Software System Selection based on First-Order Security Risk Assessment

By Ruma Das

B.E. in Computer Engineering, June 1995, University of Bombay, India
M.B.A, May 2004, University of Baltimore, USA
M.S.E in Computer Science, May 2006, Johns Hopkins University, USA

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial satisfaction of the requirements
for the degree of Doctor of Philosophy

May 19, 2013

Dissertation directed by

Thomas Mazzuchi
Professor of Operations Research and of Engineering Management

Shahram Sarkani
Professor of Operations Research and of Engineering Management

The School of Engineering and Applied Science of The George Washington University certifies that Ruma Das has passed the Final Examination for the degree of Doctor of Philosophy as of February 6th 2013. This is the final and approved form of the dissertation.

Software System Selection Based on First-Order Security Risk Assessment

Ruma Das

Dissertation Research Committee:

Shahram Sarkani, Professor of Engineering Management and Systems Engineering, Dissertation Co-Director

Thomas Andrew Mazzuchi, Professor of Operations Research and of Engineering Management, Dissertation Co-Director

James Wasek, Professorial Lecturer of Engineering Management and Systems Engineering, Committee Member

E. Lile Murphree, Jr., Professor of Engineering Management, Committee Member

© Copyright 2013 by Ruma Das
All Rights Reserved

Dedication

Dedicated to my parents, Bimal Kumar Mandal and Pranati Mandal.

Acknowledgment

First and foremost, I would thank God, for putting me on the path to achieve my dream of acquiring the doctoral degree.

My sincere gratitude goes to my advisors Dr. Sarkani and Dr. Mazzuchi for their insight and guidance. Thank you for keeping me focused and on the right path during this program.

Special thanks to Sarah and Michelle for keeping up with our administrative requests.

This degree would not have been possible without the sacrifices made by my husband and kids. I appreciate them for tolerating the lost vacation time, lonely weekends and missed family functions. I thank my kids for providing the welcome and appropriate diversion needed during this process. They are the lights of my life and they keep me grounded. I am grateful for my husband's understanding and thank him for giving me the strength to go on. I would be lost without him.

Finally, I am thankful for having wonderful parents and siblings who always believed that I could do it and encouraged me to take up this program. It is the values and teachings of my parents that drove me toward higher education. I am grateful to my parents for their sacrifices and for providing me with the foundation to grow intellectually and as a human being.

Abstract

Software System Selection Based on First-Order Security Risk Assessment

"Often, multiple software products exist on the same server and vulnerability in one product compromises the entire system. It is imperative then, to perform a security risk assessment when selecting the software products that will become part of a larger system. Having a quantitative security risk assessment model provides an objective criterion for such assessment and comparison between candidate software systems. This research presents a software product evaluation method using such a quantitative security risk assessment model. This method utilizes prior research in quantitative security risk assessment based on empirical data from the National Vulnerability Database (NVD), and compares the security risk levels of the products evaluated. We introduced topic modeling to build a security risk assessment model. The risk model has been created using the Latent Dirichlet Allocation (LDA) algorithm to classify the vulnerabilities into topics, which are then used as the measurement instruments to evaluate the candidate software product. Such a procedure could supplement the existing selection process and assist decision-makers' evaluation of Open Source Software (OSS) systems, to ensure that the software product is safe and secure enough to be put into organization's server environment. Thus, using a design science research methodology, we propose that historic vulnerability data be utilized to develop a quantitative security risk assessment model, to provide a tool to system engineers for comparison and evaluation of OSS systems before their adoption. Finally, the procedure is demonstrated using an experimental case study."(Ruma Das, 2012)

Table of Contents

Dedication iv

Acknowledgment v

Abstract vi

List of Figures xii

List of Tables xiv

1 Introduction 1

1.1 Prior Studies and Deficiencies 2

1.2 Motivation 5

1.3 Relevance and Importance 6

1.4 Purpose of the Study 7

1.5 Statement of the Problem 8

1.6 Research Questions 9

1.7 Goals and Objectives 9

1.8 Contributions 10

1.9 Dissertation Organization 10

1.10 Summary 11

2 Background and Definition 12

2.1 Background 12

2.1.1 National Vulnerability Database (NVD) 12

2.1.2 Topic Modeling 13

2.1.3 Design Science Research Methodology 17

2.2 Definitions 19
2.2.1 Vulnerability 19
2.2.2 Static and Dynamic Analysis 20
2.2.3 CVE 21
2.2.4 CWE 22
2.2.5 CVSS 22
3 Literature Review 25
3.1 System Engineering and Risk Assessment 25
3.1.1 Risk Identification 27
3.1.2 Risk Assessment 28
3.1.3 Risk Prioritization and Handling 28
3.1.4 Risk Tracking 28
3.1.5 Risk Assessment Challenges 29
3.2 Quantitative Security Risk 29
3.2.1 Research using Public Data 32
3.2.2 Models using Custom Data 39
3.2.3 Framework, Metrics and Measurement Models 41
3.2.4 Identifying Security Risk Groups 43
3.3 Software Selection 47
3.4 Gaps in Research 52
3.5 Needs for a New Study 53

4 Research Framework 57
4.1 Conceptual Framework 57
4.1.1 Hypothesis 57
4.2 Comparative Analysis of Related Research 58
4.2.1 Text Mining 58
4.2.2 Security Metrics and Measurements 59
4.3 Research Framework Summary 65
5 Research Methodology 67
5.1 Design Science Research Methodology (DSRM) 67
5.2 Rationale for DSRM 68
5.3 Problem Identification and Rationale 69
5.4 Objective of the Solution 71
5.5 Model Design and Development 71
5.5.1 Proposed Model 71
5.5.2 Dimensions of Risk 73
5.5.3 Quantitative Security Risk Model Development 74
5.5.4 Verification and Validation 84
5.6 Demonstration 104
5.6.1 System Requirements Risk Analysis 105
5.6.2 Product Design Risk Analysis 107
5.7 The Prototype 112

5.7.1 Topic Cluster Analysis 113
5.7.2 Term Analysis 114
5.7.3 Comparative Analysis of CIA metrics 115
5.7.4 System Requirements Risk 116
5.7.5 Product Design Risk 118
5.7.6 Product Risk Score Calculation 118
5.7.7 Final Risk Score Display 119
6 Evaluation and Results 121
6.1 Quantitative Security Risk Assessment Model Evaluation 121
6.2 Procedure Evaluation 124
6.3 Hypothesis Reference 129
6.4 Limitations 131
6.5 Conclusion 132
7 Future work 134
8 Publication history 137
8.1 Communication 137
9 References 138
11 Appendix 157
11.1 Appendix A: Scanner Categories 157
11.2 Appendix B: CWE used in NVD 158
11.3 Appendix C: String Match using Q-Gram 160

11.4 Appendix D: Terms in each topic 167

List of Figures

Figure 1-1. IT Risk Management Process(Yue et al., 2007) 5

Figure 1-2. Study Focus 7

Figure 2-1. LDA Model (Blei, 2012) 14

Figure 2-2. Design Science Research Methodology Steps (Hevner et al., 2004) 18

Figure 2-3. DSRM Process Model(K. E. N. Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007) 19

Figure 2-4. CVSS Metric Groups (Peter Mell, 2012) 24

Figure 3-1.Total Risk Management (Haimes)(Sage & Rouse, 2009) 26

Figure 3-2. Steps in Risk Management (MITRE, 2012) 27

Figure 4-1. Conceptual Model 57

Figure 4-2. Quantitative Security Risk Level Estimation(Houmb et al., 2010) 62

Figure 4-3. Research Framework 66

Figure 5-1. DSRM adapted from (Peffers et al., 2008) 68

Figure 5-2. Model for OSS selection using Quantitative Security Risk Analysis(Ruma Das, 2012) 71

Figure 5-3. Risk assessment dimensions for the candidate product(Ruma Das, 2012) 73

Figure 5-4. Quantitative Model Components 77

Figure 5-5. Quantitative Security Risk Assessment Model Creation Process 78

Figure 5-6. Topic Selection Evaluation(Ruma Das, 2012) 79

Figure 5-7. nvddata-document-topic-distributions output 80

Figure 5-8. Crosstab visualization verifying topic terms overlap 91

Figure 5-9. Sample Tag cloud of Topic Terms 93

Figure 5-10. Prototype Design 112
Figure 5-11. Topic Cluster Analysis Screen 114
Figure 5-12. Term Analysis Screen 115
Figure 5-13. Comparative Analysis of CIA metrics Screen 116
Figure 5-14. System Requirements Risk 117
Figure 5-15. Product Design Risk Screen 118
Figure 5-16. Product Risk Score Calculation Screen 119
Figure 5-17. Final Risk Score Display Screen 120
Figure 6-1. RQ1- Dimensions of Risk(Ruma Das 2012) 128

