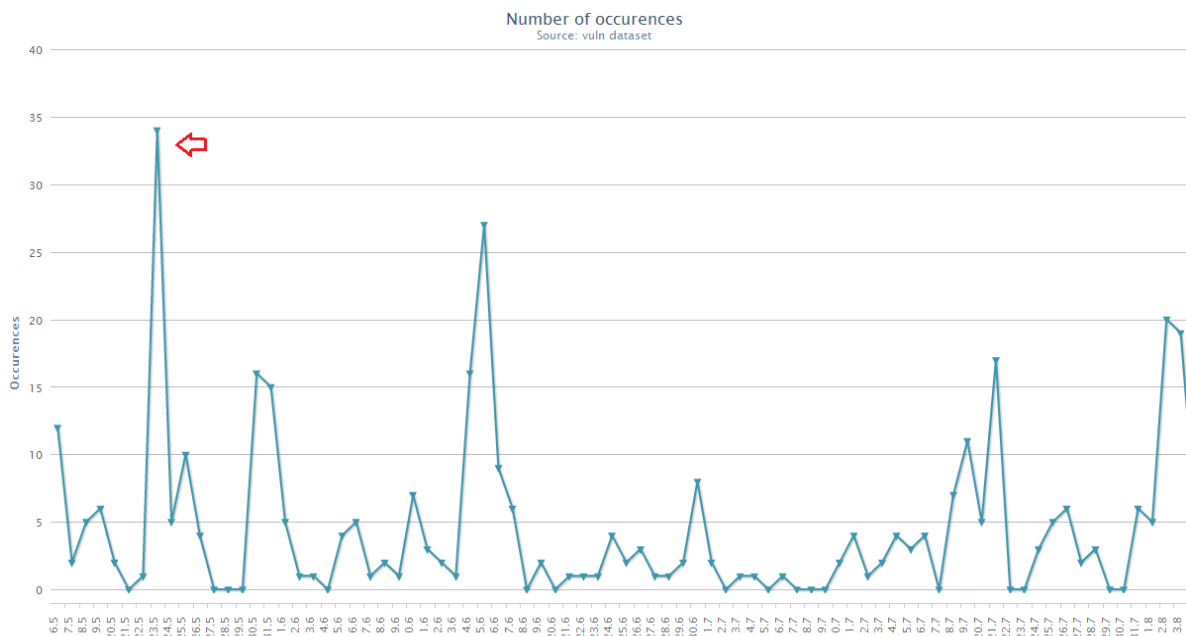
⁵⁷ Example message:

honey4 : There are 522 non-recurring vulnerable paths (google dorks) in the database.

although highly positioned in the table, Debian as a *packager* is not “responsible” in the corresponding extent for the vulnerability related to the product of *the other* vendor.

The next essential detail worth notion is the ambiguity problem in the CPE list related to Linux distributions in general. When the term such as *linux* is included in the tweet message, the system relates it with the several known Linux distributions concurrently. The Table 3 shows that Gentoo, Corel, SuSe and Conectiva have a similar number of mentions, which is the result of that flaw. As the CPE database connects the term *linux* with the several vendors, so this error propagates consequently to the results in table. This is also obvious from the example of the vendor Corel, which long time ago closed the development of its Linux distribution.

The third case is the relation of Sun and Oracle. Oracle acquired Sun recently. The result is that its products are referred under two different brand names. This discrepancy is not entirely correlated to the CPE database: the users also mention the different company names. Now, the numbers presented in the Table 3 and Table 4 will be visualized in the light of two representative examples. Figure 24⁵⁸ displays the traffic on Twitter stream related to IBM as a vendor, and all its products. IBM maintains a complete and complex portfolio of the software products.



The red arrow on Figure 24 points to the spike representing the burst in the mentions of IBM as a vendor. The spike started at May 23th and ended before the May 27th. The activity in that period has been caused by discovery of new XSS vulnerability in IBM's WebSphere Portal.

Initially, the vulnerability has been disclosed by the vendor⁵⁹ on May 23th, followed with advisory released by Secunia⁶⁰, company specialized in the vulnerability management. It is interesting to note that the same vulnerability has been covered also by CVE-2011-2172, published May 26th, three days after initial activity spike.

Figure 25 shows the development of WordPress mentions in the Twitter stream. The region marked with the red circle refers to the period from August 1st to August 6th. In this period great and rapid variation happened, relative to the average number of product mentions in the Twitter stream.

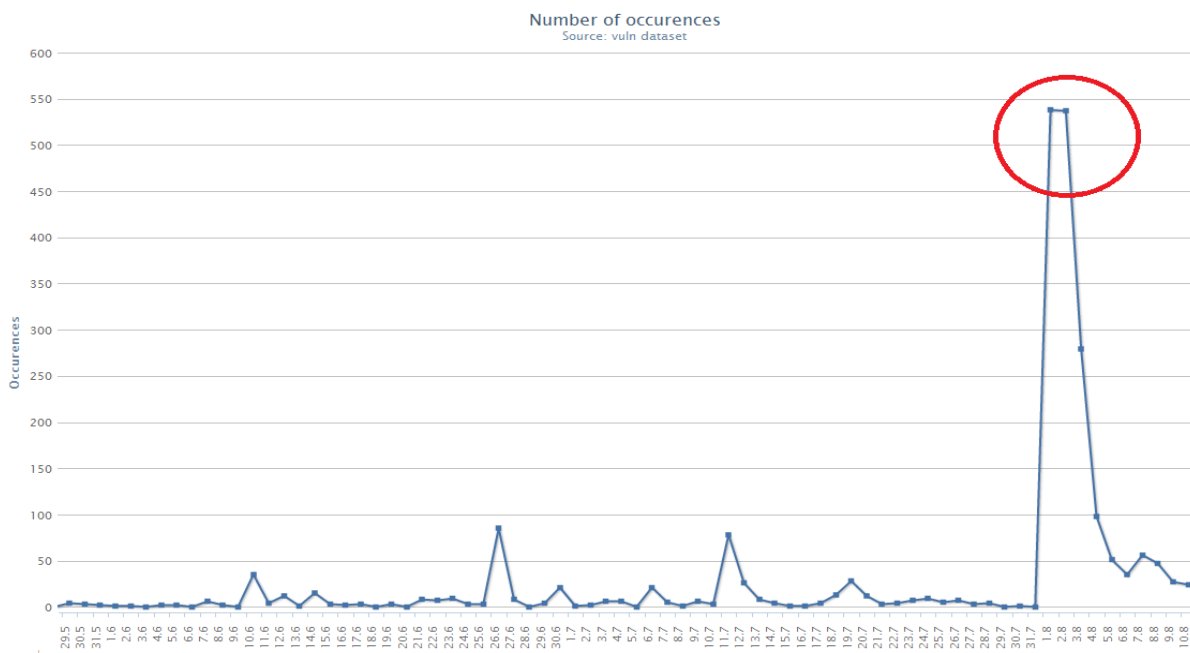


Figure 25: WordPress mentions

⁵⁹ <http://www-01.ibm.com/support/docview.wss?uid=swg24029452>

⁶⁰ <http://secunia.com/advisories/44700>

The high level of activity among Twitter users has been caused by zero-day vulnerability in the WordPress image utility⁶¹, which impacted many WordPress themes⁶². The vulnerability allowed remote attackers to upload and execute arbitrary PHP code in the application’s cache directory. Although this vulnerability has been fixed in the short time, it has been widely exploited on Internet⁶³. It was not directly reported in the CVE database, at least in the time of this writing.

Figure 26 illustrates the relative trends in the usage of the search service by Google (upper part) and the relative reference volume across the news providers covered by Google News (bottom part). Although the Google Trends provides approximate information only, the significant spike in the beginning of August is easily to be spotted, confirming the rise in the activity.

These examples illustrated the relevance of the Twitter stream activity to the analysis and discovery of the trends in software vulnerability discovery and disclosure.



Figure 26: TimThumb relative search and news trends by Google

In the first example, the spike in Twitter conversation in topics related to the vendor unveiled the release of software vulnerability in its product. It happened the same day of its release by

⁶¹ TimThumb, image resizer for WordPress

⁶² <http://markmaunder.com/2011/08/01/zero-day-vulnerability-in-many-wordpress-themes/>
<http://www.networkworld.com/news/2011/080211-zero-day-vulnerability-found-in-a.html>

⁶³ In the November 2011 Google search for the term “timthumb vulnerability” returned nearly as 2.5 million of hits, while the term “timthumb exploit” returned more than 1.2 million of hits. These numbers confirm the wide impact of that vulnerability.

the vendor and well before other relevant sources approved and distributed information⁶⁴ [IBMXSS11, CVE2011-2172].

In the second example, the communication burst has been exceedingly strong, indicating the enormous importance, underlying risk and possible impact of the vulnerability. The widespread use of WordPress on Internet and already mentioned amount of the search results in Google are in correlation and support this statement further.

The second example further shows that the following of the public security advisories may not be enough to protect some systems. In some cases the exploits may be available well before the relevant security body approves, evaluates and confirms the vulnerability release. In the current case, unlike the example of IBM WebSphere Portal, the vulnerability has not been officially published in the CVE list at all.

The case of WordPress and its vulnerable utility describes the software product which is published by the individual person, on the FLOSS basis. In some cases where the vendors are not commercial companies and do not own the processes or practices related to the vulnerability discovery, management and release process, the complete integration with the security professional related bodies and advisories may be omitted.

Therefore, the users of such products may be exposed to a greater level of risk. The usage of social networking systems, their crowdsourcing properties, automated analysis of their data and integration with the security systems and databases may be one of the ways to prevent or reduce such risk.

5.3 Types of Weaknesses

In the process of the software vulnerability discovery and analysis, new vulnerabilities are appended to the NVD list and thus promoted to the NVD list entries. Each entry, as described in previous chapters, consists of structured description of software vulnerability.

In many cases, the reference to the corresponding software weakness type in the CWE list is assigned to the new NVD list entry. This relation maps the NVD list entry, e.g. the software

⁶⁴ For instance, three days before it got published in CVE repository

vulnerability, with the corresponding entry from CWE classification representing the category or type of related software weakness.

However, not all NVD list entries receive a CWE reference, nor are they classified into the full range supported by the CWE list. CWE entries used by NVD are described in CWE’s list *View 635 “Weaknesses used by NVD”*.

This view includes 19 items, of which 6 are described as categories, 12 as weaknesses and 1 as a compound element. Therefore, all NVD’s vulnerabilities which obtain the classification to the CWE list are related to one of these members.

As of the time of this writing and based on the database of vulnerabilities administered by NVD, the total number of software vulnerabilities reported since January 1st 2009 is 11,883. Of these, 9,719 entries include reference to corresponding CWE list entry.

Around 81.79% of the vulnerabilities included in NVD’s CVE list since 2009 have been given matching CWE entry. Figure 27 represents the development of that relation on the monthly level⁶⁵. The total number of vulnerabilities published by NVD list since its beginning is 48,208⁶⁶.

Source \ Year	2009	2010	2011 ⁶⁷
CVE	4,174	4,594	3,115 ⁶⁸
CWE	3,502 (83.90%)	3,708 (80.71%)	2,509 (80.55%)

Table 5: Yearly development of publication of CVE entries

Table 5 lists the amount of vulnerabilities discovered each year and listed in NVD software vulnerability database. CVE row represents that amount in absolute values, while the CWE row shows the fraction of software vulnerabilities discovered having reference to the corresponding CWE list entry.

⁶⁵ The NVD CVE vulnerability list has been analyzed for period from January 1st 2009 to October 7th 2011. It should be noted that CVE entries have identifiers of *publication time* and *last modification time*. This comparison includes only entries which are *published* in related time range.

⁶⁶ Based on October 14th report from <http://nvd.nist.gov>

⁶⁷ CVE related data for 2011 contains entries published up to the October 7th. It is incomplete for the year 2011, but this fact has been taken into account during the consideration and comparison of the data. Therefore, only the strong trends with significant difference have been considered.

⁶⁸ If extrapolated to the end of the year, the expected number would be 4,143

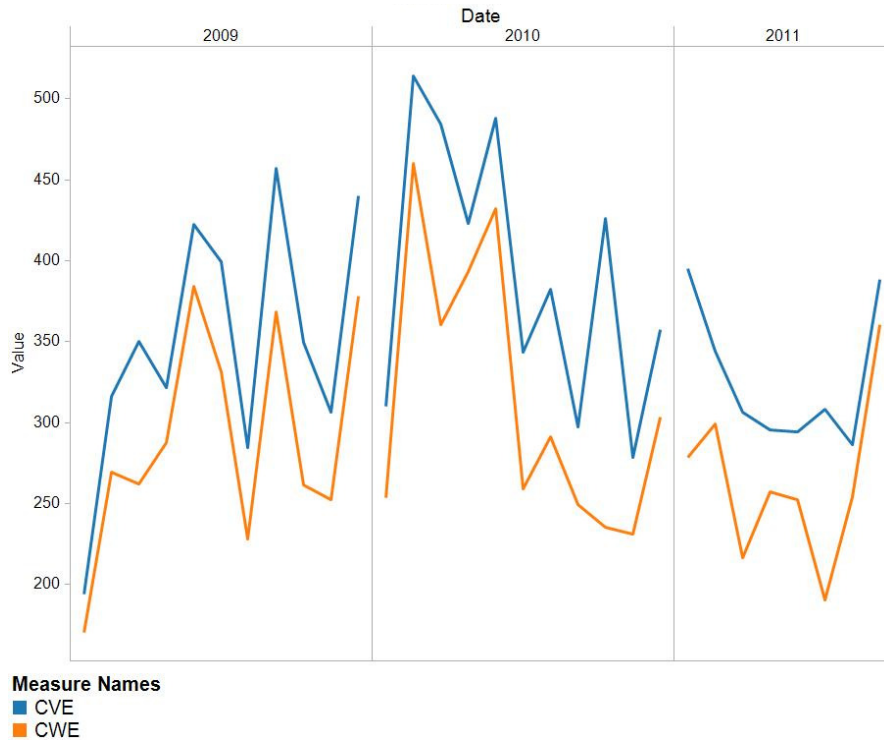


Figure 27: Vulnerabilities published in NVD (total, including ones with CWE reference)

The quantity of those vulnerabilities is characterized both by using absolute and relative values. It should be noted that, as of the time of this writing, the values from the fourth quarter of 2011 are missing. However, even partial data available for 2011 can be used to recognize particular courses with strong affinities.