List of Tables

Table 2-1. Topic Modeling Artifact mapped to the Study 16

Table 3-1. Impact categories - Secunia & X-force 45

Table 3-2. Vulnerability Cause categories - BID 45

Table 4-1. Prior work in Vulnerability Database Classification 58

Table 4-2. Comparative Analysis of Related Models 59

Table 5-1. Artifacts for Design Science Research (Hevner et al., 2004) 67

Table 5-2. Topics and Top 10 Related Terms Output From LDA toolbox 80

Table 5-3. Vulnerability and Topic with maximum probability distribution 81

Table 5-4. Scanner Category Topic Mapping using AppScan categories 82

Table 5-5. Topic CWE verification 85

Table 5-6. Precision-Recall between Topic Groups and CWE Types 87

Table 5-7. Multi-label check 89

Table 5-8. Final Topic Labels 89

Table 5-9. Model Score Standard Deviation Verification 93

Table 5-10. t-Test: Paired Two Sample for Means 95

Table 5-11. Topic CVE verification 97

Table 5-12. Topic Top Terms & Scanner Categories - String Match Q-Gram 98

Table 5-13. Topic Probabilities 100

Table 5-14. Top 10 Risks by Year 102

Table 5-15. Topic Vulnerability change from year 2007-2012 103

Table 5-16. Sample CVE base score, exploit score(Ruma Das, 2012) 105

Table 5-17. openMRS System Requirements Risk Score(Ruma Das, 2012) 106

Table 5-18. openEMR System Requirements Risk Score(Ruma Das, 2012) 107

Table 5-19. openMRS Dynamic Analysis(Ruma Das, 2012) 108

Table 5-20 Skipfish Vulnerability Types-Topic Term Assignment (Ruma Das, 2012). 108

Table 5-21. Sample Topic vulnerability count and date range(Ruma Das, 2012) 109

Table 5-22. Sample Topic vulnerability count and Frequency(Ruma Das, 2012) 109

Table 5-23. openMRS Topic Average score and Frequency (Ruma Das, 2012) 109

Table 5-24. openMRS vulnerability weight analysis(Ruma Das, 2012) 110

Table 5-25. Sample openEMR Dynamic Analysis(Ruma Das, 2012)..... 110

Table 5-26. openEMR vulnerability weight analysis(Ruma Das, 2012) 111

Table 5-27. Quantitative Security Risk evaluation of openMRS(Ruma Das, 2012) 111

Table 5-28. Quantitative Security Risk evaluation openEMR(Ruma Das, 2012) 111

Table 6-1. 'Android' term dataset topic analysis 123

Table 6-2. Analysis of newer technology related topics 124

Table 11-1. Skipfish Scanner Categories Used..... 157

Table 11-2. AppScan Scanner Categories used 158

Table 11-3. CWE used in NVD 158

Table 11-4. Topic Top Terms & Scanner category String Match using Q-Gram 160

Table 11-5. Top Ten Terms in Each Topic 167

1 Introduction

Adopting an open source product or an off-the shelf product offers multitude of benefits and brings along multitude of challenges. These challenges need system engineering approach, so that the system selected not only meets "the mission of the organization, but also satisfies the security risk level needed by it" (J. L. Bayuk, 2011). For example, a web-based open source system selected for electronic health records management must satisfy not only functional requirements, but also the audit needs, the authorization needs, and the security control requirements. Finally, it must also meet the confidentiality levels required by the organization. The definition of system engineering as provided by International Council on Systems Engineering (INCOSE) version, says that system engineering "an interdisciplinary approach and means to enable the realization of successful systems"(INCOSE, 2003) version 3.0. We apply this for an OSS or a COTS selection and propose that "realization" in this case means "system implementation". For successful "system implementation" "it is imperative to conduct the security risk assessment of the software product which will eventually become part of a larger system"(Ruma Das, 2012). For a software system developed in-house, making it secure entails undertaking proactive measures during the different phases of the software development lifecycle (SDLC). In such cases, the security assessment is conducted in the development, design, verification, and validation (V&V) phases through activities such as vulnerability scanning (Childs, 2003), secure code review, threat modeling, etc. However, for an off-the-shelf or OSS product, the party selecting the software product is not likely to be involved in the SDLC, and thus has no control over ensuring that security

aspects have been followed during the product design . In this case, “a security risk assessment becomes even more essential and must be conducted upfront during the initial evaluation”(Ruma Das, 2012). However, this is not done during the initial evaluation of a software product as the security risk assessment is often seen as a non-critical task, requiring immense time and expertise. As pointed out by Schneidewind, an unsecure COTS or OSS product may be a weak link in the system environment (Schneidewind, 1998) impacting the reliability of the entire system. Therefore, it is more critical that security risk be evaluated before implementing a software product not developed in-house. Having a quantitative model helps provide an objective view to security risk assessment during product selection. However, providing a rationale for risk assessment rather than objectivity remains the goal (Hall, 2011). Risk identification is often a manual process that depends on the experience levels of the practitioners. Identifying risk by mining historic data can help support this manual process(Leidner & Schilder, 2010). Determining the numerical impact value and the probability of the occurrence of risk is often affected by a subjective bias in the manual process. In terms of software vulnerability, a wealth of historic data that provides information about the discovered vulnerabilities in different software is available. Mining this information offers a valuable resource for risk identification and assessment, thereby, supporting the manual process. However, this vulnerability information is mostly in text format. So exploring text mining algorithms for quantitative security risk assessment modeling is expected to lead to a potential, useful solution.

1.1 Prior Studies and Deficiencies

“Security flaws in a software product increase the security risk of the whole

environment in which it operates. When evaluating an open or closed source off-the-shelf software system, one must conduct the software security assessment to ensure that the product has a relatively low security risk. The term ‘software product’ in this research refers to a web-based database-driven application. When comparing multiple candidate software products, there is a need for an objective measure to evaluate and support decision-making”(Ruma Das, 2012). The quantitative security risk assessment in prior research is prominent in areas such as “ decision-making, risk prediction, and resource allocation”(Chenmeng Sui, 2012; Houmb, Franqueira, & Engum, 2010; Joh, 2011; Mkpong-Ruffin, Umphress, Hamilton, & Gilbert, 2007; Wang, Guo, Wang, Xia, & Zhou, 2009; Woo, Joh, Alhazmi, & Malaiya, 2011; Zhihu & Xin, 2010) . “Prior studies in quantitative software security risk assessment have mainly focused on: browsers, application servers, database servers, individually or on the entire network topology. A new study is needed to identify how we can apply or extend such a model and metrics to compare and select software products, such as web-based applications”(Ruma Das, 2012). In the area of mining historic data and developing security risk groups, Self Organizing Map(SOM) (Venter, Eloff, & Li, 2008) and classifiers(Zhongqiang Chen, Zhang, & Chen, 2010) have been proposed, both, for categorization and standardization of security risk groups. Topic Modeling algorithms used for security trend analysis(Neuhaus & Zimmermann, 2010) have demonstrated their potential for security risk group identification along with their ability to generate groups identified by ‘topic terms’ which make topic modeling attractive options for standardizing vulnerability and assigning user-friendly labels, so as to create models for decision support.

According to Bayuk et al. systems security engineering is a branch of system

engineering that implements “scientific and engineering principles to identify security vulnerabilities and minimize or contain risks associated with these vulnerabilities”(Jennifer L. Bayuk & Horowitz, 2011). Our research adds to this field by investigating security attributes and security risk measurements derived from empirical data, which can help evaluate the risk when selecting a COTS or OSS software package. Bayuk has noted that in systems engineering, the focus has been on implementing security standards rather than the measurement of systems security (J. L. Bayuk, 2011). The author emphasizes the need to adopt “empirical methods and leverage known security measurement models, tools and processes into system engineering” (J. L. Bayuk, 2011). Thus, we investigate empirical methods to evaluate a software product with the aim of assessing its security risk. Black-box testing and white-box testing are the traditional methods used for this purpose but the challenge lies in converting them into a quantitative model for decision-making. One of the goals of our research is to fill this gap and help integrate the output of the vulnerability scanner with a risk assessment model.

IT Risk management process for security, as defined by Yue et al. consists of 3 main stages: Risk Identification, Analysis, and Control (Yue, Çakanyıldırım, Ryu, & Liu, 2007) as shown in Figure 1-1. Software Risk Management or assessment is a subset of IT System Risk Management since the software is a part of the entire system. Our research in software risk assessment aims at supporting the activities in the first two steps of the process given below for the given software.

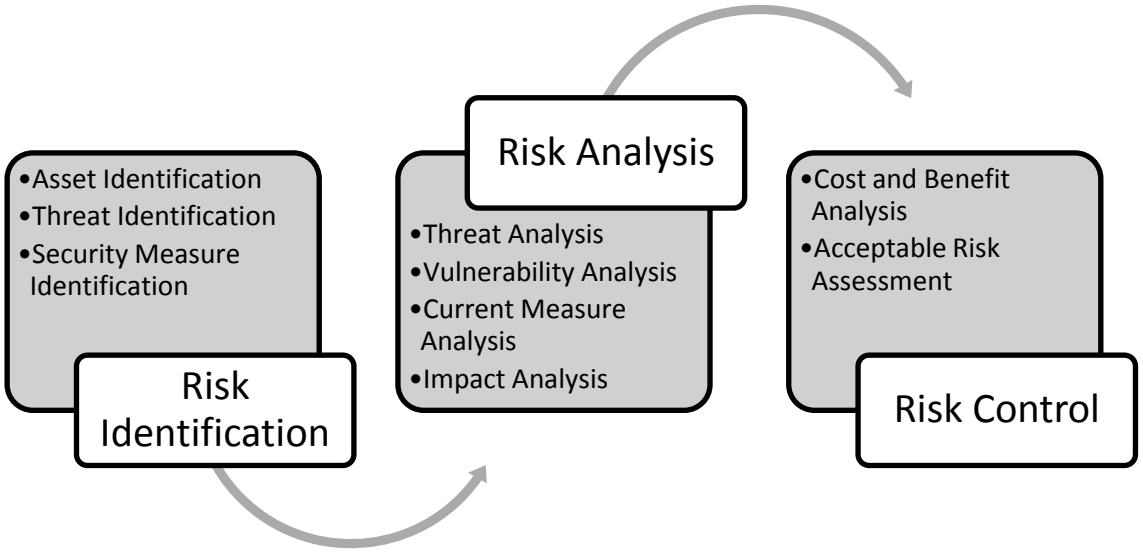


Figure 1-1. IT Risk Management Process(Yue et al., 2007)

1.2 Motivation

An insecure software product can increase the security risk of the entire environment in which it is installed. Potential users or system engineers selecting it have no control over security features architected into the OSS- or COTS-based product. In this situation, it is critical to have a procedure to perform security risk assessment during COTS and OSS selection. A survey of OSS selection processes has shown that the method of OSS selection is the 'first-fit' process. A security expert is most often not used during the initial COTS or OSS selection process because security assessment is time consuming, resource intensive, and needs specialized expertise. Therefore, this research is aimed at designing a quantitative security risk assessment procedure, based on empirical data, which is simple yet effective, without heavy burden on the resources and can be used during the early stages of software selection.

1.3 Relevance and Importance

This research covers two areas: the development of a quantitative security risk assessment model using historic data and the usage of that model to evaluate the security risk of an OSS product or other off-the-shelf products. “The relevance of the research is observed in a case study undertaken by Austin et al. that evaluated multiple open source electronic health record systems. The researchers detected approximately 1321 issues through static analysis and 710 issues through dynamic analysis (Austin & Williams, 2011). Security assessment during OSS selection has also been proposed in prior works(Sung, Kim, & Rhew, 2007). However, skilled security professionals were rarely used during the initial selection process of a software system”(Ruma Das, 2012). The framework discussed in this research may be utilized by practitioners to augment the existing security assessment processes during the early stages of software product evaluation. Another benefit exuded by this framework is that having such a model to standardize software risk groups and map it to the vulnerability scanner may help develop security control strategies for risk mitigation. Finally, having a model that can be automatically fed with vulnerability reports, which in turn are automatically categorized into predefined risk groups, may help create a security risk monitoring dashboard or score card. Figure 1-2 shows the focus of this study.

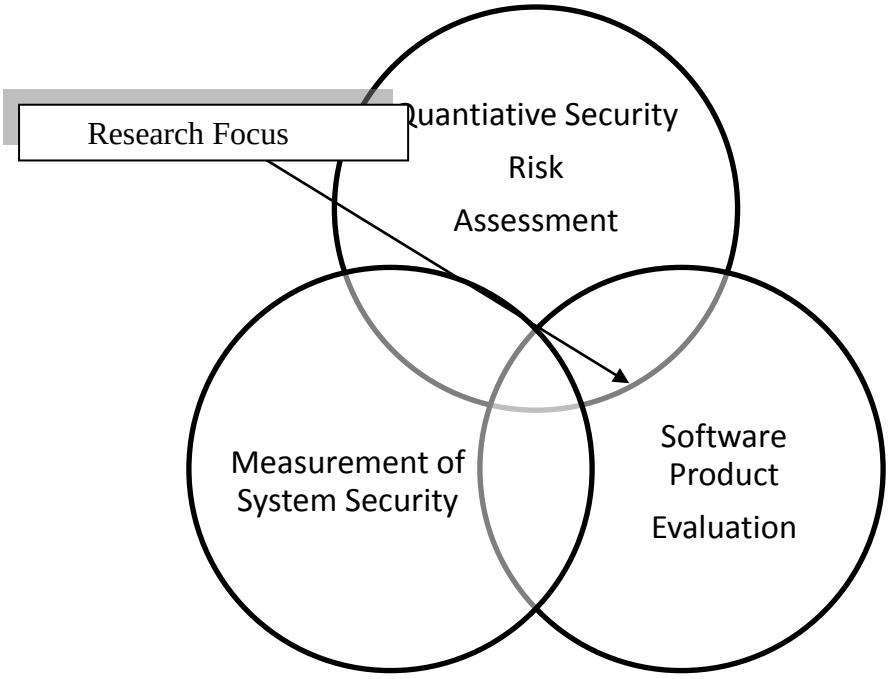


Figure 1-2. Study Focus

1.4 Purpose of the Study

This study aims at developing a quantitative security risk assessment model that can be used as a measurement instrument, to quantitatively evaluate the security risks of a software product. “This study introduces a procedure based on a quantitative security risk assessment to enable the selection of a product from a multitude of OSS products. Our approach investigates the application of topic models to create the quantitative security risk assessment model. This approach of having the machine generate a list of topics has been chosen since not all vulnerabilities can be categorized in the NVD using the CWE (Common Weakness Enumeration) groups(MITRE, 2011b). Consequently, it is difficult to determine the actual likelihood or frequency, and thereby, severity of the security risk group”(Ruma Das, 2012). In addition, security information is often in the summary text field of the reported vulnerability record, in the vulnerability database. Thereby, mining

the textual data to develop security risk models helps extract the latter based on the natural semantic nature of the data. In addition, it may be integrated with other sources, where such descriptions are available in text format, such as the categories in the vulnerability scanner (Venter et al., 2008). Finally, the research was conducted using the Design Science Research Methodology (DSRM). The primary reason for adopting this research methodology is its support for 'artifact" creation for "problem-solving" (K. Peffers, Tuunanen, Rothenberger, & Chatterjee, 2008) which meets our goal, namely "to propose an artifact to support the evaluation of off-the-shelf or OSS software products before adoption using the quantitative security risk assessment model"(Ruma Das, 2012). We have described our research activities by adopting the steps mentioned by Peffers et al., which helped structure the information generated in this study, performed using Design Science Research Methodology (DSRM) (K. Peffers et al., 2008).

1.5 Statement of the Problem

We use multiple criteria for assessment when selecting an off-the-shelf or OSS framework. There are OSS selection frameworks, such as openBRR, QualOSS, openCMM etc. that are used for conducting such evaluations. However, focus on quantitative security for software selection is lacking. Security must be one such important criterion since an unsecure system or component can reduce the security level of the entire environment in which it is installed. With an OSS, 'security-by-design' practices cannot be implemented since the system engineers have no control over the design and build phases. "The problem that system engineers need to therefore address is how to evaluate and compare the potential security risks of the different open source systems before adoption. Having a quantitative security risk assessment model provides

objective criteria for comparison between different systems. This study proposes a method for software system comparison and selection using such a quantitative security risk assessment model. Our goal is to utilize prior research in quantitative security risk assessment based on empirical data from the NVD and estimate the security risk of the system evaluated. Such a method could help practitioners evaluate and compare off-the-shelf or OSS software products to ensure that they are safe and secure enough to be installed in their environment"(Ruma Das, 2012).

1.6 Research Questions

"The primary research questions investigated here are:

RQ1: What are the essential dimensions for evaluating the security of a candidate software product before selection?

RQ2: How do we analyze the quantitative security risk level of a potential software product?

RQ3: Can we use a quantitative security risk assessment model to compare and select between custom software products? "(Ruma Das, 2012)

1.7 Goals and Objectives

The primary goal of the study is to develop a procedure, which supplements the existing OSS selection framework and assists practitioners during the early stages of software product evaluation, on the basis of a quantitative security risk assessment. The main objective research is to develop this procedure so that it is:

Effective during preliminary assessment

Helpful to an analyst not having extensive security expertise

Able to provide numeric values for objective analysis

In order to meet this goal and objective, we explore and validate the use of topic modeling for vulnerability categorization using historic data. This research intends to examine quantitative risk assessment methods with focus on security risk, build procedure based on empirical data, conduct an experiment to evaluate the process, and determine the answers to the research questions.

1.8 Contributions

This research lays the ground work for the usage of topic modeling algorithms for security risk mining, risk identification, and risk impact analysis from historic vulnerability data. The research identifies dimensions and provides procedures for security assessment of candidate software products during their initial evaluation and selection phases. It adds to the body of knowledge by integrating concepts from quantitative security risk assessment modeling, text mining, machine learning, and software selection frameworks.

1.9 Dissertation Organization

The remainder of this dissertation focuses on developing a procedure for software selection based on quantitative security risk assessment. Chapter 2 provides definitions of terms and the background necessary to lead to the next chapters. Chapter 3 reviews prior research both, in areas of quantitative security risk assessment as well as software selection criteria. Chapter 4 presents the framework on which the research is based. Chapter 5 describes the methodology used in the research and gives an overview of the

data collection approach. Chapter 6 discusses the evaluation and results. Chapter 7 provides the conclusion and explores work that can be undertaken in future. The final chapter outlines the conferences where the work was presented and the publication that made it known.

1.10 Summary

The anticipated contribution of this research is to develop a quantitative risk assessment model and procedures that will help select software products without knowing software product's internal structure. There are many qualitative as well as quantitative criteria for software product selection. Our goal is to provide a supporting tool to evaluate a product, based on the quantitative security risk score armed with the goal of having a secure system.