5.3.1 CVE List Publication Dynamics

The development of the software vulnerabilities in their absolute amount⁶⁹ for the years 2009, 2010 and 2011 is presented in Figure 28⁷⁰. From the Figure 28 some trends can be observed.

Discovery of software vulnerabilities of the class *Code Injection* has dropped significantly in 2011 (43 CVEs), compared to 2009 (231) and 2010 (262). Similarly, *Path traversal* related vulnerabilities since the beginning of 2011 have been reported in 72 CVE list entries, compared to 269 entries found in 2010.

⁶⁹ As classified by NVD in CVE list and mapped to the CWE entries. The data for 2011 is incomplete, as explained on the previous page. Only the strong trends are considered.

⁷⁰ The incomplete labels refer to *Improper Restriction of Operations within the Bounds of a Memory Buffer (CWE-119)* and *Permissions, Privileges, and Access Controls (CWE-264)*, respectively

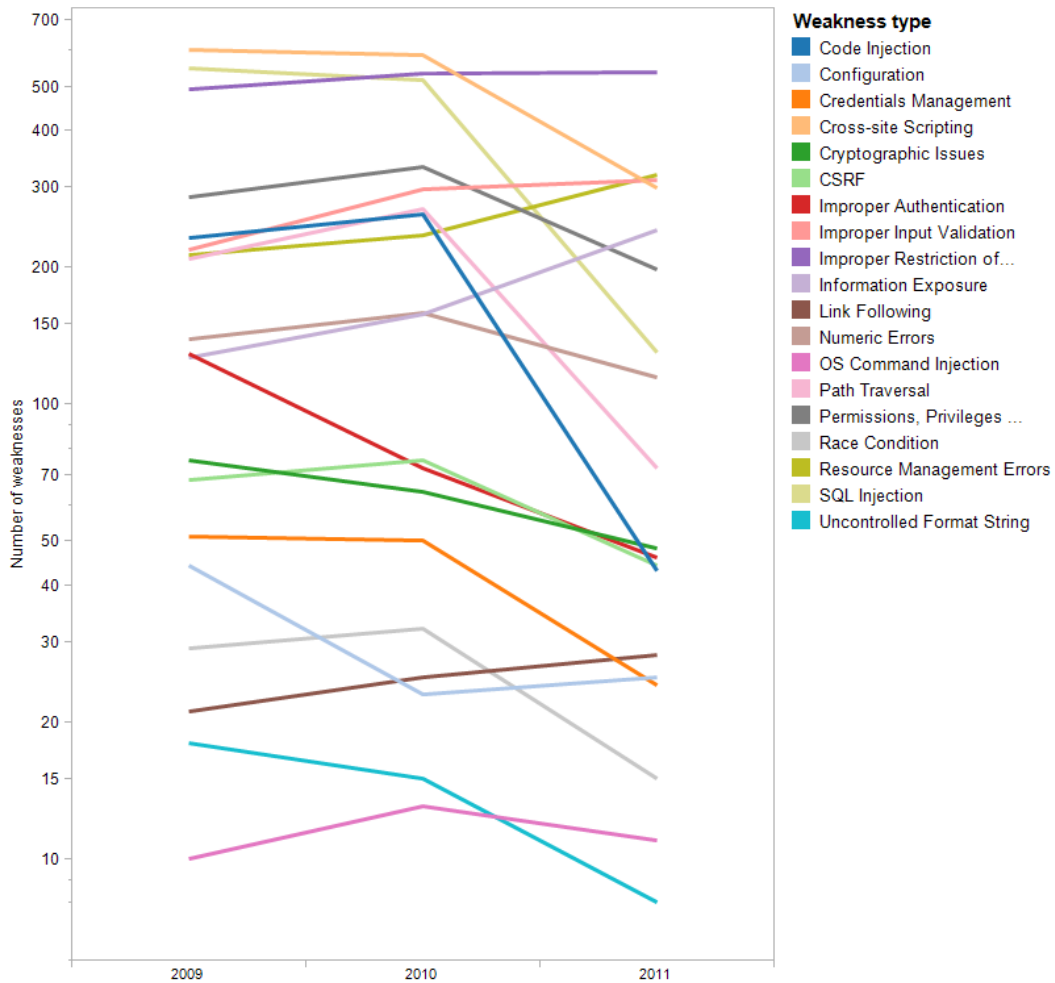


Figure 28: Distribution of CWE weaknesses in NVD CVE list

This trend is even more accented in the case of *SQL Injection* related vulnerabilities, which dropped from 546 and 514 inclusions (2009 and 2010 respectively) to only 130 in 2011. Neuhaus and Zimmermann describe the similar trend for the years 2008 and 2009 in their technical report [SECTR10].

On the other side of the trending, the class of *Information Exposure* related vulnerabilities has strong upward tendency, followed by *Resource Management Errors* and *Improper Input Validation* classes. The situation visually described in the Figure 28 is further detailed in the Table 6. It lists the exact numbers representing the development of each weakness type through the years of 2009, 2010 and 2011. Table 7 helps to get deeper insight and understand the changes in the growth of some weaknesses types. The Figure 29 and Figure 30 serve the same purpose, for the weaknesses types *SQL Injection* and *Information exposure*.

Weakness type	2009	2010	2011	
Code Injection	231	262	43	🔴
Configuration	44	23	25	
Credentials Management	51	50	24	🔴
Cross-site Scripting	599	584	298	🔴
Cryptographic Issues	75	64	48	
CSRF	68	75	44	
Improper Authentication	129	72	46	
Improper Input Validation	218	296	310	🟢
Improper Restriction of Operations within ...	491	532	536	🟢
Information Exposure	126	157	241	🟢
Link Following	21	25	28	
Numeric Errors	139	158	114	🔴
OS Command Injection	10	13	11	
Path Traversal	209	269	72	🔴
Permissions, Privileges, and Access Controls	285	332	198	🔴
Race Condition	29	32	15	
Resource Management Errors	213	235	318	🟢
SQL Injection	546	514	130	🔴
Uncontrolled Format String	18	15	8	

Table 6: Distribution of CWE weaknesses in NVD CVE list

Weakness	Vendor	Count	
		2010	2011
Improper Input Validation	Google	16	74
	Microsoft	47	32
	Adobe	16	25
	Apple	14	14
	Linux	15	14
	Opera	4	10
Improper Restriction of Operations ...	Adobe	88	97
	Apple	73	96
	Microsoft	46	39
	Google	20	41
	HP	10	25
	IBM	17	22
Code Injection	Microsoft	90	5
	Adobe	19	1
	Apple	14	0

Table 7: Trends among the vendors, excerpt

In the Table 7 the *vendor* column refers to the software vendors who got the highest share for the weaknesses type in the time frame. The column *count* lists the number of hits of the software products released by the vendor⁷¹. In the cases of *Improper Input Validation* and *Improper Restriction of Operations within the Bounds of a Memory Buffer* the situation with the top six vendors for the year 2011 was described. The *Code Injection* lists top three vendors from the year 2010.

The row for the type *Improper Input Validation* shows that, even after the first three quarters of the 2011, the number of CVE entries exceeded the previous years' numbers. The cause for that change can be found in the sudden increase of weaknesses of that type in the products published by Google.

The change from 16 reports in 2010 to 74 in the 2011 in this case is significant and amounts for nearly as 20% of all CVE reports for that weakness type in the current year. The variance shown in the reports of other vendors is not as much significant, therefore, it can be concluded that Google was in great extent responsible for such increase.

The second weakness from the Table 7 shows a different trend. Although in the current year the products of Google and HP were more vulnerable in that category by more than 50% each, the products of other vendors also demonstrated noticeable growth. Therefore, it can be concluded that the change in this category is rather the result of the global trend.

In the third row, describing the *Code Injection* weakness, the 2010 is used as a reference year for the top three vendors. The cause for that is the fact that in 2011 the distribution among the vendors is relatively equal, e.g. there are no vendors noticed who have relatively higher share in the distribution.

On the contrary, in 2010 the products of several vendors took a considerable share in the distribution. To be precise, Microsoft, Adobe and Apple caused as nearly as 50% of all weaknesses in this category for the year 2010. The sudden decrease for the whole category in the year 2011 can be in significant extent attributed to these vendors.

In the case of *SQL Injection*, which according to the Figure 28 and Table 6 has dropped significantly in 2011, the vendors or products responsible for that change in the large amount could not be found. The Figure 29 shows the distribution of the CVE entry publication related to

⁷¹ The vendor-product relation is derived from the CPE list maintained by Mitre Corporation. It should be noted that some software products do not have the vendor assigned. This is rare case though.

this weakness among the different software products, for the years 2010 and 2011. The similar trend can be spotted for both years, implying that reason in the decrease may be connected to the global trend.

Likewise, Figure 30 shows a similar pattern for the *Information exposure* weakness class. In this case, there has been noticed an increase in 2011, compared to the previous years. In the year 2011 the individual products involved in that trend got greater share.

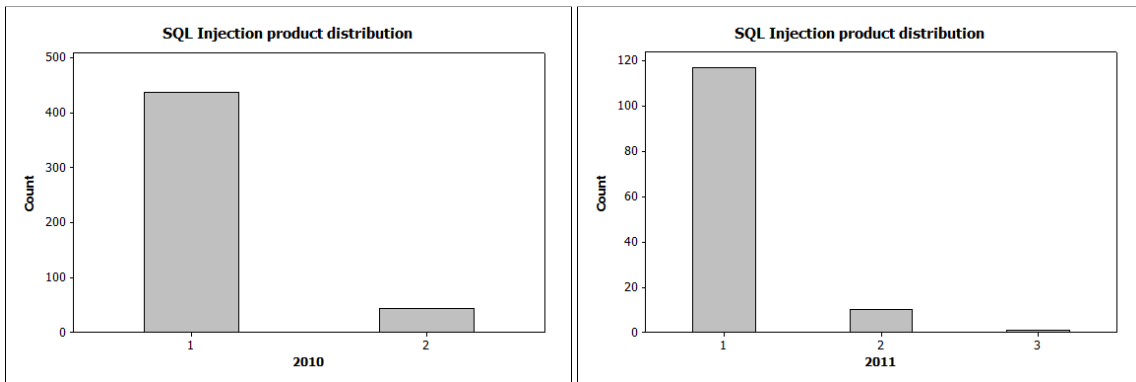


Figure 29: SQL Injection distribution among the products

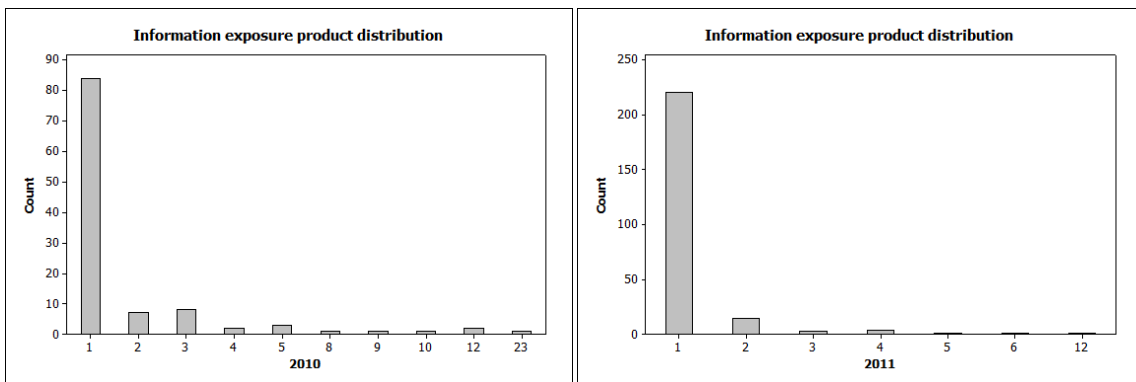


Figure 30: Information exposure weakness among the products

5.3.2 Twitter Chatter Dynamics

The tendencies depicted by Figure 28 and noticed here are based on software vulnerability discovery and reporting in the CVE list. The vulnerabilities in the CVE list are published as they are found (and properly validated). However, the list entries do not bear the information about

general impact and degree of the importance level of a vulnerability, considered from the real-world or user experience aspect.

There may be different factors characterizing the effect or real-world magnitude of a software vulnerability. These effects, or impacts, can be observed or evaluated on different ways, depending on the standpoint used.

For instance, the software vulnerability may be considered in a different way by the diverse categories of observers. Consequently, security professionals – analysts, company CEO/CIO-s, end users or malicious persons can have different priorities or criteria when evaluating the vulnerability.

The emergence of the vulnerability of less popular, narrowly focused business software may be crucial for company CIOs due to the increased risk and costs demanded to prevent losses. For end users, announcement of software vulnerability may be important in an aggregate sense if the software is quite widespread and thus impacts greater range of the user population.

Likewise, one of the parameters considered within the vulnerability assessment may be an audience using the software. Is the software widespread, what determines its user base segmentation characteristics? The software availability to the users, its platform base (server or client side), popularity or exploitability complexity of related vulnerability may also play a role in the perception of its significance.

These criteria and measures are hard to predict using the provided reports only. The user perception and insight on software vulnerabilities could be gained directly from the user interactions or their public statements expressed explicitly or implicitly. This work uses the Twitter platform for the exchange and distribution of real-time short messages to examine user activity related to the software vulnerabilities. It is assumed that this type of user interaction may provide additional information about perception of software vulnerabilities.

Figure 31 illustrates the weekly development of the classes of weaknesses based on the CVE entry publication in official NVD CVE list. These classes of weaknesses are derived from the published CVE entries, which contained the reference to the matching CWE weaknesses.

Figure 32, likewise, shows the relative frequency of weaknesses referred by the CVE entries discussed on Twitter. The time range considered spans through the weeks 21 to 38 of the year

2011, the same during which the Twitter streams have been collected in the database. The sizes of the squares are relative to individual dataset, and not to the combination of both of them.

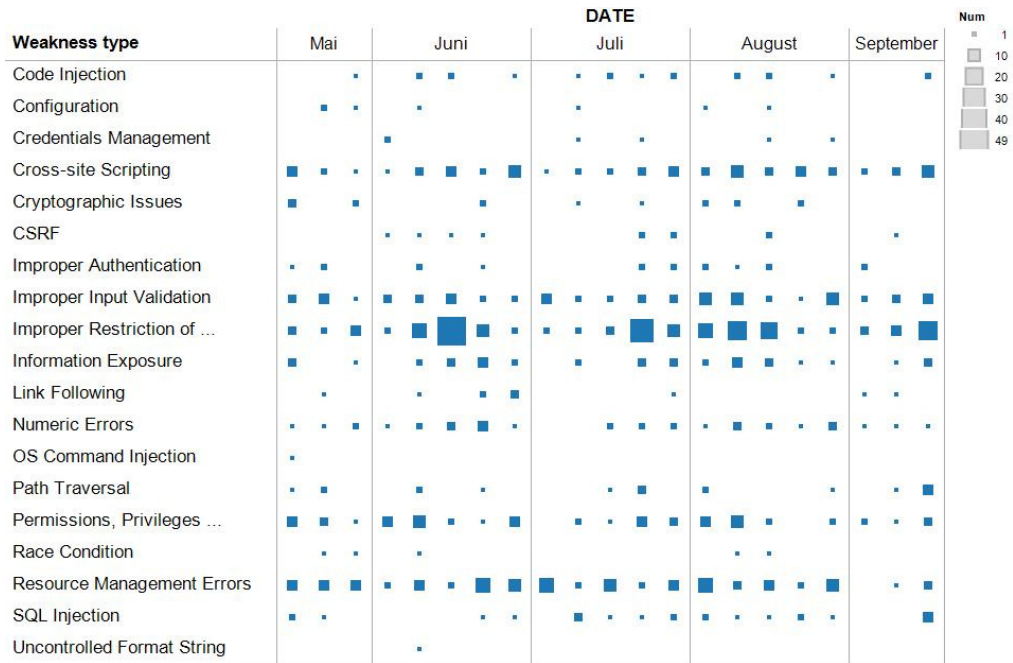


Figure 31: Classes of software weaknesses (CWE) derived from CVE publication

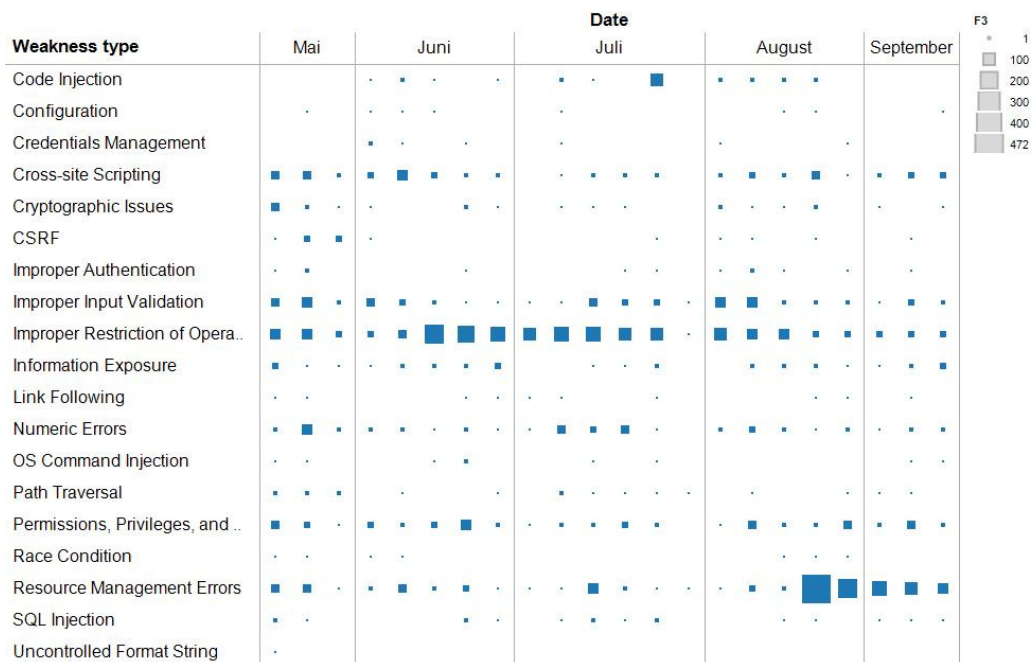


Figure 32: Classes of software weaknesses (CWE) derived from Twitter conversation

The figures show that some categories of the software weaknesses have visibly disparate distributions between datasets among the time ranges. While *Resource Management Errors* are relatively constantly reported in CVE list, the level of their appearing in Twitter stream is relatively low on average, except for a couple of weeks in August and September, when it comprised a significant portion of the messages posted. Similarly, *Code Injection* follows the distribution disproportionately. *Cross-site Scripting* is relatively less commented by Twitter users than it could be expected by its publication dynamics, while the representation of weakness of the type *Improper restriction of Operations within the Bounds of a Memory Buffer* has relatively fluctuating development.

As already presented, Figure 32 depicted the reporting of weaknesses types on the Twitter in the function of the time, in weekly resolution. The next step in the analysis is to provide aggregate information for the whole period considered. In that sense, Figure 33 shows the scatter plot relation of CVE list publication and Twitter stream trends during the same period.

This figure shows unusual values for weaknesses *Improper Restriction of Operations within the Bounds of a Memory Buffer (#9)* and *Resource Management Errors (#17)*. These categories have been assigned to the CVE entries reported on Twitter much more frequently than others, both in CVE list publication and Twitter stream dataset.

The scatter plot also shows that some of the weaknesses do not stand in the approximate field around imaginary line defining Tweet-CVE diagonal. In the case of *Resource Management Errors* or *Code Injection* it means that those weaknesses have been more frequently mentioned in the Tweet dataset. On the other side, it also means that *Cross-site Scripting*, *Permissions, Privileges and Access Controls* and *SQL Injection* in the relative terms have been more accented in the CVE dataset.

Again looking at the same data, Figure 34 shows the relative line plots of both datasets. Figure 35 has been introduced to represent those relationships when the weaknesses #9⁷² and #17 are excluded from consideration⁷³. This adjustment has been done only for the technical reasons and the clarity of representation – to illustrate the interrelation between other types of weaknesses on their scale. The global data as the sum of all messages have been left intact.

⁷² Their absolute position is referred to the order as displayed in Figure 33

⁷³ Combined they represent more than 50% of tweets generated. The purpose of the second diagram is to present visually smoother dependence of other weaknesses types involved.

These figures represent the same trends from the Figure 33 but using a different visual approach.

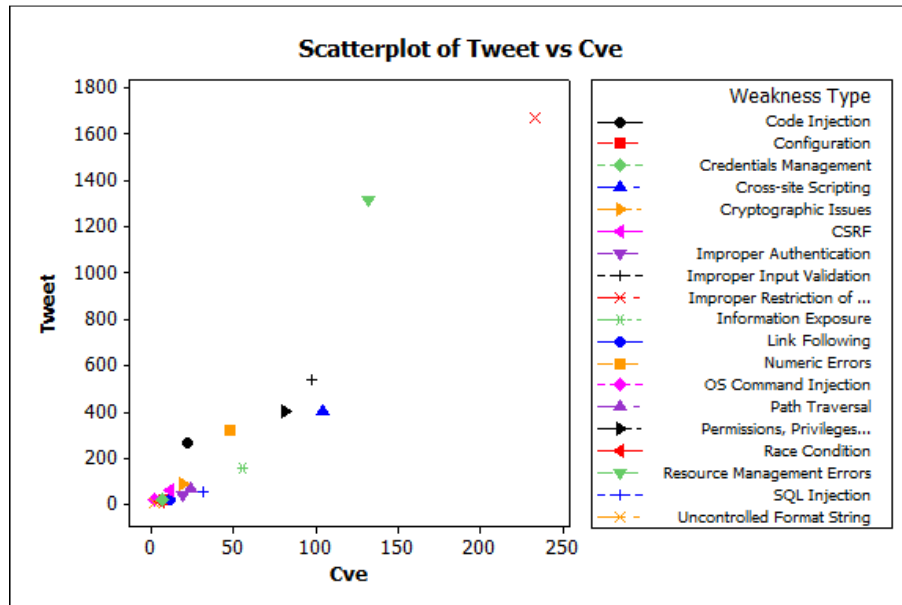


Figure 33: Cumulative ratio of Twitter based reports compared to CVE publication

The bottom parts of the figures display standard deviation tests for both datasets, respectively. It can be read that CVE list in both cases demonstrates relatively lower deviation across the weaknesses types. Given the numbers, mean and standard deviation for CVE dataset amount 47.68 and 59.65 respectively, while the same numbers for Twitter dataset amount 287.63 and 458.61. Also, it can be concluded that the standard deviation of Twitter dataset is significantly different from one from CVE dataset ($P=0,036$)⁷⁴.

The Pearson coefficients for both cases amount $r=0,948$ ($p=0$) and $r=0,915$ ($p=0$), respectively⁷⁵. Both numbers indicate high correlation between CVE and Twitter dataset, which is expected. It would be highly surprising to find that Twitter conversation has a low correlation to the publishing dynamics of the CVE entries. However, these numbers, together with figures accompanied, demonstrate that the Twitter conversation to some extent follows a different pattern.

⁷⁴ Calculation is performed on the complete datasets.

⁷⁵ The Pearson coefficient is used to represent the degree of association between two variables, ranging from -1 to +1. A positive value implies positive association and corresponds to the extent of that association. The numbers presented here describe high level or strong correlation between the datasets.

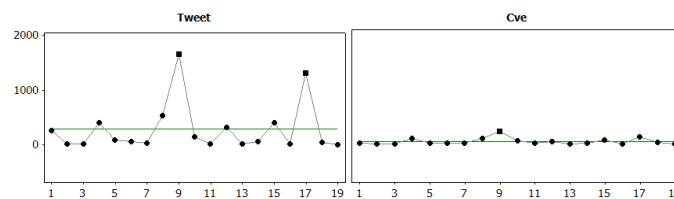
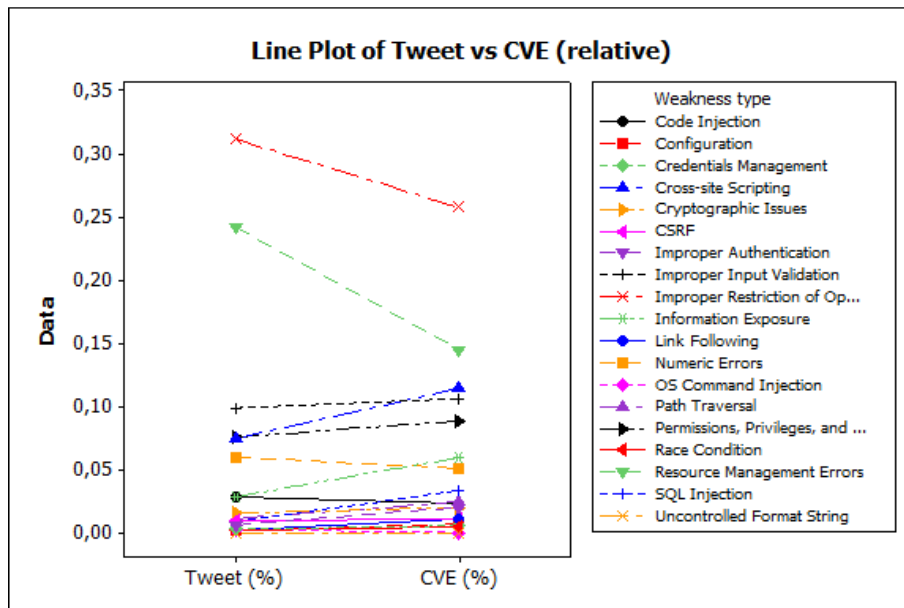


Figure 34: Line plot and standard deviation

In the case presented in the first plot (Figure 34), the weakness type #9 (*Improper Restriction of Operations within the Bounds of a Memory Buffer*) has been clearly more often discussed in the Twitter stream than it has been embedded in the CVE entries published in the same time range. The similar but not so strong trend is demonstrated by the types #1 (*Code Injection*) and #12 (*Numeric Errors*), which can be spotted on Figure 33 too.

The scatter plot on Figure 35 shows stronger tendency also for #13 (*OS Command Injection*). On the other side, #10 (*Information Exposure*), #18 (*SQL Injection*) and #4 (*Cross-Site Scripting*) are under-proportionally included in the Twitter stream as it could be expected based on the amount of their inclusion in CVE list.

Looking back at the Figure 32 and rapid saturation of *Resource Management Errors* in August and September, it can be concluded that Twitter conversation from that period contributes highly to the aggregate results presented in Figure 33 and Figure 34.

Analyzing the Twitter stream for the period beginning from the 3rd week of August, it can be noticed that CVE-2011-3192 had disproportionally higher share in the Twitter conversations. It

amounted for 864 mentions, while the next positioned CVE-2011-1928 and CVE-2011-3190 occurred in the stream only 61 times each.