2 Background and Definition

2.1 Background

2.1.1 National Vulnerability Database (NVD)

"The National Vulnerability Database (NVD, 2012), is a US federal government repository, produced by the National Institute of Standards and Technology (NIST) Computer Security Division. This repository contains publicly available vulnerability data. Currently there are over 44,000 Common Vulnerabilities and Exposures (CVE) entries in the NVD"(Ruma Das, 2012). Being a public data source, it is popular in many research areas related to security such as, vulnerability discovery models (VDM) (Ozment, 2007), and security metrics creation(Schryen & Kadura, 2009). Schryen et al. developed a software metrics named 'Patch index'(Schryen & Kadura, 2009) and used the NVD data to compare the security of the open source and the closed source software. They categorized vulnerability into three classes (high, low, and medium) and calculated the patch index based on the number of patched versus un-patched vulnerabilities(Schryen & Kadura, 2009). Another area of research that utilized NVD is 'decision support' to "allow for resource and effort management, through prioritization"(Fruhwirth & Mannisto, 2009). In this study, Fruhwirth et al. recalculated the security risk score such that the temporal and environmental context information was provided by the practitioners. This study demonstrated that the context information can result in a re-prioritization of the vulnerabilities and thus support efficient allocation of resources(Fruhwirth & Mannisto, 2009). The authors' rationale for developing this model lay in that not all organizations exercised the same amount of importance for the same security metrics, and thus allowed context information to be supplied, leading to change

in the risk impact score and therefore, the priority(Fruhwirth & Mannisto, 2009). In another study, Chung et al. created a procedure for security risk assessment in which the vector composed of attributes “Data , Server and Application” was cross-referenced with the vector “confidentiality, integrity, and availability”(Chung, Kim, Lee, Lee, & In, 2005). An et al used NVD data to propose a similarity-based measurement and demonstrated its use for patching ,risk mitigation, and performing other functions in the security management areas (Ju An, Linfeng, Minzhe, Hao, & Camargo, 2010). In this research, An et al. utilized Ontology for Vulnerability Management (OVM) and data from NVD to determine the similarities between vulnerabilities. Using a Hierarchical Similarity Algorithm, the vulnerabilities were compared using the attributes “Type, Product, Access Vector, Access Complexity, Authentication, Confidentiality Impact, Integrate Impact and Availability Impact” (Ju An, Linfeng, et al., 2010). NVD data contains information about different vulnerabilities found in the COTS, such as database servers, application servers, browsers etc. We integrate this information in our research to determine the overall security composure of a custom software product and its potential implementation environment.

2.1.2 Topic Modeling

Topic Modeling algorithms are popular in areas requiring clustering and analysis of large amount of unstructured text. The term topic modeling refers to the technique of identifying themes in large collection of documents, as indicated by sets of terms or tags ("Probabilistic Topic Models," 2012). It is categorized as unsupervised machine learning and probabilistic modeling(Anthes, 2010). “The Latent Dirichlet Allocation (LDA) algorithm is a type of topic modeling, which considers the input text as bag of words and

uses Bayesian Inference to understand the distribution for topic grouping"(Ruma Das, 2012). A text document is often composed of a number of topics. Thus, the algorithm identifies a set of topics and provides the proportion of that topic in that document after considering the whole corpus. Also, it provides the proportion of the terms in a topic, thus indicating the most relevant words based on the whole corpus.

The order of the words and the document does not affect the algorithm. However, the number of topics must be manually assigned and this affects the simple implementation of the LDA algorithm. There are topic modeling algorithms, such as hLDA (Hongyan & Lei, 2011), which do not required a number of topics to be set in advance, since it creates a tree structure with general terms at the head of the tree and more specialized terms at the bottom nodes. However, hLDA takes much longer to run than the LDA algorithm. Figure 2-1 below, shows the basic structure and explains the LDA algorithm as provided by Blei (Blei, 2012) (Lafferty, 2009).

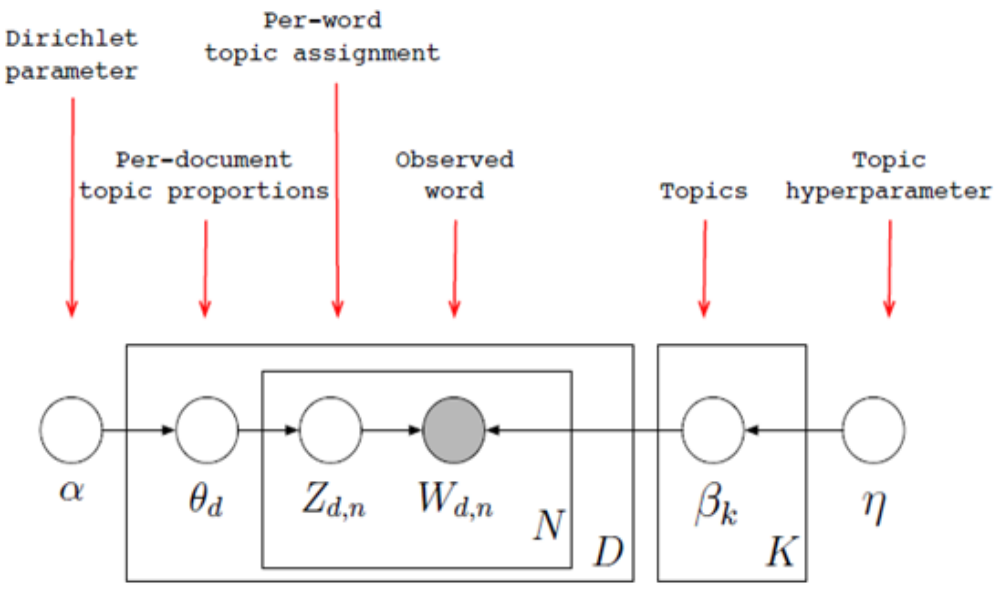


Figure 2-1. LDA Model (Blei, 2012)

Where

D= Document‘d’ indicates d^{th} document

N= Words, n indicates n^{th} word

θ: the posterior distribution of dirichlet, for a multinomial distribution. Posterior distribution is the conditional distribution of the hidden variables given the observed variable.

α: Affects the mean shape and sparisity of θ

$Z_{d,n}$: Per-word Topic Assignment

$W_{d,n}$: Per-document Topic Proportion

β_k :Per-corpus topic Distribution

k : indicates k^{th} topic

Due to its ease of use, the topic modeling algorithm finds application in a number of areas, not only for text mining but also analyzing voice data, scanned Optical Character Recognition (OCR) text data and software code evolution. In area of software evolution, Asuncion et al. investigated software artifact traceability (Asuncion, Asuncion, & Taylor, 2010), where as Linstead et al. used topic modeling algorithm to “extract concepts or functional topics from the software code”(Linstead, Rigor, Bajracharya, Lopes, & Baldi, 2007). In addition, a few of the other varied applications making use of this modeling were shortlisted as those with a potential for speech recognition(Kuan-Yu, Hsuan-Sheng, & Chen, 2010), understanding social network ecosystems (Negoescu & Gatica-Perez, 2010), investigating software evolution (S. W. Thomas, Adams, Hassan, & Blostein,

2010), analyzing genomic data (Xin, Xiaohua, Xiajiong, & Rosen, 2010) etc. "The essence of the algorithm is its power to extract the semantic nature of the text. However, the challenge is then to analyze the meaning of the latent topics that emerge"(Ruma Das, 2012). Understanding the meaning of the topics requires analysis of the terms and labeling of some sort. Automatic labeling algorithms are another active area of research investigating solution to this issue (Hindle, Ernst, Godfrey, & Mylopoulos, 2011; Magatti, Calegari, Ciucci, & Stella, 2009). An example of 'domain-specific labeling' is proposed by Hindle et al, using word-lists derived from ontology for non-functional requirements designed by prior research, and the ISO 9126 documentation. This word-list is then used to automatically assign labels to the topic groups generated by the LDA algorithm, by matching the terms in the topic group with terms in the domain-specific word-list.

The primary limitation of LDA is the need to predefine the number of topics before execution of the algorithm. Incorrect estimate of number of topics can result in the topic clusters to be too generic in case the number of topic is small or having too much overlap if the number of topic is large. Finally the interpretation of the topic cluster semantic requires manual or automatic effort.

The mapping between, the topic modeling components and its relation to the artifacts of this research is shown below in Table 2-1:

Table 2-1. Topic Modeling Artifact mapped to the Study

Topic Modeling Artifact	Artifacts of this research
Document	Summary Text from each CVE record in the NVD
Topic	Latent Security Risk Groups Identified
Terms	Words representing each security group

2.1.3 Design Science Research Methodology

Design Science Research Methodology (DSRM), which is popular in the field of information systems, is rarely found in system engineering. Hevner et al. proposed the design-science paradigm for information systems (IS) since it is a problem-solving research paradigm and results in the creation of an artifact to solve the problem(Hevner, March, Park, & Ram, 2004). As per Hevner, IS being an applied science field, research here, often contributes to the body of knowledge by the creation of "new and innovative artifacts" (Hevner et al., 2004). Hevner structured DSRM by defining 2 processes and 4 artifacts. The processes were, "design and evaluate." The artifacts were "constructs, models, methods, and instantiations" (Hevner et al., 2004). As per Hevner et al., the artifacts created using DSRM help to "address unsolved problems" and the value relates to "utility provided in solving those problems"(Hevner et al., 2004). The high-level steps of DSRM suggested by Hevner et al. have been shown in Figure 2-2 below.