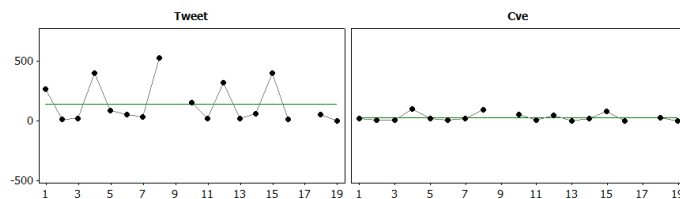
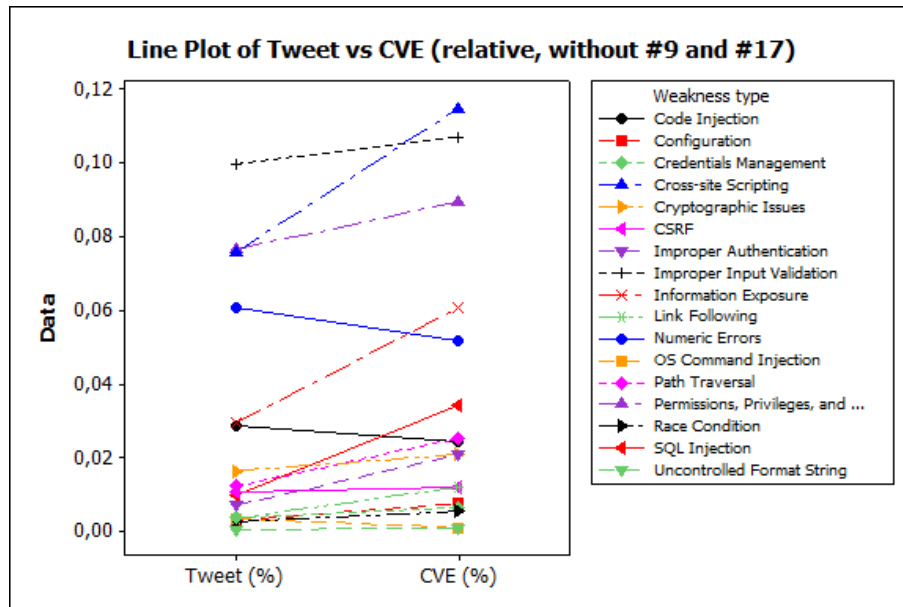


Figure 35: Line plot and standard deviation, without types 9 and 17

Entry in the CVE list identified as CVE-2011-3192, published on August 29th, identifies the software vulnerability found in the range of Apache HTTP Server versions, allowing remote attackers to cause a denial of service (memory and CPU consumption). This vulnerability refers to *Resource Management Errors* (CWE-399) category from the CWE list.

The described case shows that the discovery of one vulnerability in Apache HTTP Server software caused abnormally high impact among the users in terms of message exchange on Twitter. Apache HTTP Server is mature and widely spread product, ran as a server product on many different platforms. As of the October 2011, it owns 64.67% in the web server market share, serving nearly 100 million active web sites worldwide [NETCRAFT11]. The fact that the

software vulnerability in question enables malicious parties to cause remote DoS in such widely used product explains its significant impact among the Twitter's user population.

The second example of a software vulnerability causing relatively higher user impact can be found in the fifth week of July 2011, as shown in Figure 32 under the type *Code Injection*. This is clearly a period in which *Code Injection* type got the highest attention in the Twitter's user community.

From the Twitter stream dataset it can be read that this activity spike has been caused by vulnerabilities identified as CVE-2011-2505 and CVE-2011-2506. They have been mentioned 93 and 92 times respectively. In the observed week, however, the vulnerability ranked as the next one occurred 21 times only. Both of these CVE entries are referred to the same product – phpMyAdmin.

CVE-2011-2505 identifies vulnerability present in the authentication module of the product, which improperly handles input query string. Similarly, CVE-2011-2506 describes the configuration generator which improperly handles the presence of comment closing delimiters submitted by remote users. The amount of the mentions of both entries summed together is more than 8 times higher than the next most commented vulnerability in the time frame considered. As they both relate to the same product, it can be concluded that phpMyAdmin at that time received significant attention by Twitter users. phpMyAdmin is an open source tool written in PHP, used for web based administration of MySQL databases, downloaded in more than 25 million copies [SRFG11].

The last three weeks of June and five weeks of July underwent the high level of activity in the Twitter stream. This can serve as a third example of high user movement caused by the software vulnerability in the CVE list. This activity has been especially evident in CWE's category of *Improper Restriction of Operations within the Bounds of a Memory Buffer*, identified shortly as *CWE-119*.

The period selected is relatively long to consider only one weaknesses type. In these weeks, 1,920 messages containing CVE reference⁷⁶ have been published. Of them, 1,055 messages referred the category *CWE-119*.

⁷⁶ Which further contains reference to a CWE weakness type

Product name	Num. of messages	Weakness	
		CWE-119	351 ⁷⁷
Adobe Flash Player	366	CWE-79	14
		CWE-119	166
Internet Explorer	202	CWE-399	31
		CWE-119	79
PHP	171	CWE-264	86
		CWE-119	98
Windows Server 2008	113	CWE-189	9
		CWE-119	98

Table 8: Distribution of the products and weaknesses types in selected June/July weeks

The distribution of the most mentioned products, sorted by the number of aggregate occurrence in the period is shown in Table 8. This table helps to understand the spike in June and July described here. The last column of the table (*Particular weakness*) lists two most accented weaknesses types from the set of all product mentions in the period observed.

The analysis of the data shows that, in the period analyzed, several products have been affected by CWE-119 weakness type. However, Adobe Flash Player accounted for more than 19% of all traffic and more than 33% of all CWE-119 reflections⁷⁸. Table 8 also shows that the greatest amount of the traffic dedicated to Adobe Flash Player referred to class CWE-119. Therefore, it can be concluded that release of CVE-2011-2110 contributed significantly to the spike apparent in June and July in Figure 32. In this case, again, Twitter users discussed widely popular software product. Many versions of that product and platform-specific implementations have been targeted by that software vulnerability.

5.3.3 Dismantling the Numbers

First part of this section provided the description of the CVE publication dataset from the aspect of underlying CWE weakness type. It illustrated the development of weaknesses types identified by security analysts and professionals from different organizations, companies and institutions

⁷⁷ CVE-2011-2110

⁷⁸ 1,055 Tweets mentioned CWE-119

worldwide, further channelized and published through the CVE publication process. The data gathered covered the development for the years 2009, 2010 and 2011.

That activity identified the tendencies inherent to the discovery and publication dynamics. It has been discovered, and then concluded, that some categories of weaknesses had strong upward or downturn tendencies. In several cases, the reasons behind these tendencies have been identified, and explained. In the case of weakness category *Improper Input Validation*, one software vendor induced the shift in the discovery dynamics for that category.

In the other case concerning *Improper Restriction of Operations within the Bounds of a Memory Buffer*, it has been found that the products of three main software vendors, the known and highly developed companies on the market, took a part in more than 50% of revealed vulnerabilities of a particular weakness category. In the following year, the number of the discovered and reported vulnerabilities of that type, referring the products of those vendors, has significantly decreased.

In some other cases it has been shown that the weaknesses were not been focused around one group of software products or companies. They were rather distributed semi-equally among the many different products. The rise or decline in the total amount of reports per weakness category, therefore, can be rather attributed to the global trend. To analyze the reasons which could define *the global trends* is out of the scope of this work. They may be determined by new technologies, frameworks, languages, libraries or practices used in the software development process, however, there is not enough data to perform an analysis and draw a conclusion.

The second part of this section has been dedicated to the description and investigation of CVE entry publication or conversation dynamics in the Twitter dataset. In other words, it approached the analysis of the weaknesses distribution from the aspect of Twitter users involved in the communication.

The development of related Twitter chatter has been illustrated visually both for the weekly and aggregate data. The differences and similarities of CVE and Twitter datasets have been statistically described, analyzed and commented. Furthermore, the three examples, describing the cases of Apache HTTP Server, phpMyAdmin and Adobe Flash Player, demonstrated the examination of the events noticed in the weekly overview, as depicted in Figure 31 and Figure 32. It has been shown how the disclosure of several vulnerabilities with high impact for the popular product caused noticeable effect on related Twitter traffic.

5.4 Characteristics of Weakness Type

As already stated in the previous sections, some 80% of the CVE entries receive mapping to one of 19 corresponding CWE weakness types. CWE weakness is, depending on the type⁷⁹, further structurally characterized by several underlying parameters. This section presents the relation and distribution of the three groups of these parameters. The bases for comparison are the data sets containing the vulnerabilities mentioned in the users' tweets, and ones published in the CVE database.

Analyzed was the share of elements of each property, both in the Twitter and CVE publication dataset. It has been assumed that the advantage in that share on the Twitter dataset side corresponds to the types of properties and their elements with higher importance or preference for the users. Based on that, the idea of this section is to determine the latent characteristics of the weakness types which have drawn the most interest and activity on the users' side.

In that sense, tracked were *Impact Scope*, *Consequence Scope* and *Time of Introduction*. The corresponding parts of this section explain their meaning and role further. The Figure 22 shows the markup of a Twitter message, illustrating the real-world usage of the three properties presented here.

Each CWE entry may consist of several different parameters of each class. As shown in Appendix 1, the weakness CWE-476 has one parameter in the class *Time of Introduction* and two parameters in class *Common Consequences*. The inclusion of those parameters is optional. This is also demonstrated on the example of CWE-476, which does not have a parameter describing *Impact Scope*.

Figure 36 illustrates the aggregate relation between the *Consequence scope* elements in the Twitter messages and CVE list datasets. The relation is determined by the dynamics of inclusion of the CVE identifiers in the messages, on the one side, and the publication dynamics of the CVE entries in the public CVE list, on the other side. The CVE entries in the both datasets refer to the CWE weaknesses type, which is further characterized by the *Consequence scope* element, among the others.

The *Consequence scope* determines the individual consequence that may be associated with the weakness. This field identifies six elements: *Confidentiality*, *Integrity*, *Availability*, *Access*

⁷⁹ Category, compound type or weakness

Control, *Non-Repudiation* and *Other*. Each weakness type may have one or more of those elements, referring to the aspects of information security affected by the weakness.

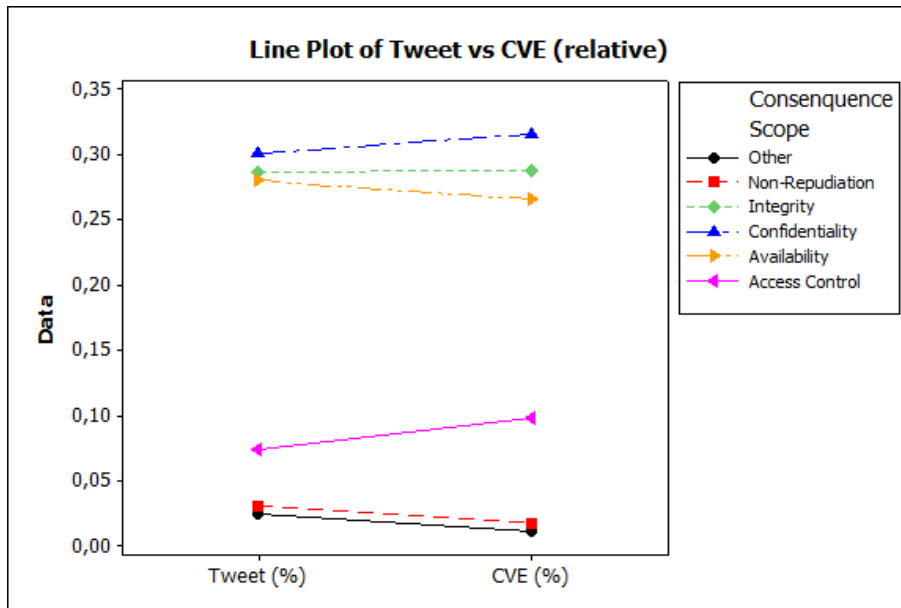


Figure 36: Line plot describing the development of *Consequence Scope*

As the Figure 36 shows, the CVE entries containing the properties such as *Availability*, *Non-Repudiation* and *Other* have been relatively more popular among the Twitter users, while the vulnerabilities affecting *Integrity* exhibited no change in the Twitter users' mentions. The scope of *Access Control* received comparatively 25% less interest in the Twitter stream, compared to the publication dynamics in the CVE list.

The other characteristic represented by the Figure 37 is the *Impact Scope*. The impact scope determines the exact classes of the impact of specific CWE weakness type. It may contain one or more of these classes. There are 16 different classes under these characteristics: *Read memory*, *Read files or directories*, *Read application data*, *Other*, *Modify memory*, *Modify file or directories*, *Modify application data*, *DoS: crash/exit/restart*, *Hide activities*, *Gain privileges/assume identity*, *Execute unauthorized code*, *DoS: resource consumption other*, *DoS: resource consumption memory*, *DoS: resource consumption CPU*, *DoS: instability*, *DoS: crash/exit/restart* and *Bypass protection mechanism*.

Figure 37 draws the aggregate relation of the specific CWE weakness type in the Twitter and CVE datasets, in the term of its underlying impact scope. Similarly as in the previous example,

the relation is determined by the frequency of Twitter mentions of CVE identifiers and CVE publication list dynamics.

Some of the impact scopes from the Figure 37 can be categorized in one of two groups. Stronger emphasis in the Twitter stream received categories such as DoS: crash/exit/restart, DoS: resource consumption (memory), DoS: resource consumption (CPU), *Other*, *Hide activities*, and *Modify application data*. This group is more often commented in the Twitter than it is in CVE dataset. Some of its members got 15% or even double relative increase.

The second group characterizes the impact scope categories which received relatively less interest among the Twitter users. These are: *Read application data*, *Execute unauthorized code and commands*, *Bypass protection mechanism* and *Read files or directories*. This group represents the impact scopes which occurred relatively less often among the Twitter population, than it would be expected based on the CVE publication dynamics. The members of this group received approximately 20-50% decrease in the Twitter dataset compared to CVE publication frequency.

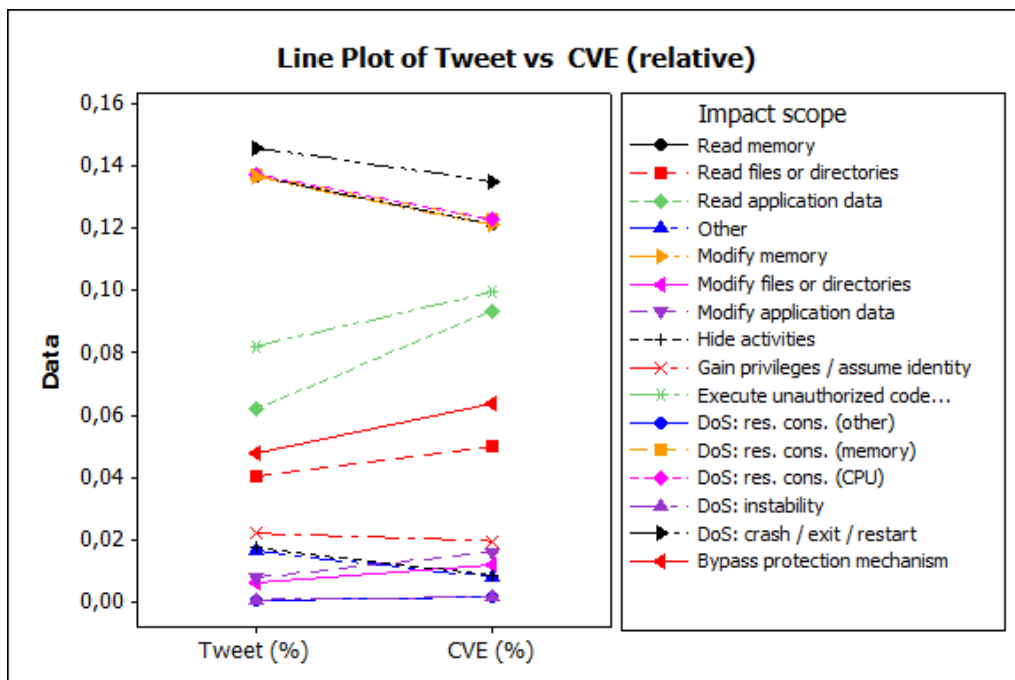


Figure 37: Line plot describing the development of *Impact Scope*

The Figure 37 shows graphically shows the difference in the relative share of impact scopes from each of the groups mentioned here. The vulnerabilities having the latent characteristics from the first group of the impact scopes have been comparatively more discussed and forwarded by the Twitter users. That group is dominated by the entries from the DoS and data manipulation families. This information can be confirmed from the example from Section 3, which discussed Apache HTTP Server vulnerability.

Based on that, the users demonstrated higher interest the vulnerabilities which impact was distributed on the large scale. On the other side, the vulnerabilities based around the impact scope described by the second group have been of less interest, importance or significance for the Twitter users.

The third property of weakness type is *Time of introduction*. This characteristic describes the time of introduction of the software weaknesses in the software development process. It can be any of the three phases of the software development: *Architecture and Design*, *Implementation* or *Operation*.

Figure 38 draws the relationship of *Time of introduction* between the Twitter stream dataset and the dataset representing the CVE publication dynamics. This plot shows that *Architecture and Design* and *Implementation* have almost the same share and development in the both datasets. Their exact numbers differ about 1% in the Twitter stream dataset.

This correlation seemed strange at beginning, but after a more detailed look at the entries from *CWE List View 635*, it could be confirmed that these two phases occur very often together among the CWE entries. That fact raises the question about the adequacy of categorization selected for this property.

Besides that, from the two groups, the vulnerabilities created in the *Operation* phase of software development had relative increase of approximately 17% in the Twitter dataset. That implies that vulnerabilities from the later phases of the software development have been a slightly more interesting and popular among the user population.

The next finding brings to the conclusion that the most of discovered software vulnerabilities have been created in the phases of *Architecture and Design* and *Implementation*, sharing nearly the same probability among those phases.

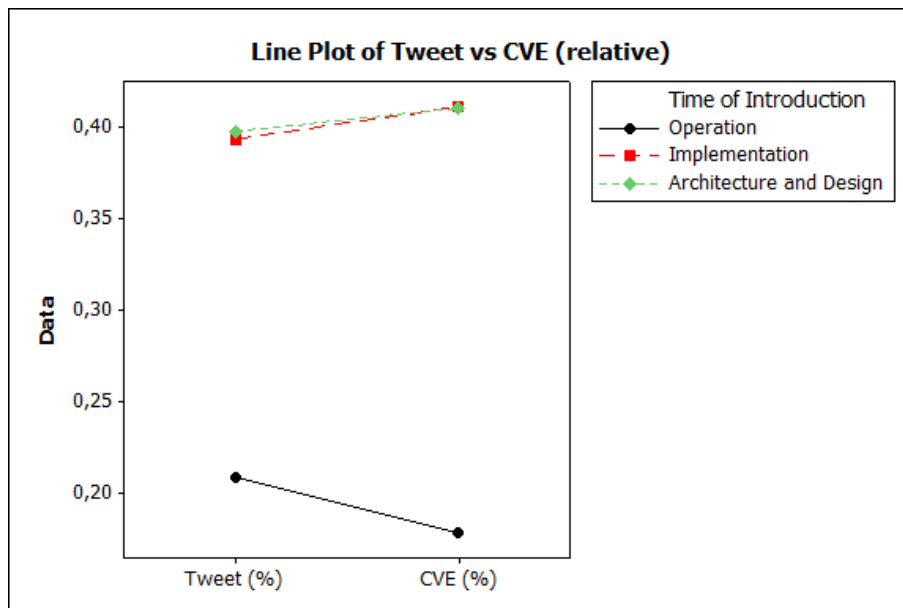


Figure 38: Time of introduction

The interpretation of comparisons performed in this section is that the software weaknesses drawing the significant part of the attention of the Twitter users have been producing consequences in the areas of *Availability*, *Non-Repudiation* and *Other*. Furthermore, considering the other category, the weaknesses whose impacts have been based on data modification operations or some denial of service types have been causing the higher level of interest among the users. Lastly, the data analyzed shows that *Operation* phase of the software development process produces almost 20% of all software weaknesses.

5.5 CVE Entry Distribution in the Term of Publication Time

The value of the Twitter information and the degree of its quality been mentioned in the Section 3.2 in Chapter 2 [TWRESP09]. Similarly, the idea behind the analysis presented in this section is to discover and describe some of the properties of Twitter communication in the domain of the software vulnerability information management.

The first crucial task is to discover how the new information propagates on Twitter, in the temporal sense. What time frame defines the critical period before and after the publication and modification of a CVE entry? In order to assess such information, the first necessary step is to investigate what kinds of data are provided by the official information sources.

The life cycle of a CVE entry begins with its candidate status. The software vulnerabilities, upon their first submission, get assigned the candidate status. Based on the predefined procedures, the submissions are further disseminated, reviewed and evaluated. That process involves several phases through which a CVE candidate passes.

After its final acceptance by the decision board, the candidate gets converted into an appropriate entry and announced on the CVE list. With this step, the candidate becomes the CVE entry officially. This is the moment when the wider public, interested parties and stakeholders get informed about the vulnerability formally, through a public announcement.

Usually, such announcement results in a public attention toward the related entry; it may affect the security bulletins, reports and recommendations worldwide. Some announcements may be further distributed by the media, with higher or lower level of interest. The level of public awareness about the vulnerability may depend on its significance, which may further depend on the perception of vulnerability on the side of the persons or bodies deciding about information delivery process.

The structure of an entry in the CVE list provides, among the other information, its time of publication and the time of last modification.

The time of the last modification provides information about the last update of a CVE entry. Usually, the most of modifications relate to the changes of descriptions, updates of the references or other smaller adjustments. Sometimes, more substantial changes are completed. These can be based on splitting of an entry into separate entries, or its merging with another one.

The time of publication of the entry provides not the time of its initial creation, but of its conversion from candidate to entry status. The CVE entries are initially announced as candidates. After the predefined review process is done, they are further converted to the entries and included in the official CVE list. Therefore, the temporal information about the status of an entry in its earlier lifecycle stages is not provided in the CVE list directly.

This section analyzes the effect of the relative time of the publication and modification of a CVE entry on its circulation in the Twitter message stream. The figures presented here describe the relation of the time difference between publication and modification of a CVE entry and its discussion and related activity level on Twitter. The time difference factor is determined as the time of Twitter message broadcast reduced by the time of publication e.g. last modification of a CVE entry⁸⁰.

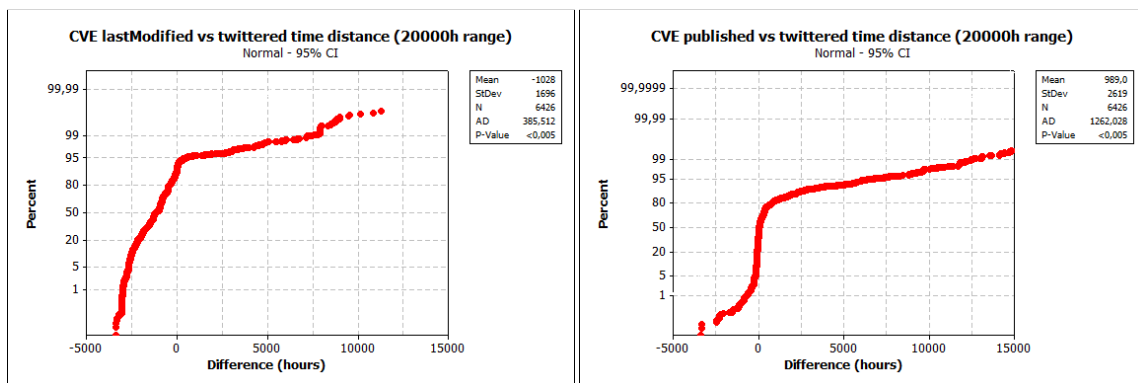


Figure 39: Comparison between CVE last modified and published time

Figure 39 describes the typical Twitter activity level related to the CVE entries in the function of time. The left part of the figure presents the activity distribution relative to the last modification time of a CVE entry. The right part of the figure, similarly, presents such relation relative to the publication time of a CVE entry. Figure 39 presents the distribution in the time range of 20,000 hours, where the central points define the last modification and publication times.

For the time of publication, it can be observed that some 36 percent of all mentions of CVE candidates refers to the negative time. Hence, the important part of all CVE mentions relates to the conversation held during their *candidate* phase, before they get published as the CVE entries.

Figure 40 further disseminates and presents the distribution depicted on Figure 39, based on the relative time periods. This makes possible to get deeper insight into the developments

⁸⁰ The times are handled in the form of the timestamp, which can be in later stage converted into the other representation type, if appropriate

related to specific and critical time periods. The figure refers to the CVE publication time, which is primarily considered.

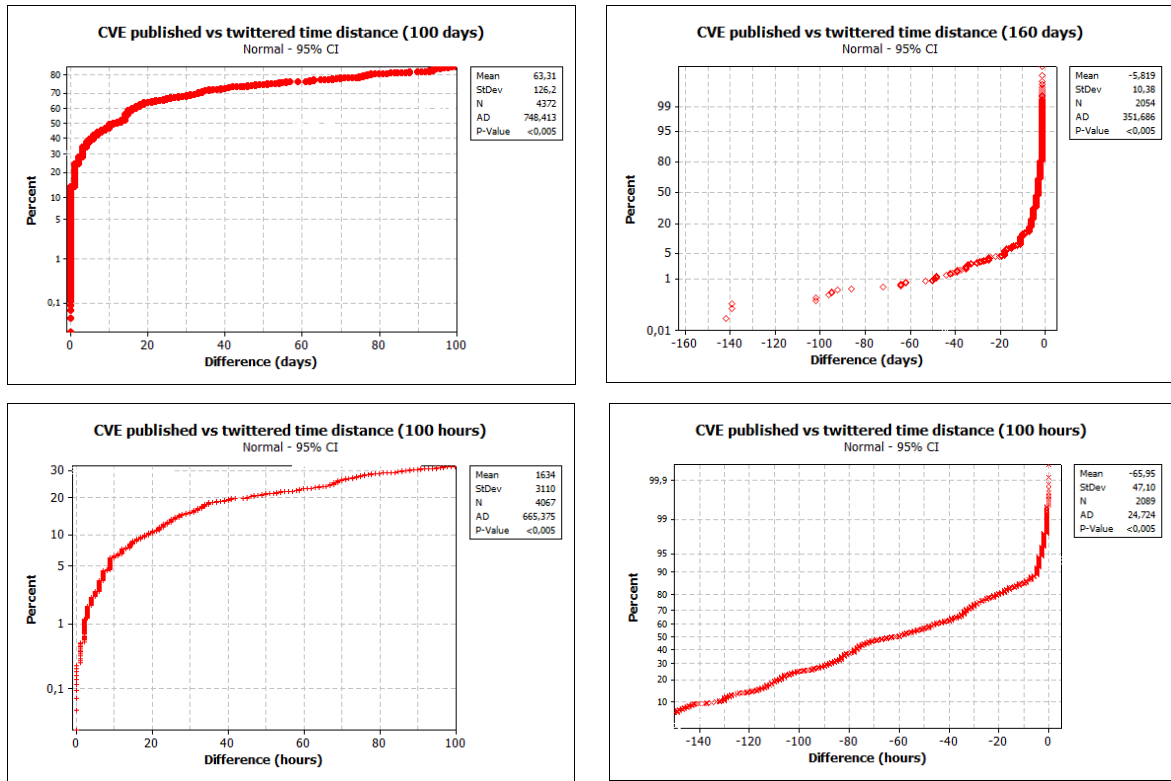


Figure 40: Difference between CVE publication time and timestamp of the Twitter message

The time frames measured in the Figure 40 are periods of 160 days before the publication, 100 days after the publication, as well as 100 hours before and after the publication. Unlike the Figure 39, the values presented in the Figure 40 refer to the data available to the one of two statuses only. Accordingly, the distribution for the positive time range covers only the status of the CVE entry.

Likewise, the plots presenting the negative time range represent the distribution of the messages only for the status of the CVE candidate. As explained before, in the period before its official publication, the CVE entry finds itself in the candidate status. The time after the publication refers to the official CVE entry status.

The top part of the figure depicts the relation for the positive⁸¹ and the negative⁸² part of the timeline. Similarly, the bottom part of the figure depicts the same relation, which is provided in the scale on the hourly level and covers a few days relative at the entry publication time.

The plots on the left side show that around 13% of all related Twitter conversations in the entry status happened by the first day after publication time. Furthermore, the 30% are reached after 4 days, while half of all the conversation referring some entry is done during the first 10 days.

The CVE candidate can be considered as software vulnerability in the phase of its investigation and status decision. As a result, in this phase the candidate is more relevant for computer security and professional community, than it is for the public. After its acceptance, confirmation and conversion into the CVE entry, it becomes “more open” and probably of interest to the wider community.