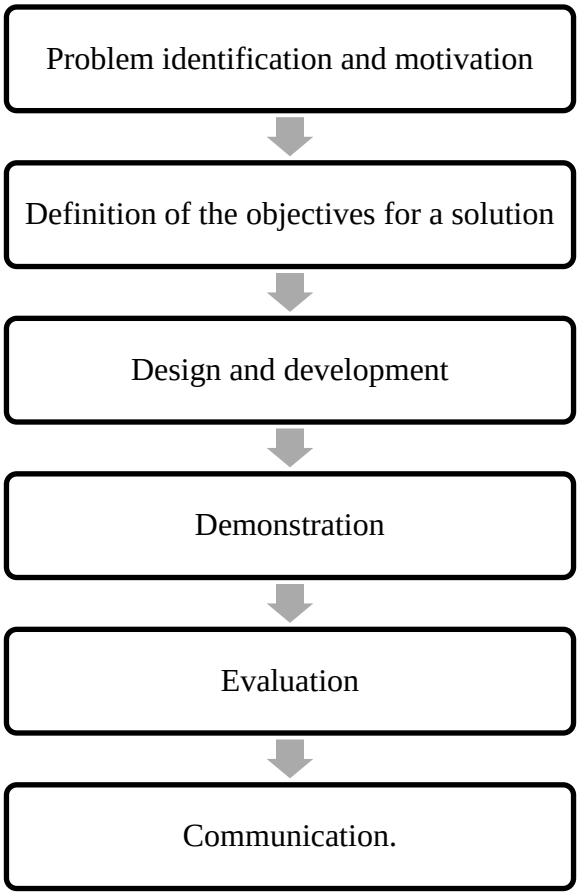


Figure 2-2. Design Science Research Methodology Steps (Hevner et al., 2004)

Peffers et al. synthesized prior research that used DRSM and provided a structured framework for DSRM research presentation as shown in Figure 2-3. Even though the model was created for developing artifacts in information system research, its structured methodology was well-suited for any research field that created artifacts to solve a problem.

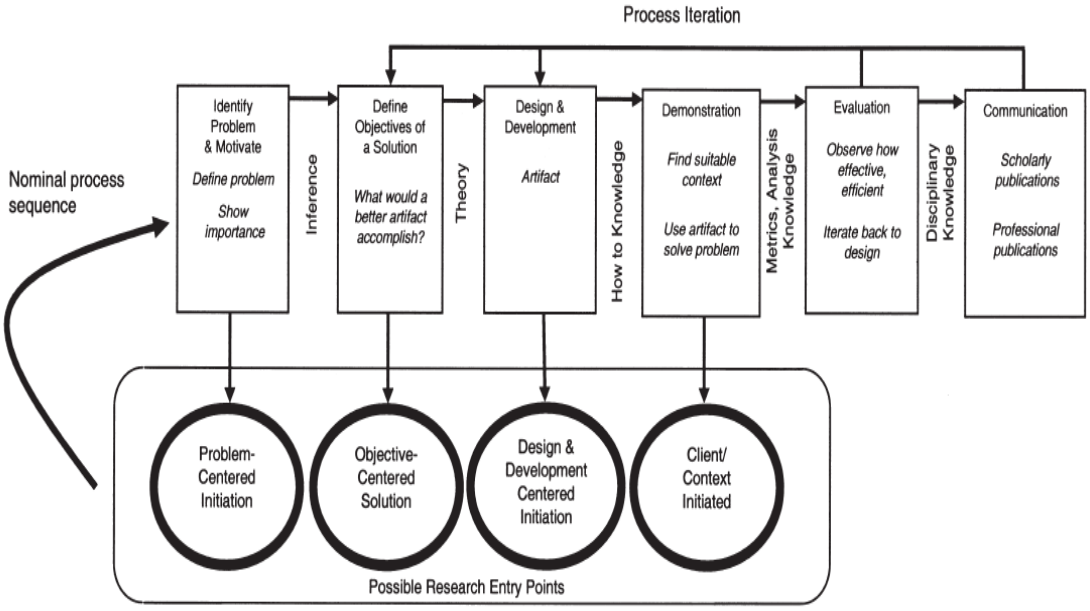


Figure 2-3. DSRM Process Model(K. E. N. Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007)

The design science approach is being increasingly adopted in fields other than IS, such as software engineering, computer engineering etc. For example, it has been used to present framework development for security profiles of web service (Alrouh, Al-Debei, & Ghinea, 2010), tool development for the application of maturity models (Krivograd & Fettke, 2012), prototype development for portfolio performance measurement (Weber, Beck, Wolf, & Vykoukal, 2010) etc. thus demonstrating its application in presenting research-based artifact creation for problem solving.

2.2 Definitions

2.2.1 Vulnerability

Vulnerability in software has been described in many different ways. An extract from some of the prior research has been provided below for reference:

"Software Vulnerability is a type of defect or weakness in the system that

makes it susceptible to attacks"(Alhazmi, Malaiya, & Ray, 2007).

"a flaw within a software system that can cause it to work contrary to its documented design and could be exploited to cause the system to violate its documented security policy"(Rehman & Mustafa, 2009).

Since our primary data source is NVD and we use Common Vulnerability Enumeration (CVE) as an essential part of this research, we follow the definition provided by MITRE.org which says "vulnerability is a mistake in software that can be directly used by a hacker to gain access to a system or network"(MITRE)[1].

2.2.2 Static and Dynamic Analysis

Design vulnerabilities are defects in the software that are introduced due to incorrect programming practices and errors, such as integer overflow or buffer overflow. Static and dynamic analysis tools help to identify and generate a list of potential vulnerabilities within software code. Static analyses or white-box testing is performed by examining the source code or the binaries of the software. Dynamic analyses or black-box testing is performed on the software application when it is installed and operational, using a vulnerability scanner tool(Aggarwal & Jalote, 2006). Both have pros and cons, resulting in different levels of usage and effectiveness. For static scanners to be useful the software code has to be available, implying that the software product has to an open-source product or built in-house. Dynamic scanner tools are fast and require less effort for analysis since the scanner report provides vulnerability information. However, scanners require large amounts of time to run and enough test cases to detect these. Both methods

[1] http://cve.mitre.org/about/terminology.html

suffer from issues relating to the generation of false positives and negatives. Also, as discovered by prior research (Fonseca, Vieira, & Madeira, 2007), no one scanner can detect all the vulnerabilities. As a result, several papers have proposed different ways to integrate static and dynamic software analyses (Aggarwal & Jalote, 2006; Ruoyu, Shiqiu, Zhengwei, & Haibin, 2011). In spite of these drawbacks, vulnerability scanners are valuable tools for detecting security flaws in the software and have been proposed for quality as well as risk assessment in prior research. Wagner proposed a framework to integrate vulnerability scanners with quality assessment (Wagner, 2011) through the use of the Bayesian Net, developed using the goal-question-metric(GQM) approach. Bau et al. in their evaluation of the scanners, have found that the effort spent and test cases undertaken by scanners are proportional to the vulnerabilities existing in the current situation and these scanners are adept at finding historical vulnerabilities(Bau, Bursztein, Gupta, & Mitchell, 2010). In our research, we selected dynamic analysis as our scope since we wanted to develop a process that was automatic, required less effort, and would be applicable to in-house, COTS, and OSS evaluations.

2.2.3 CVE

Common Vulnerabilities and Exposures(CVE) are publicly available standard identifiers for vulnerabilities reported in the national vulnerability database.(MITRE, 2011a). Each CVE Identifier is composed of an identifying number and a description of the reported security vulnerability or exposure along, with references to vulnerability reports and advisories[2]. A vulnerability reported in the NVD has a unique CVE assigned

[2] http://cve.mitre.org/cve/identifiers/index.html

to it. The CVE initiative, which started in year 1999 is funded by the U.S Department of Homeland Security. MITRE Corporation serves as the editor of the CVE list. An example of a CVE Identifier is as follows: CVE-2012-0001.

2.2.4 CWE

Common Weakness Enumeration [3] (CWE) refers to publicly available categories of software weaknesses maintained by the MITRE Corporation. As defined in CWE website, "Common Weakness Enumeration (CWE™) is a formal list or dictionary of common software weaknesses that can occur in software's architecture, design, code or implementation that can lead to exploitable security vulnerabilities"(MITRE, 2011b)[4]. The CWE taxonomy is organized in a three-tiered hierarchy and used by analysts to assign scores to the NVD vulnerabilities. Some of the NVD data have been assigned CWE type and help compare various approaches of security risk categorization. Even though there are hundreds of CWE types, only 19 have been used to categorize the NVD data (as shown in Appendix B).

2.2.5 CVSS

The Common Vulnerability Scoring System [5] is a standardized procedure used for scoring the vulnerabilities in the NVD data. It is supported by FIRST (Forum of Incident Response and Security Teams[6] and provides an independent method of evaluating the impact and severity of a vulnerability. Each NVD data has a CVSS score, which is calculated from three metrics: the base metrics, temporal metrics, and environmental

[3] http://cwe.mitre.org/
[4] http://cwe.mitre.org/about/faq.html#A.1
[5] http://www.first.org/cvss
[6] http://www.first.org/

metrics.

2.2.5.1 Base Metrics

The base metrics are independent of the changes with regard to the time and the environment. It is composed of six sub-metrics:

Access Vector: Access required for the attack. Values: Local, Adjacent Network, Network(Peter Mell, 2012).

Access Complexity: Difficulty of the attack based on the existence of a specialized condition. Values: High, Low, Medium(Peter Mell, 2012).

Authentication: Number of authentication attempts. Values: Multiple, Single, None(Peter Mell, 2012).

Confidentiality Impact: Impact on information disclosure. Values: None, Partial, Total(Peter Mell, 2012).

Integrity Impact: Impact on system integrity. Values: None, Partial, Total(Peter Mell, 2012).