The right part of Figure 40 describes the conversation in the phase before the CVE entry is published in the CVE list, e.g. during the time it is considered as a candidate. It is not uncommon that some vulnerabilities stay on the candidate list for a couple of months – the discussion in that direction is visible on the top right plot. However, as the publication date gets closer, the communication intensifies. The result is that some 50% of related conversation happens during the last three days prior to publication.

Figure 39 shows the similar relation, but for the last modification date of a CVE entry. It should be noted that CVE candidate can be converted to the CVE entry only once. For this reason, the publication time is generally set only once, without later changes. The time of the last modification, however, can be changed several times in the lifecycle of a CVE entry.

The plot on the left part of Figure 39 suggests that some 91% of all conversations referring an individual CVE entry occur before its last modification time. However, that information should be taken with the reserve, as the moment of an analysis of the dataset followed several weeks after the final data sample has been taken.

This section presented the findings related to the distribution of the CVE references in the Twitter stream in relation to their publication and last modification date. It has been demonstrated that some 91% of CVE entries are not updated in the period following their occurrence in the Twitter stream.

⁸¹ Discussion in entry status

⁸² Discussion as candidate

The fact that around 30% of all CVE entry mentions on the Twitter happen during the first four days shows that, on average, the CVE entries on Twitter propagate more slowly than it could be expected. However, that fact may be useful when constructing the real time significance measurement and recognition tool for the vulnerabilities. The entities broadcasted on a faster rate could be candidates for a further investigation.

On the other side, the related Twitter activity increases rapidly as the CVE candidate approaches official publication event, covering 50% of activity in the last three days prior to the publication. This information may be useful to predict those event types timely.

6 Conclusion and Further Directions

As the online social networks represent a relatively new approach in the areas of information sharing and online collaboration, the goal of this work was to explore their potential usage in the field of software vulnerability identification and tracking.

Online social networks are still a rapidly growing and varying concept. They are changing and reshaping other types of services on the Internet, affecting the paradigm of the Internet and its presence in the personal and business world. In many domains there are opportunities to investigate potentials for research and practical application of the OSNs. At the time when this thesis proposal was prepared, the application of OSNs in software vulnerability research has not been largely explored. From that point, this work represents a contribution in that direction.

The first task of the work was to perform the literature survey. The second and third chapters are dedicated to this segment of activities. The second chapter investigated the general area of OSNs, with the focus on Twitter. Twitter concentrates on single, but a powerful concept of the real time exchange of short messages. The third chapter explored the area of software vulnerabilities. The taxonomy systems in the field were surveyed, and the problems of different approaches were presented.

In the second task, which is based on the findings from the previous survey, the software framework was constructed and employed in the work. This software framework has been used to monitor and gather data from Twitter during a couple of months. Afterwards, the framework processed the gathered data and prepared it for further analyses. Chapter 4 presents that framework, the external data sources and the basics on the data sets gathered.

Once the data in the form of Twitter messages and their descriptions have been gathered and organized, they have been processed, augmented and enriched by the software framework accordingly. The idea behind that was to recognize the entities in the Twitter messages and

structure their meaning through the usage of semantic technologies and data from external sources. That has been done, and the results of the analyses were presented in Chapter 5.

It should be mentioned that both the software framework and the research done had to be accommodated to the scope of this work. Therefore, there is still a potential present to extend the software framework, introduce new approaches and include new data sources. On the research side, even the current basis holds enough data for the extensible investigation.

The result of this work is a database of tweets covering the period from the May 15th to the September 17th 2011, occupying some 2.5 GB of disk storage. The data analysis involved two main approaches. The first one, presented in the sections 1 and 2 in Chapter 5, is based on the keywords representing the software vendors, products, or identifiers of software security related bulletins and advisories. This approach proved the possibility to get timely information about the unknown vulnerability of tremendously popular and spread software.

The spike displayed in Figure 25 demonstrated how the occurrence of the exploit for the popular software drastically increased the related activity on Twitter. Such approach could be used to detect the security breaches in the known software and inform users and system administrators a shortly after the detection of the incident.

The second approach in the analysis of the gathered messages was to recognize the identifiers of the CVE list and employ their descriptors to mine underlying information and trends further. This approach can be applied to gather periodical information about the trends and hot topics in the software vulnerability industry. They can be related to vendors, products, programming languages, development approaches or any other information which is provided in the underlying and related structured resources.

Another idea used in this direction has been to approach already disclosed software vulnerabilities from the user's perspective. As it has been shown in Chapter 4 and Chapter 5, the assessment of scopes of CVE identifiers is usually being done by the security professionals. Using CVSS, they are subjectively evaluating and determining the impact and criticality of a vulnerability.

On the other side, however, it could be useful to assess the impact of the vulnerability from the aggregate user point of view. With the practical example, Section 3 demonstrated that users do

have preferences and that their global voice could be used to assess and predict the potential impact of vulnerabilities.

During the work on this thesis I found issues in the several areas:

1) Security bulletins, advisories, enumeration lists

Some of the public sources do not provide structured and easily obtainable information to other users. These are, among others, MS (Microsoft) and ZDI (Zero Day Initiative). There exist a number of companies providing specialized packages and solutions for the software vulnerability management. They monitor the software and publish advisories; however, they do not provide their data in the form structured for the interchange, ingestion and further integration. Although Mitre and NVD tried to approach the field from the structured and cooperative perspective, not all organizations are trying to follow this approach. The resulting situation is not optimal in the term of effort invested and results produced globally.

2) Naming and classification of the software packages

Although Mitre approached this question with their CPE, the solution is still far from the perfect one. OSVDB follows their approach, but it also has comparable issues and too many similarities. The problem with CPE is that the structure of their database and naming schema are not optimal. There are too many redundancies. There are also other problems related to the data change in the function of time. For instance, how to approach the acquisitions and alterations of the vendor names, or how to define the vendors of public domain software or software which has been authored by several physical persons?

Experimenting on the Freshmeat database enlightened one other problem, which is the ambiguity in the software package and vendor names. In the open source area especially, there are many software packages having the names based on the common English words. Many of those packages are in an immature development phase or are not used widely. The successful inclusion of such data requires a specially developed technique of product name disambiguation in the messages.

This technique should be developed also in the other cases of ambiguities, for instance, in the cases where the incomplete program names were published. Recognizing separate versions, editions or updates of products was out of the scope of this work – these also pose new challenges.

3) CVE, CWE issues

CWE represents a comprehensive list of software weaknesses. However, in the CVE database only 19 of them are recognized and assigned to the vulnerabilities found. There is a rate of about 20% of the vulnerabilities not receiving the CWE weakness classification. It seems unlikely that only 19 of all the weaknesses types fit the CVE list, especially as the field is always advancing and new types of weaknesses are discovered.

Another potential problem with CWE is related to the classification of the underlying properties of the weakness. The strong correlations of some values⁸³ indicate that either the classification system of the subcategory is not optimal, or the quality of the process of manual classification alone should be reviewed.

The following suggestions apply to the further work:

1) The semantic relation in the areas of naming and classification of software packages and vendors should be further defined and improved. The same applies to different systems for vulnerability enumeration and classification. The proposals of Howison, Berger et al. [HOWISON08, SEMANTICOSS10] in the area of OSS could be extended or used as starting points in this sense.

2) As already mentioned, the further development of the application and vendor name disambiguation techniques would be of use in the further work. That especially applies to the area of commercial software, as the data for OSS software are generally available in relatively accessible forms. However, the data and representations are not completely consistent among the different OSS repositories. There is an ongoing work in that direction [HOWISON08].













⁸³ For instance, Figure 38 shows that *Implementation* and *Architecture and Design* have nearly the same values.

Appendix 1: Example of the CWE List Entry

CWE-476: NULL Pointer Dereference	
Weakness ID: 476 (Weakness Base)	
Description	
Description Summary A NULL pointer dereference occurs when the application dereferences a pointer that it expects to be valid, but is NULL, typically causing a crash or exit.	
Extended Description NULL pointer dereference issues can occur through a number of flaws, including race conditions, and simple programming omissions.	
Time of Introduction	
<input type="checkbox"/> Implementation	
Applicable Platforms	
Languages	
<input type="checkbox"/> C <input type="checkbox"/> C++ <input type="checkbox"/> Java <input type="checkbox"/> .NET	
Common Consequences	
Scope	Effect
Availability	<u>Technical Impact:</u> DoS: crash / exit / restart NULL pointer dereferences usually result in the failure of the process unless exception handling (on some platforms) is available and implemented. Even when exception handling is being used, it can still be very difficult to return the software to a safe state of operation.
Integrity Confidentiality Availability	<u>Technical Impact:</u> Execute unauthorized code or commands In very rare circumstances and environments, code execution is possible.
Likelihood of Exploit	
Medium	
Detection Methods	
Automated Dynamic Analysis	This weakness can be detected using dynamic tools and techniques that interact with the software using large test suites with many diverse inputs, such as fuzz testing (fuzzing), robustness testing, and fault injection. The software's operation may slow down, but it should not become unstable, crash, or generate incorrect results. Effectiveness: Moderate
Manual Dynamic Analysis	Identify error conditions that are not likely to occur during normal usage and trigger them. For example, run the program under low memory conditions, run with insufficient privileges or permissions, interrupt a transaction before it is completed, or disable connectivity to basic network services such as DNS. Monitor the software for any unexpected behavior. If you trigger an unhandled exception or similar error that was discovered and handled by the application's

	environment, it may still indicate unexpected conditions that were not handled by the application itself.
Demonstrative Examples	
<p>Example 1</p> <p>While there are no complete fixes aside from conscientious programming, the following steps will go a long way to ensure that NULL pointer dereferences do not occur.</p> <pre data-bbox="220 510 574 616"> if (pointer1 != NULL) { /* make use of pointer1 */ /* ... */ } </pre> <p>If you are working with a multithreaded or otherwise asynchronous environment, ensure that proper locking APIs are used to lock before the if statement; and unlock when it has finished.</p>	
<p>Example 2</p> <p>This example takes an IP address from a user, verifies that it is well formed and then looks up the hostname and copies it into a buffer.</p> <p>(Bad Code)</p> <p>Example Language: C</p> <pre data-bbox="220 922 1340 1227"> void host_lookup(char *user_supplied_addr) { struct hostent *hp; in_addr_t *addr; char hostname[64]; in_addr_t inet_addr(const char *cp); /*routine that ensures user_supplied_addr is in the right format for conversion */ validate_addr_form(user_supplied_addr); addr = inet_addr(user_supplied_addr); hp = gethostbyaddr(addr, sizeof(struct in_addr), AF_INET); strcpy(hostname, hp->h_name); } </pre> <p>If an attacker provides an address that appears to be well-formed, but the address does not resolve to a hostname, then the call to gethostbyaddr() will return NULL. Since the code does not check the return value from gethostbyaddr (CWE-252), a NULL pointer dereference would then occur in the call to strcpy().</p> <p>Note that this example is also vulnerable to a buffer overflow (see CWE-119).</p>	
<p>Example 3</p> <p>In the following code, the programmer assumes that the system always has a property named "cmd" defined. If an attacker can control the program's environment so that "cmd" is not defined, the program throws a NULL pointer exception when it attempts to call the trim() method.</p> <p>(Bad Code)</p> <p>Example Language: Java</p> <pre data-bbox="220 1688 750 1738"> String cmd = System.getProperty("cmd"); cmd = cmd.trim(); </pre>	
Observed Examples	
Reference	Description
CVE-2005-3274	race condition causes a table to be corrupted if a timer activates while it is being modified, leading to resultant NULL dereference; also involves locking.
CVE-2002-1912	large number of packets leads to NULL dereference

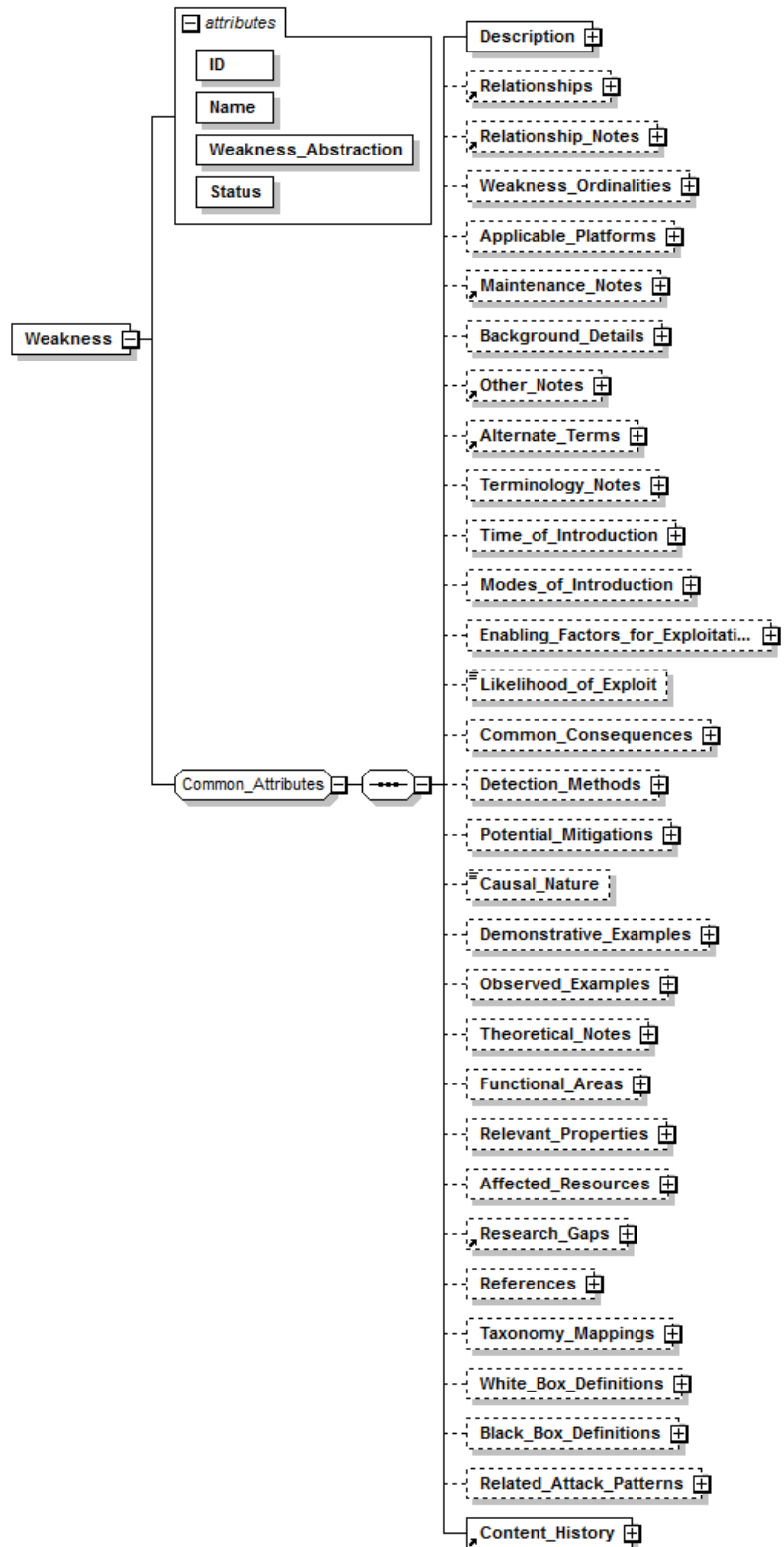
CVE-2005-0772	packet with invalid error status value triggers NULL dereference
CVE-2009-4895	chain: race condition for an argument value, possibly resulting in NULL dereference
CVE-2009-3547	chain: race condition might allow resource to be released before operating on it, leading to NULL dereference
CVE-2009-3620	chain: some unprivileged ioctl's do not verify that a structure has been initialized before invocation, leading to NULL dereference
CVE-2009-2698	chain: IP and UDP layers each track the same value with different mechanisms that can get out of sync, possibly resulting in a NULL dereference
CVE-2009-2692	chain: uninitialized function pointers can be dereferenced allowing code execution
CVE-2009-0949	chain: improper initialization of memory can lead to NULL dereference
CVE-2008-3597	chain: game server can access player data structures before initialization has happened leading to NULL dereference
CVE-2008-5183	chain: unchecked return value can lead to NULL dereference
CVE-2004-0079	
CVE-2004-0365	
CVE-2003-1013	
CVE-2003-1000	
CVE-2004-0389	
CVE-2004-0119	
CVE-2004-0458	
CVE-2002-0401	
Potential Mitigations	
Phase: Implementation	If all pointers that could have been modified are sanity-checked previous to use, nearly all NULL pointer dereferences can be prevented.
Phase: Requirements	The choice could be made to use a language that is not susceptible to these issues.
Phase: Implementation	Check the results of all functions that return a value and verify that the value is non-null before acting upon it. Effectiveness: Moderate Checking the return value of the function will typically be sufficient, however beware of race conditions (CWE-362) in a concurrent environment. This solution does not handle the use of improperly initialized variables (CWE-665).
Phase: Architecture and Design	Identify all variables and data stores that receive information from external sources, and apply input validation to make sure that they are only initialized to expected values.
Phase: Implementation	Explicitly initialize all your variables and other data stores, either during declaration or just before the first usage.
Phase: Testing	Use automated static analysis tools that target this type of weakness. Many modern techniques use data flow analysis to minimize the number of false positives. This is not a perfect solution, since 100% accuracy and coverage are not feasible.
Weakness Ordinalities	
Ordinality	Description
Resultant	NULL pointer dereferences are frequently resultant from rarely encountered error conditions, since these are most likely to escape detection during the testing phases.

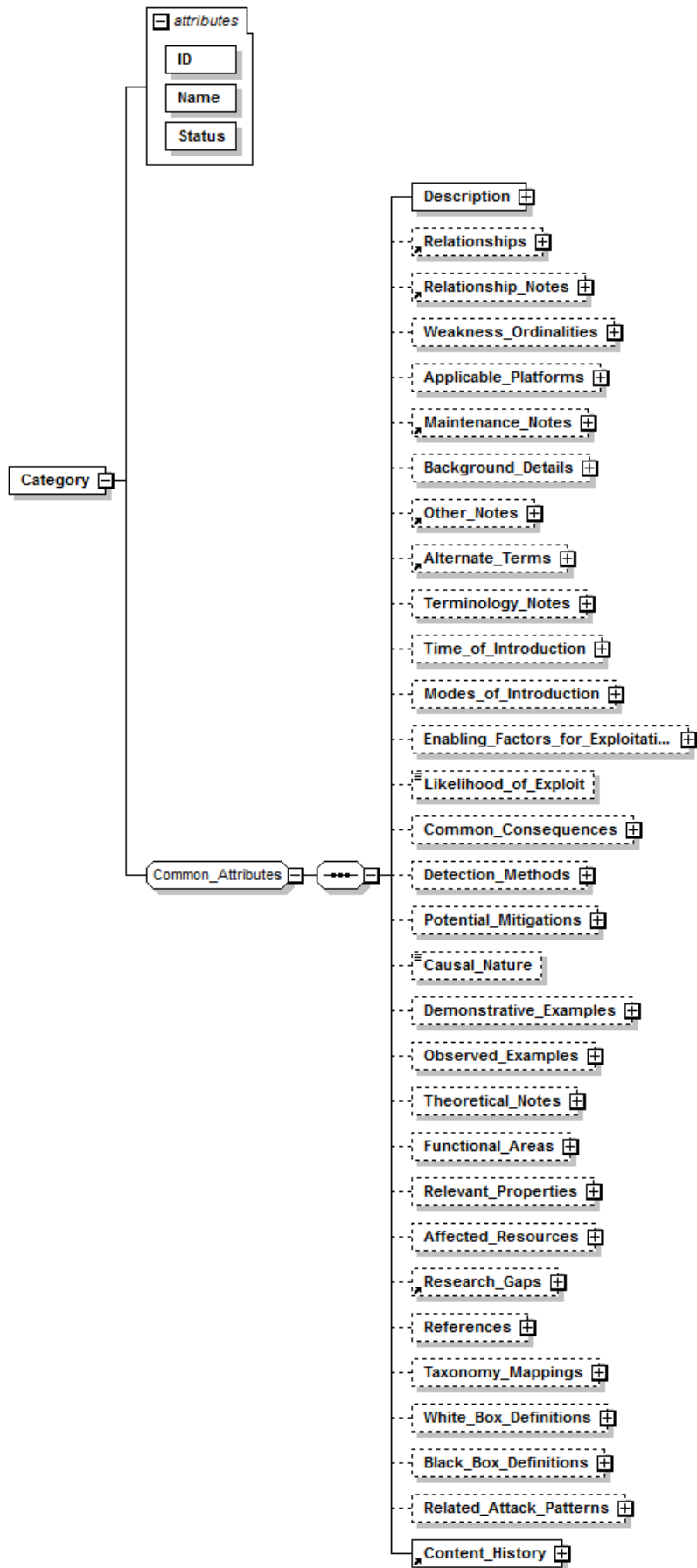
Relationships					
Nature	Type	ID	Name	View	N. Chain
ChildOf		398	Indicator of Poor Code Quality	699 700 1000	
ChildOf		465	Pointer Issues	700	
ChildOf		730	OWASP Top Ten 2004 Category A9 - Denial of Service	711	
ChildOf		737	CERT C Secure Coding Section 03 - Expressions (EXP)	734	
ChildOf		742	CERT C Secure Coding Section 08 - Memory Management (MEM)	734	
ChildOf		808	2010 Top 25 - Weaknesses On the Cusp	800	
ChildOf		867	2011 Top 25 - Weaknesses On the Cusp	900	
ChildOf		871	CERT C++ Secure Coding Section 03 - Expressions (EXP)	868	
ChildOf		876	CERT C++ Secure Coding Section 08 - Memory Management (MEM)	868	
MemberOf		630	Weaknesses Examined by SAMATE	630	
CanFollow		252	Unchecked Return Value	1000	690
CanFollow		789	Uncontrolled Memory Allocation	1000	
Taxonomy Mappings					
Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name		
7 Pernicious Kingdoms			Null Dereference		
CLASP			Null-pointer dereference		
PLOVER			Null Dereference (Null Pointer Dereference)		
OWASP Top Ten 2004	A9	CWE_More_Specific	Denial of Service		
CERT C Secure Coding	EXP34-C		Ensure a null pointer is not dereferenced		
CERT C Secure Coding	MEM32-C		Detect and handle memory allocation errors		
CERT C++ Secure Coding	EXP34-CPP		Ensure a null pointer is not dereferenced		
CERT C++ Secure Coding	MEM32-CPP		Detect and handle memory allocation errors		
Related Attack Patterns					
CAPEC-ID	Attack Pattern Name		(CAPEC Version: 1.7)		
54	Probing an Application Through Targeting its Error Reporting				
28	Fuzzing				
129	Pointer Attack				
White Box Definitions					
A weakness where the code path has:					
1. start statement that assigns a null value to the pointer					
2. end statement that dereferences a pointer					
3. the code path does not contain any other statement that assigns value to the pointer					
Content History					

Submissions			
Submission Date	Submitter	Organization	Source
	7 Pernicious Kingdoms		Externally Mined
Modifications			
Modification Date	Modifier	Organization	Source
2008-07-01	Eric Dalci updated Time_of_Introduction	Cigital	External
2008-08-01	added/updated white box definitions	KDM Analytics	External
2008-09-08	CWE Content Team updated Applicable_Platforms, Common_Consequences, Relationships, Other_Notes, Taxonomy_Mappings, Weakness_Ordinalities	MITRE	Internal
2008-11-24	CWE Content Team updated Relationships, Taxonomy_Mappings	MITRE	Internal
2009-05-27	CWE Content Team updated Demonstrative_Examples	MITRE	Internal
2009-10-29	CWE Content Team updated Relationships	MITRE	Internal
2009-12-28	CWE Content Team updated Common_Consequences, Demonstrative_Examples, Other_Notes, Potential_Mitigations, Weakness_Ordinalities	MITRE	Internal
2010-02-16	CWE Content Team updated Potential_Mitigations, Relationships	MITRE	Internal
2010-06-21	CWE Content Team updated Demonstrative_Examples, Description, Detection_Factors, Potential_Mitigations	MITRE	Internal
2010-09-27	CWE Content Team updated Demonstrative_Examples, Observed_Examples, Relationships	MITRE	Internal
2010-12-13	CWE Content Team updated Relationships	MITRE	Internal
2011-06-01	CWE Content Team updated Common_Consequences	MITRE	Internal
2011-06-27	CWE Content Team updated Related_Attack_Patterns, Relationships	MITRE	Internal
2011-09-13	CWE Content Team updated Relationships, Taxonomy_Mappings	MITRE	Internal

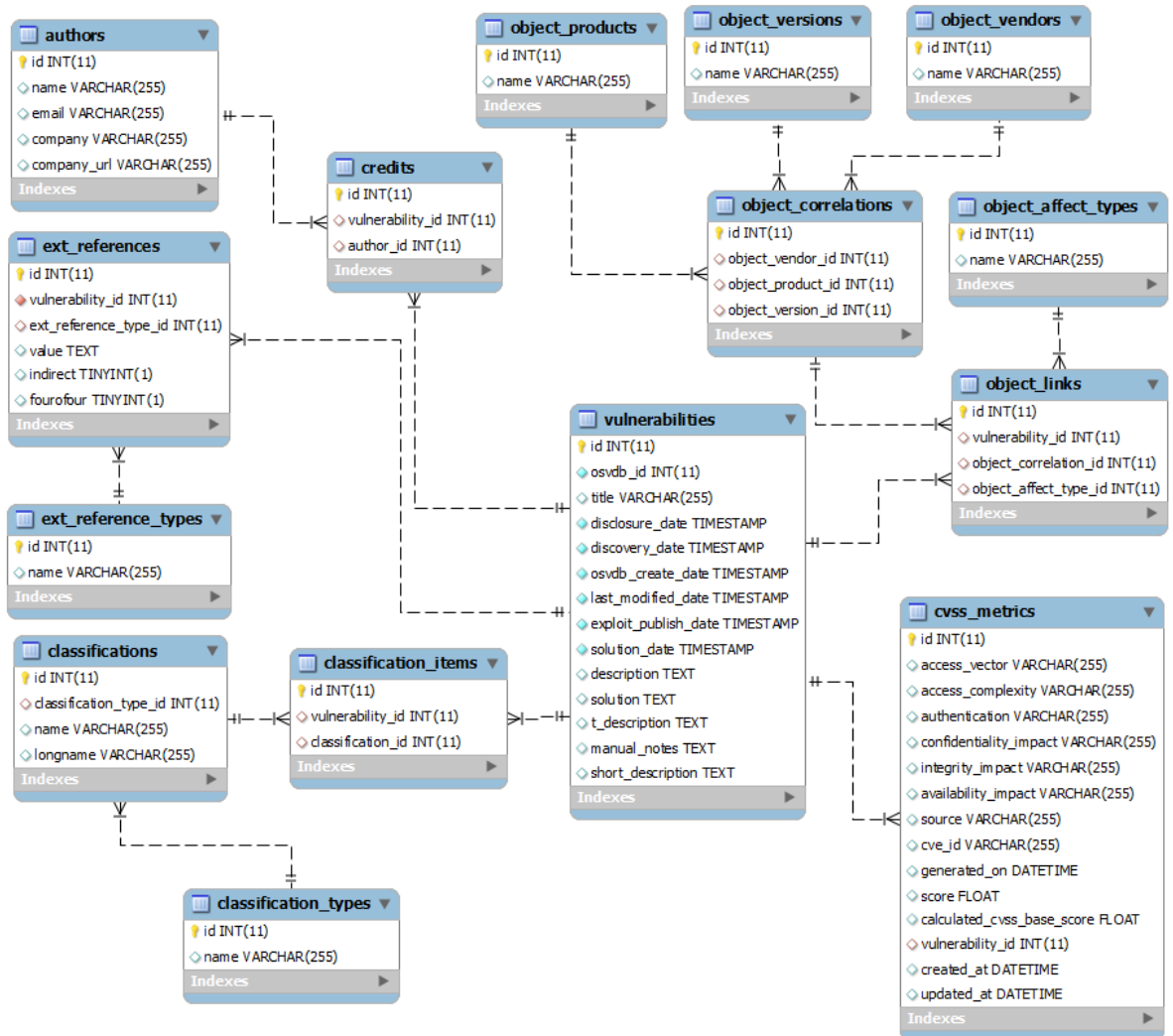
Notice: this example has been adapted based on the original representation from <http://cwe.mitre.org/data/definitions/476.html>.