Availability Impact: Impact on system or resource availability. Values: None, Partial, Total(Peter Mell, 2012).

2.2.5.2 Temporal Metrics

As the name indicates, the value of these metrics change over the lifetime of the vulnerability. It is composed of three sub-metrics:

Exploitability: Measure if exploit code is available. Values: Unproven, Proof of Concept, Functional, High(Peter Mell, 2012).

Remediation Level: Measures level of fix available. Values: Official Fix, Temporary Fix, Workaround, Unavailable(Peter Mell, 2012).

Report Confidence: Measures the confidence level or credibility of the reported vulnerability. Values: Unconfirmed, Uncorroborated, and Confirmed(Peter Mell, 2012).

2.2.5.3 Environmental Metrics

These metrics represent the characteristics of the environment in which the software product exists. It is composed of three sub-metrics:

Collateral Damage Potential: Measures loss of physical or property damage. Value: None, Low, Low-Medium, Medium-High, High(Peter Mell, 2012).

Target Distribution: Value: None, Low, Medium, High(Peter Mell, 2012).

Security Requirements: Customized score based on the affected IT assets. Value: Low, Medium, High(Peter Mell, 2012).

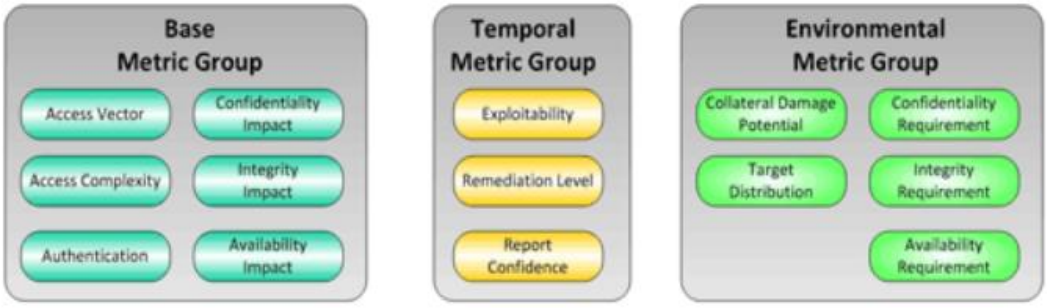


Figure 2-4. CVSS Metric Groups (Peter Mell, 2012)

3 Literature Review

This chapter aims to present an overview of prior work related to this research. This research deals with the intersection of two areas: design of quantitative security risk assessment model and quantitative evaluation of a software product's security risk. A literature review was completed in the areas of quantitative security risk assessment and software selection to investigate what had been done to discover potential opportunities and initiate further research.

3.1 System Engineering and Risk Assessment

Risk Assessment is an important part of System Engineering(SE) due to its impact on reducing the "total lifecycle cost of the system acquisition"(Kujawski & Angelis, 2010). It is an overarching process and applies to every phase of system engineering. The primary purpose of risk assessment in SE is to answer the questions "What can go wrong? What are the associated likelihoods? and What are the consequences?"(Kujawski & Angelis, 2010; Stanley Kaplan, 1981). As pointed out in INCOSE handbook, "Incorporate risk assessment early in the planning process to identify areas that need special attention or contingencies". This relates to the motivation of our research "to provide a procedure that can be used to perform quick assessment during the early stages of the software selection process"(Ruma Das 2012).

As illustrated by Haimes (Sage & Rouse, 2009), the holistic approach to total risk management from SE perspective is multidimensional and complex process as shown by Figure 3-1 below.

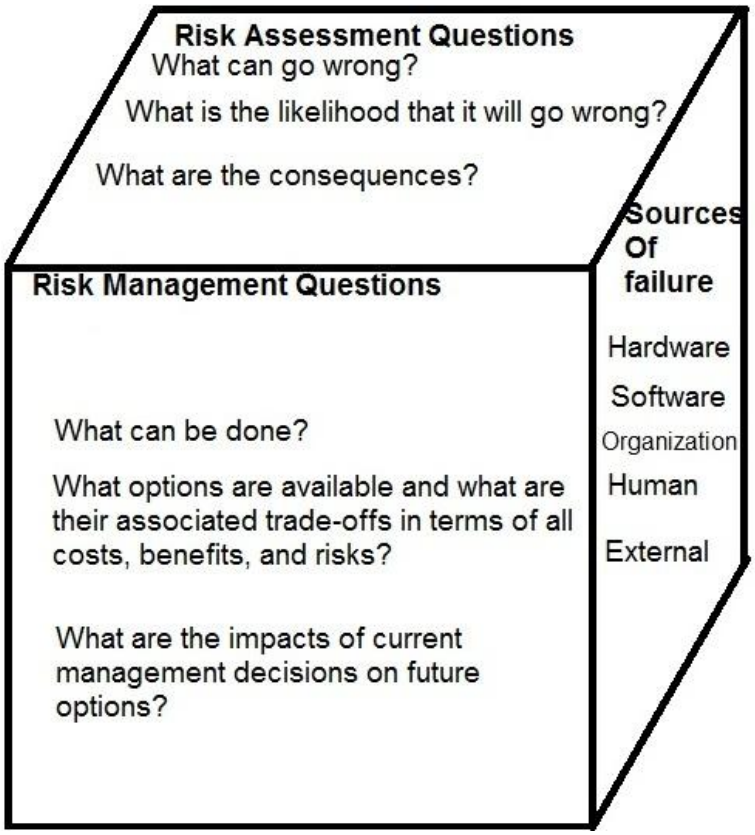


Figure 3-1.Total Risk Management (Haimes)(Sage & Rouse, 2009)

In SE, risk can arise from different sources and each source needs a structured methodology for analysis and evaluation, which can qualitative, quantitative or both. The aspect we seek to research with respect to "total risk management" figure above is the cross-section between "risk assessment and "software risk". Our research aims to provide dimensions and steps to perform quick, first-order security risk assessment during software selection.

As per MITRE's System Engineering guide (MITRE, 2012), the fundamental steps of risk management can be divided into "Risk Identification, Risk Impact Analysis, Risk Prioritization Analysis , Risk Mitigation, Planning, Implementation and Progress Monitoring", all of which are supported by 'Risk Tracking" step, as shown in Figure 3-2.

With regards to scope of our research, our goal is to support step 1 and step 2 in the area of security risk assessment of a software product during evaluation.

Figure 3-2. Steps in Risk Management (MITRE, 2012)

3.1.1 Risk Identification

Risk identification process help to identify and prioritize risk events that can impact the realization of a successful system(INCOSE, 2003). The methods used for this purpose are brainstorming approach like SWOT analysis, scenario analysis, Delphi techniques, historical data analysis and process mapping(Shi, 2004). Most of these methods, except historic data analysis, are subjective and based on the experience level of the analyst. Historic data analysis approach holds great potential to support this step by bringing forth

events that have already occurred in past.

3.1.2 Risk Assessment

Assessment of the risk is performed to determine the impact and the probability. Expert Interviews(INCOSE, 2003) and historic data analysis(Shi, 2004) are common techniques used for this purpose. Historic data may assist in providing quantitative value, but expert judgment is needed to validate the relevance(Shi, 2004).

3.1.3 Risk Prioritization and Handling

This is the step 3 and 4 of the Figure 3-2 and its goal is to support appropriate allocation of resources(MITRE, 2012). The impact and probability value identified during risk assessment step, help to determine the priority of the risks. This step may be performed using Qualitative techniques, Quantitative techniques or mixture of both. As described in MITRE system engineering guide, examples of "qualitative techniques include analysis of probability and impact, developing a probability and impact matrix, risk categorization, risk frequency ranking (risks with multiple impacts), and risk urgency assessment. Examples of quantitative techniques include weighting of cardinal risk assessments of consequence, probability, and timeframe; probability distributions; sensitivity analysis; expected monetary value analysis; and modeling and simulation"(MITRE, 2012). Risk mitigation step define the actions planned to handle the risk. These can be summarized as: accept, avoid, mitigate/control, transfer, and watch.

3.1.4 Risk Tracking

The overarching step for the entire risk management process is 'risk tracking'. The risks list must be continuously monitored, updated and mitigation approaches

implemented during the course of the project.

3.1.5 Risk Assessment Challenges

In SE, number of models and methods exists that are applicable and can be extended to security risk assessment (Wright, 2011) such as Monte Carlo, System Survival Calculations, Reliability functions, non-homogeneous Poisson process (NHPP) etc as explained in paper by Wright et al. For example, the concept of 'Vulnerability Discovery Models' is based on the equation developed for reliability models. The challenge with risk assessment in SE is to make it comparable and repeatable(Hall, 2011). Hall proposed a risk assessment methodology where a set of predefined risk statements, likelihood values and consequence levels were provided to the system engineers. As per Hall, even though this procedure made the process "comparable and repeatable", 'subjectivity' is still a major issue with risk assessment(Hall, 2011). Emphasizing the quantitative value in his procedure, Hall points out that "for risk assessments to mean something—to be value-added—the numbers or levels developed do not need to be absolutely related to anything. They only need to be related to each other in an ordinal sense"(Hall, 2011). The use of empirical data can help reduce some of this 'subjectivity', if we can find a method to extract the risk groups, their likelihood and their consequences from the historic data. One of the goals of this research is to explore how we can use historic vulnerability data for this purpose.