Appendix 2: Definition of Types Weakness and Category in CWE 2.0





Appendix 3: OSVDB Data Model Overview



Bibliography

- [ALEXA11] *The top 500 sites on the web.* Alexa.
Retrieved from <http://www.alex.com/topsites>, November 2011.
- [ASLAM95] Aslam, T. *A Taxonomy of Security Faults in the UNIX Operating System.* MSc thesis. Purdue Univeristy, 1995.
- [ASSET96] Arthur, W.B. et al. *Asset Pricing Under Endogenous Expectations in an Artificial Stock Market.* (Working papers) Wisconsin Madison - Social Systems, 1996.
- [BPOINT10] Vincent, A. and Armstrong, M. *Predicting break-points in trading strategies with Twitter.* October 2010.
- [CABRERA2002] Cabrera, A. and Cabrera, E.F. *Knowledge Sharing Dilemmas.* Organization Studies, September 2002.
- [CCS11] *Attribution-ShareAlike 3.0 Unported.*
Retrieved from <http://creativecommons.org/licenses/by-sa/3.0/>, November 2011.
- [CHIRPS09] Krishnamurthy, Balachander et al. *A few chirps about Twitter.* Proceedings of the first workshop on online social networks. ACM New York, USA, 2008.
- [CMSEI05] Seacord, R.C. and Householder, A.D. *A Structured Approach to Classifying Security Vulnerabilities.* Software Engineering Institute, Carnegie Mellon University, 2005.
- [CULP00] Culp, S. 2000. *Definition of a security vulnerability.* MicrosoftTechNet, 2000.
- [CVE2011-2172] *Common Vulnerabilities and Exposures - CVE-2011-2172.* Retrieved from <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2172>, November 2011.
- [CVEED11] *CVE Editorial Board.*
Retrieved from <http://cve.mitre.org/community/board>, September 2011.

- [CVSS07] Mell, P., Scarfone, K., Romanosky, S. *A Complete Guide to the Common Vulnerability Scoring System Version 2.0*. National Institute of Standards and Technology and Carnegie Mellon University, 2007.
- [CWE07] *About CWE*. The Mitre Corporation.
Retrieved from <http://cwe.mitre.org/about>, September 2011.
- [CWEFAQ11] *CWE FAQ*. The Mitre Corporation.
Retrieved from <http://cwe.mitre.org/about/faq.html>, September 2011.
- [DBLCL11] *The 1000 most-visited web sites on the web*. DoubleClick. Retrieved from <http://www.google.com/adplanner/static/top1000>, November 2011.
- [DBP11] *The DBpedia Knowledge Base*. Retrieved from <http://dbpedia.org/About>, November 2011.
- [FBTW10] *Facebook, Twitter and The Two Branches of Social Media [OP-ED]*, retrieved from <http://mashable.com/2010/10/11/facebook-twitter-social>, Mai 2011.
- [GNUFD11] *GNU Free Documentation License*.
Retrieved from <http://www.gnu.org/copyleft/fdl.html>, November 2011.
- [HC11] *HighCharts product page*.
Retrieved from <http://www.highcharts.com/products/highcharts>, November 2011.
- [HEILPISK09] Heil, Bill and Piskorski, Mikolaj. *New Twitter Research: Men Follow Men and Nobody Tweets*. Harvard Business Blog, June 1, 2009.
- [HEYF99] Heylighen, Francis. *Collective Intelligence and its Implementation on the Web: algorithms to develop a collective mental map*. Computational & Mathematical Organization Theory, Springer, 1999.
- [HL98] Howard, J.D. and Longstaff, T.A. *A Common Language for Computer Security Incidents*. Sandia National Laboratories, 1998.
- [HOWISON08] Howison, J. *Cross-repository data linking with RDF and OWL*. 3rd Workshop on Public Data about Software Development (WoPDaSD 2008), p.15-22, 2009.

- [IATAC11] *Information Assurance Tools Report – Vulnerability Assessment*. Sixth Edition. IATAC, Herndon 2011.
- [IBMXSS11] *PM3664: Search Center - Cross Site Scripting Vulnerability*. Retrieved from <http://www-01.ibm.com/support/docview.wss?uid=swg1PM36644>, November 2011.
- [ISO27005] ISO/IEC 27005:2008: Information technology -- Security techniques -- Information security risk management. International Organization for Standardization, Geneva, Switzerland.
- [JANET08] *How “Janet” Fooled the Twittersphere (and me) She’s the Voice of Exxon Mobil*. Retrieved from <http://www.web-strategist.com/blog/2008/08/01/how-janet-fooled-the-twittersphere-shes-the-voice-of-exxon-mobil>, Mai 2011.
- [JAVA07] Java, Akshay et al. *Why We Twitter: Understanding Microblogging Usage and Communities*. In Proceedings of the Joint 9th WEBKDD and 1st SNA-KDD Workshop 2007. San Jose, USA, 2007.
- [JKALUC10] Kalucki, J. *Twitter Streaming API Architecture*, 2010. Retrieved from <http://www.slideshare.net/jkalucki/chirp-2010streamingapiarchpost>, September 2011.
- [JONES06] Jones, A. L. *Have internet message boards changed market behavior?*, info, Volume 8, Issue 5, 2006.
- [KOOBFACE10] Thomas, K. and Nicol, D.M. *The Koobface Botnet and the Rise of Social Malware*. 5th International Conference on Malicious and Unwanted Software (Malware), 2010.
- [KRSUL98] Krsul, Ivan V. *Software vulnerability analysis*. Purdue University, 1998.
- [LAMPSON04] Lampson, Butler W. *Computer Security in the Real World*. Computer, p. 37-46, June 2004.
- [LANDWEHR01] Landwehr, Carl E. *Computer security*. In International Journal of Information Security, Springer, 2001.

- [LP11] Carpenter, B and Baldwin, B. *Text Analysis with LingPipe 4*. LingPipe Publishing, New York, 2011.
- [METCAL05] Odlyzko, Andrei and Tilly, Benjamin. *A refutation of Metcalfe's Law and a better estimate for the value of networks and network interconnections*. Manuscript, March 2, 2005.
- [MSFT11] *Infographic: The Growth of Mobile Marketing and Tagging*. Microsoft, 03/2011. Retrieved from http://tag.microsoft.com/community/tag-blog-item/11-03-21/The_Growth_of_Mobile_Marketing_and_Tagging.aspx, May 2011.
- [MSGBRD04] Antweiler, W. and Frank, M. Z. *Is all that talk just noise? The information content of internet stock message boards*". Journal of Finance, volume 59, number 3, 2004.
- [NATHAZ10] Vieweg, S. et al. *Microblogging During Two Natural Hazards Events: What Twitter May Contribute to Situational Awareness*. Proceedings of the 28th international conference on Human factors in computing systems. ACM, 2010.
- [NETCRAFT11] *October 2011 Web Server Survey*. Netcraft. Retrieved from <http://news.netcraft.com/archives/2011/10/06/october-2011-web-server-survey.html>, November 2011.
- [NISTSP02] Stoneburner, G. et al. *Risk Management Guide for Information Technology Systems*. Special publication 800-30. National Institute of Standards and Technology, July 2002.
- [NRL93] Landwehr, C.E. et al. *A Taxonomy of Computer Program Security Flaws, with Examples*. Center for Computer High Assurance Systems, 1993.
- [OIS04] *Guidelines for Security Vulnerability Reporting and Response*. Organization for Internet Safety, 2004.
- [OLOUFA03] Oloufa, A. A. et al. *Situational awareness of construction equipment using GPS, wireless and web technologies*. Automation in Construction, Volume 12, Issue 6. November 2003.

- [ONT04] Garshol, L.M. *Metadata? Thesauri? Taxonomies?* Topic Maps! Ontopia, 2004.
- [OREILLY03] *The Architecture of Participation*, June 2004, retrieved from http://oreilly.com/pub/a/oreilly/tim/articles/architecture_of_participation.html, May 2011.
- [OREILLY07] O'Reilly, Tim. *What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*. O'Reilly Media, USA, 2007.
- [OZMENT07] Ozment, A. *Vulnerability Discovery & Software Security*. PhD Thesis. University of Cambridge & Magdalene College, 2007.
- [PARAMO96] Parasuraman, R. and Moloua, M. *Automation and human performance. Theory and applications*. Lawrence Erlbaum Associates Inc, 1996.
- [PLEVY98] Levy, Pierre. *Becoming Virtual: Reality in the Digital Age*. Plenum Trade, New York, 1998.
- [PRACUNIX96] Garfinkel, S. and Spafford, G. *Practical UNIX and Internet Security, Second Edition*. O'Reilly & Associates, Inc, 1996.
- [PRIVTW10] Humphreys, L. M., Krishnamurthy, B. and Gill, P. *How Much Is Too Much? Privacy Issues on Twitter*. Paper presented at the annual meeting of the International Communication Association, Singapore, 2010.
- [RISOS76] Abbott, R.P. et al. *Security Analysis and Enhancements of Computer Operating Systems*. Institute for Computer Sciences and Technology. National Bureau of Standards, 1976.
- [RTWFACT10] Suh, B. et al. *Want to be Retweeted? Large Scale Analytics on Factors Impacting Retweet in Twitter Network*. In Proceedings of the IEEE Second International Conference on Social Computing (SocialCom), p. 177-184, August 2010.
- [SARWOOD91] Sarter, N. B. and Woods, D. D. *Situation Awareness: A critical but ill-defined phenomenon*. International Journal of Aviation Psychology, p. 45-57, 1991.
- [SECTR10] Neuhaus, S. and Zimmermann, T. *Security trend analysis with CVE topic models*. Technical Report, DISI, University of Trento, May 2010.

- [SEMANTICOSS10] Berger, O. et al. *Weaving a Semantic Web across OSS repositories: a spotlight on bts-link, UDD, SWIM*. International Journal of Open Source Software and Processes, 32/2010, Volume 2, Issue 2, p. 29 - 40, 2010.
- [SFLU09] Morozov, E. *Swine flu: Twitter's power to misinform*. Retrieved from http://neteffect.foreignpolicy.com/posts/2009/04/25/swine_flu_tweeters_power_to_misinform, Mai 2011.
- [SITANAL05] Grégio, A.R.A. et al. *Taxonomias de Vulnerabilidades: Situação Atual. V Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*. Brasil, 2005.
- [SKING05] Tsipenyuk K. et al. *Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors*. Security & Privacy, IEEE, Vol. 3 Issue 6, 2005.
- [SOM10] Sakaki, T. et al. *Earthquake shakes Twitter users: real-time event detection by social sensors*. In Proceedings of the 19th international conference on World wide web. ACM, 2010.
- [SOOD11] Sood, A.K. and Enbody, R. *Chain Exploitation – Social Networks Malware*. ISACA Journal, Volume 1, 2011.
- [SPDGROUP00] Sonnenwald, D.H. and L.G. Pierce. *Information behavior in dynamic group work contexts: interwoven situational awareness, dense social networks and contested collaboration in command and control*. Information Processing and Management 36, p. 461-479, 2000.
- [SRFG11] *Download Timeline of phpMyAdmin*. Sourceforge. Retrieved from <http://sf.net/projects/phpmyadmin/files/stats/timeline>, November 2011.
- [SVCONS05] Polepeddi, S. *Software Vulnerability Taxonomy Consolidation*. Master's Thesis, Information Networking Institute, Carnegie Mellon University, 2005.
- [TADOPT09] Hughes, A. L. and Palen, L. *Twitter Adoption and Use in Mass Convergence and Emergency Events*. In Proceedings of the 6th International ISCRAM Conference – Gothenburg, Sweden, May 2009.

- [THONO10] Nowey, T. *Konzeption eines Systems zur überbetrieblichen Sammlung und Nutzung von quantitativen Daten über Informationssicherheitsvorfälle*. PhD Thesis. University of Regensburg, 2010.
- [TREE06] Engle, S. et al. *Tree Approach to Vulnerability Classification*. University of California at Davis, 2006.
- [TWEET09] *Twitter Blog*. Retrieved from <http://blog.twitter.com/2009/11/whats-happening.html>. May 2011.
- [TWITTER11] *#numbers*, retrieved from <http://blog.twitter.com/2011/03/numbers.html>, May 2011.
- [TWNEWS10] Haewoon, K. et al. *What is Twitter, a social network or a news media?*. In Proceedings of the 19th International Conference on World wide web, ACM, 2010.
- [TWRESP09] Mills, A. et al. *Web 2.0 Emergency Applications: How useful can Twitter be for emergency response?* In Journal of Information Privacy & Security, Volume 5, Issue 3, 2009.
- [TWRTW10] Boyd, D. et al. *Tweet, Tweet, Retweet: Conversational Aspects of Retweeting on Twitter*. In Proceedings of the 43rd Hawaii International Conference on System Sciences, 2010.
- [TWSTAT09] Lenhard, Amanda and Fox, Susannah. *Twitter and status updating*. Pew Internet & American Life Project, Washington D.C. 2009.
- [TWTRADE10] Sprenger, O. T. and Welpel, I. M. *Tweets and Trades: The Information Content of Stock Microblogs*. Working paper. Technische Universität München, 2010.
- [VCMOD08] Engle, S and Bishop, M. *A Model for Vulnerability Analysis and Classification*. Department of Computer Science. University of California, Davis, 2008.
- [WMOUTH09] Jansen, B. J. et al. *Twitter power: Tweets as electronic word of mouth*. In Journal of the American Society for Information Science and Technology, Volume 60, Issue 11, 2009.