3.2 Quantitative Security Risk

Quantifying security risk is a difficult problem. Even though much work has been done, it still remains a challenge due to changing landscape of engineering systems and

the difficulty in defining metrics. Verendel performed a critical survey of research focused on quantifying security in operational/realistic settings (Verendel, 2009) and described the challenges that made this difficult. Verendel points out that one of the problems in quantitative security research is that the assumptions are not tested in operational settings and the comparison between methods is not performed. Some of his key findings were that the traditional measures of information security, such as "confidentiality, integrity, and availability" (Peter Mell, 2012) were less represented in quantitative security research, implying this may be an area seen as being qualitative in nature. According to Verendel, the Vulnerability Detection Model (VDM) is one area where empirical data has been used to a great extent to validate the models by multiple researchers and thus has gained significant improvement.

In terms of the systems, research has focused on different aspects, such as network, software systems, or the entire information system. Many of the procedures and measures applicable to the network are applicable to software as well. Ahmed et al. have proposed three measures for quantitative security risk evaluation at the network level. The first measure is to measure the risk based on past data for a network service termed the "Historical Vulnerability Measure". The second measure is called the "Probabilistic Vulnerability Measure", which measures the potential of a vulnerability to be discovered in next internal. Finally, the "Existing Vulnerability Measure", provides an indication of the risks due to unpatched vulnerabilities existing in the system. (Ahmed, Al-shaer, Taibah, & Khan, 2011)

In another research related to operational security, Ouedraogo et al. developed a quantitative framework for measuring security assurance in order to determine the

confidence of deployed security controls(Ouedraogo, Khadraoui, Mouratidis, & Dubois, 2012). Their security assurance metrics were developed by understanding the fact that there were practitioners with and without knowledge of security exposure. This was an important aspect of security metrics development, since for the metrics to be usable, it had to be comprehended by its practitioners and applicable to the problem's context (J. Bayuk & Mostashari, 2012).

Secure design principles have been proposed to make the software secure through a proactive inclusion of security risk measurement activities during the entire software development lifecycle (Khan, 2011) (Rehman & Mustafa, 2009). However, during the selection and implementation of a commercial off-the-shelf (COTS) or open source system (OSS), such opportunity may not be available. Therefore, it is more critical that security be evaluated before implementing the software in an operational setting. As pointed out by Hissam et al. every software product not built in-house is a "potential source of vulnerability" (Scott A. Hissam, 1998) which is why security risk assessment is important. This fact was also emphasized by Schneidewind, who pointed out that an insecure COTS product could be a weak link in the system environment (Schneidewind, 1998) and impact the reliability of the entire system.

Security flaws make the software vulnerable to attacks. Quantitatively measuring this flaw may provide methods to manage the software development process, evaluate software, patch the process, and undertake overall security risk management. There are methods and metrics to quantitatively predict and assess the reliability of the system by analyzing past defects in similar software or the current defects in existing and developing software. Vulnerability is a type of defect that makes the software weak

against threats. However, the literature review indicates that quantitatively assessing the security risk is a relatively new research area. The primary reason for this is the availability of empirical data. Some research efforts have used publicly available databases for collection of empirical data to design quantitative risk models. Others have used experiments, elicited expert judgment (Ryan, Mazzuchi, Ryan, Lopez de la Cruz, & Cooke, 2012), undertaken surveys (Karabacak & Sogukpinar, 2005), and written observations to collate empirical data.

3.2.1 Research using Public Data

There are a number of public data sources for vulnerability data, such as NVD[7], OSVBD[8], SECUNIA[9], Bugtraq[10], X-FORCE [11]etc. Numerous researchers have attempted to compare them, evaluate differences, and integrate them (Arnold, Hyla, & Rowe, 2006; Gu & Li, 2010). In a study to evaluate the quality of these public data sources, Massacci et al. concluded that the prediction power of the studies might be impacted by missing data and therefore, proposed the use of multiple data sources (Massacci & Nguyen, 2010). However, there exists no known solution for this issue other than their integration as suggested by some studies (Arnold et al., 2006; Gu & Li, 2010). Tripathi et al. compared the various vulnerability databases to analyze the attributes that could be used for taxonomic analysis and found that different descriptions or values were used for impact and severity metrics in different databases. Therefore, the quantitative analysis by combining the information from different vulnerability databases proved challenging

[7] http://nvd.nist.gov/
[8] http://www.osvdb.org/
[9] http://secunia.com/
[10] http://www.securityfocus.com/
[11] http://xforce.iss.net/

owing to different classification schemes(Tripathi & Singh, 2011).

In terms of public data, the National Vulnerability Database(NVD) data has been used in many of the prior studies, to predict trends (Neuhaus & Zimmermann, 2010) and develop vulnerability discovery models (Ozment, 2007) (Austin & Williams, 2011) (Woo et al., 2011). The NVD has also been used for trend analysis either by grouping them based on individual attributes, such as frequency, severity etc. (Yung-Yu, Zavarsky, Ruhl, & Lindskog, 2011) or by using topic modeling algorithms (Neuhaus & Zimmermann, 2010). We selected the NVD for our research, since the different public data sources complemented each other and there was no one best source. The NVD had the necessary data attributes for the creation of our model.

Scholte et al. analyzed the vulnerability trends in web application, with a focus on SQL injection and cross-site scripting vulnerabilities (Scholte, Balzarotti, & Kirda, 2012). "This research showed that complexities in these vulnerabilities did not increase over time. Most of these vulnerabilities occurred due to simple reasons, such as lack of input validation"(Scholte et al., 2012). In addition, in a study aimed at evaluating different vulnerability scanners, Bau found that they were adept at finding historical vulnerabilities (Bau et al., 2010) but inept at discovering second--level vulnerabilities, such as cross-channel scripting. Thus, calculating the probability of occurrence from historic data and using vulnerability scanners on web applications would be a feasible option for evaluating our study procedure.

Neuhaus et al. performed similar trend analysis using the NVD data. Their research used the concept of the 'topic model' to group the vulnerabilities (Neuhaus & Zimmermann, 2010). Post grouping, the researchers undertook manual annotation and

performed trend analysis. The authors found that the topics that were formed were similar to the Common Weakness Enumeration (CWE) classification undertaken by MITRE, which indicated the viability of using topic models to group vulnerabilities.

In another study involving a similar trend analysis using the NVD, Yung-Yu et al. investigated the trends of 15 CWE vulnerability types, with respect to frequency, severity, and the base metrics. The goal of such trend analyses is to predict threats, focus security risk mitigation strategies, and develop polices for relevant emerging issues (Yung-Yu et al., 2011).

Frei et al. analyzed the trends related to zero-day vulnerability and patching time, using a number of open source databases (Frei, May, Fiedler, & Plattner, 2006). The authors developed the vulnerability lifecycle model, using the dates in the public database, namely, discovery date, disclosure date, and exploit date. This study proposed a quantitative method to estimate the exposure levels of the system based on the gap between exploit and patch availability.

Houmb et al. utilized the NVD data to develop a risk estimation model for controlling the network risks. Instead of using the NVD-based CVSS score, they estimated the frequency and impact of the vulnerability using the attributes of the base metrics, the temporal metrics, and the environmental metrics of the NVD entry (Houmb et al., 2010). The frequency is estimated using the three attributes from the base and temporal group metrics, whereas the impact is calculated using three attributes from the base metric group and four attributes from the environment metric group. The values are fed as posterior probability into a BBN (Bayesian Belief Network) model built using the

HUGIN[12] tool. The final risk level of the system has been derived from these estimates. The authors discussed a real industry case study to demonstrate the evaluation of the security risk level assessment of a network. The procedure followed by Houmb et al. was: "Identify vulnerabilities and potential fault introduction sources, then estimate frequency and impact of vulnerabilities using CVSS, and finally derive risk level from frequency and impact estimates"(Houmb et al., 2010). Our research used a similar procedure but this applied to a software product and used the web application scanner for extracting the vulnerabilities.

Mkpong-Ruffin et al. designed a "software security risk assessment model" (Mkpong-Ruffin et al., 2007) using vulnerability data from national vulnerability database (NVD). Their model was based on concept of the 'annual loss expectancy (ALE)' calculation and inspired by the insurance industry. They used the K-means clustering algorithms to group the vulnerability and calculated the loss expectancy of each group. The model aimed at providing a method to prioritize the security risks in the design phase of the SDLC so that practitioners could estimate where to apply their attention and modify the design. We aimed at applying such a model to software products in an operational setting or a product not developed in-house, such as an OSS.

Karabacak et al. also used ALE to provide a quantitative method for risk analysis (Karabacak & Sogukpinar, 2005). Their model "Security Risk Analysis Method (ISRAM)" (Karabacak & Sogukpinar, 2005) is based on preparation of survey and conducting the survey with IT staff and users to determine incident rates. The system

[12] http://www.hugin.com/

entails obtaining probability and the consequences of the security breach through the survey and then, evaluating the security risk score using the ALE formula. However, such a survey procedure is sometimes affected by subjective bias and thus, historical data provides a stronger evidence of the probabilities.

In another research, Mendes et al. proposed a benchmarking strategy to compare systems and components. The security risk was calculated based on the attributes 'exploitability' and 'impact' of the vulnerabilities found within the internal components of the system. The relative weight of the component was also considered. (Mendes, Duraes, & Madeira, 2011). The risk of the vulnerability detected is calculated as a product of the impact and the exploitability levels. A component (OS, app server, database server etc.) may have a large number of vulnerabilities and thus, the total security risk of a component is the sum of all its vulnerability risks. Finally, the total security risk of the system is calculated by summing the individual component risks, taking into considering the weighted importance of each component.

Related to the area of security risk assessment, are the quantitative models for security risk predictions such as Vulnerability Discovery Models(VDM) (Ozment, 2007). As defined by Woo, "Vulnerability discovery models allow prediction of the number of vulnerabilities that are likely to be discovered in the future"(Woo et al., 2011). VDM's have been inspired by the software reliability theory, with the goal "to estimate the total number of vulnerabilities present in the software"(Nguyen & Massacci, 2012) using historical data.

Another usage of the NVD data has been to predict vulnerability occurrence and discovery security metrics(Alhazmi et al., 2007). Alhazmi et al. define two metrics

“vulnerability density” and “vulnerability discovery rate” to describe the vulnerability characteristics of a product. The “vulnerability density” calculation is based on the number of vulnerabilities and lines of code where as the “vulnerability discovery rate” is based on the discovery date and the number of vulnerabilities. Their goal is to develop a quantitative model similar to the work done in the reliability field.

Vache et al. developed a vulnerability lifecycle model using the Discovery date, the Disclosure date, and the Patch date from the OSVDB database (Vache, 2009). The purpose was also to discover the vulnerability categories for building a “quantitative security evaluation model”.

In another related research, Schryen et al. developed he software metrics ”Patch index” (Schryen & Kadura, 2009) to compare the security of the open source versus closed source software. They used NVD data to categorize vulnerability into three classes (high, low, and medium) and then calculate the patch index which is defined as “the number of patched versus un-patched vulnerabilities” (Schryen & Kadura, 2009).

In the area of decision support, Fruhwirth et al. proposed a security model where the temporal and environmental context information was provided by the practitioners, which allowed for a recalculation of the CVSS score and a re-prioritization of the vulnerabilities. The purpose of this research was to allow for resource and effort management through prioritization (Fruhwirth & Mannisto, 2009).

Chung et al. developed a set of metrics called ‘data Rick Vector’ and ‘Application Risk Vector’ to perform security risk assessment for IT assets (Chung et al., 2005). The security risk level for data and application was calculated as follows:

Data Risk vector: $V_D = \sqrt{V_C^2 + V_I^2 + V_A^2}$ (Chung et al., 2005)

Application Risk Vector: $V_{AP} = \sqrt{V_I^2 + V_A^2}$ (Chung et al., 2005)

The values used for each of the categories (V_C *(Confidentiality)*, V_I *(Integrity)*, and V_A *(Availability)*) were high, low, and medium, with a numeric custom range assigned for each level.

Teng et al. proposed measuring the trustworthiness of the system using "Assets, Vulnerabilities, and Trustworthiness" where the vulnerability score was proposed to be derived from CVSS, the asset score derived using the subject/domain expert's scoring, and trustworthiness derived using "weight of actor and attacker" (Teng, Lin, Yijun, & Zhi, 2009).

Another research used NVD data to develop tool to support patching, mitigation, and other security management areas (Ju An, Linfeng, et al., 2010). In this research, An et al. used Ontology for Vulnerability Management (OVM) and NVD data to determine similarities between different vulnerability records using a hierarchical similarity algorithm. The attributes considered for the similarity algorithm were "Type, Product, Access Vector, Access Complexity, Authentication, Confidentiality Impact, Integrate Impact and Availability Impact"(Ju An, Linfeng, et al., 2010).

Other than CVSS, VRSS is a scoring mechanism put forth by Liu and Zhang, since they found that the distribution of vulnerabilities was not normal based on a statistical analysis of three open source databases:"IBM ISS X-Force, Vupen Security, and National

Vulnerability databases"(Q. Liu & Zhang, 2011). In a study of the CVSS score formula, Ibidapo et al. pointed out a flaw in the CVSS scoring that made it negative, and due to the impact of the environmental metrics on the base score, it was not taken into consideration (Ibidapo, Zavarsky, Lindskog, & Ruhl, 2011). In another study that used simulation to evaluate the impact of different environmental metrics on the CVSS score, the authors found that the environmental metrics' effect was not appropriate. Due to this, the CVSS score did not allow practitioners to see the differences based on the security levels of the organization and therefore, did not enable the development of a strategy for IT risk management based on the score (Ibidapo et al., 2011).

3.2.2 Models using Custom Data

The attack tree-based quantitative evaluation was proposed by Grunske et al. to quantitatively evaluate the security of component-based systems in the design phase using SysML. The proposed procedure used modular attack trees indicating how a system could be attacked, the attacker characteristics, asset costs, and the probabilities of basic events indicating how the attackers exploited vulnerabilities (Grunske & Joyce, 2008). However, no empirical data was used to predict these probabilities. It was noted by the author of this paper (Grunske & Joyce, 2008). that it was difficult to assess the asset value. The components in this case were servers, workstations, network host, user etc.

In another study involving quantitative analysis of security, the authors put forth the concept of "survivability," which was defined as the "probability to meet security objective". An e-business application was evaluated using Microsoft's STRIDE model to identify threats and sequences of messages sent in the system. Survivability scores and security measures were then used to calculate the risks. The goal of their procedure was

to determine risks during the early stages of software development (Youn, Park, & Lee, 2011).

Dwaikat et al. presented a method of evaluating the risk of a software system based on an analysis of its components, transactions, and people (Dwaikat & Parisi-Presicce, 2005). In their example, they divided the systems as web server, database server, browser, and app server. Each of these components was assigned a risk value based on integrity and confidentiality risks. The design of each component was evaluated against the security policy to determine the risk levels. If the risk level was not acceptable, the design was adjusted until the risk values were acceptable. The authors mentioned that the risk values were based on "information types of channels, software vulnerabilities, IT personnel, and context" (Dwaikat & Parisi-Presicce, 2005).

Chowdhury et al. investigated the correlation between the software code metrics and software vulnerability using the information from the Mozilla Foundation Security Advisories (MFSA) in combination with Bugzilla ("Bugzilla,") and found significant correlation with some of the code metrics (Chowdhury & Zulkernine, 2011). The purpose of this research was vulnerability prediction of a software product based on code metrics, using classification methods, such as decision tree, random forest, logistic regression, and Naive Bayes.

In another approach, Wache et al. proposed a quantitative security evaluation and vulnerability classification based on "vulnerability lifecycle ," given by three stages: "discovery, disclosure, and exploit" (Vache, 2009). The authors studied the probability distribution of these three vulnerability lifecycles and found that it had 'beta distribution.' As per the author, the security evaluation should take into account "the vulnerability

lifecycle , the behavior of the attacker and the system administrator" (Vache, 2009).

3.2.3 Framework, Metrics and Measurement Models

With regard to our first research question, "How can we quantitatively assess the security risk of software and how do we quantify software security Risk?" we analyzed prior research in the areas of security framework, metrics, and measurement models. Providing clarity in this area, Verendel pointed out the relationship between measurement, metrics, and the model by saying that "measurement is made by observing outcome of an event" whereas metrics could be defined as "an measurement-based indicator , assigned to a scale for measuring some security attribute"(Verendel, 2009). He defined a 'model' as a formal mathematical representation derived using the security attributes and 'framework' as "methods of performing the quantitative analysis"(Verendel, 2009).

Researchers Schryen et al. have identified several drawbacks with security measurement research and propose new metrics for the same (Schryen & Kadura, 2009). These authors observe that most of the security measurements have been derived from the fields of reliability and dependability. In addition, the empirical analysis in this area has been focused primarily on the operating systems and browsers. We propose to fill some of these gaps in the field of security assessment for software selection. Schryen et al. proposed new metrics namely, 'Cumulated weighted vulnerability within time window t' and 'Patch index at time t' to compare the open source product 'openoffice' with a closed-source product 'Microsoft office.' As per the authors, it is not only the number of vulnerabilities, but how quickly they are patched that indicates the quality of the product.

The purpose of metrics in system engineering is to control the process and measure

the outcome (Vanek, Jackson, & Grzybowski, 2008). Vanek et al. summarized the characteristics that made metrics and their development effective in the field of system engineering and emphasized the following common characteristics: simplicity, relevance, and repeatability. In our study, we propose that developing the quantitative risk model by clustering historic data and deriving metrics for risk assessment from the model makes it repeatable. Developing metrics to solve a specific problem, in a specific environment makes it 'relevant.' 'Simplicity' is the hardest characteristic to satisfy. Similar views have been presented by Breier et al. who summarized the purpose, benefit, and quality of good metrics as something that was "cheaper and easier to gather, context specific and relevant to the decision maker" (Breier & Hudec, 2011).

In areas of security metrics, development and design metrics have been analyzed by several researchers. However, as Savola pointed out, the field of security metrics is still in its infancy and provides a procedure to validate them (Savola, 2010). As per the author, the purpose of metrics is to support decision-making rather than provide an absolute number. NVD has been a popular choice for developing security metrics. Wang et al. derived security metrics for ranking attacks on a software product (Ju An, Hao, Minzhe, Linfeng, & Camargo, 2010). They performed their study using a web browser (e.g. Mozilla Firefox) as a software product and deriving the vulnerabilities using OVM. These vulnerabilities were classified using the 14 CWE types. Each of the CWE types was ranked using their custom equation in which 'the time of discovery' was weighted as 'past, present and recent', with weights assigned in an ascending order. These were then combined with the number of CVEs found in that type along with the CVSS score, using the custom equation to derive the rank. In another study to rank vulnerabilities, the

authors suggested using context information based on the importance assigned to the CIA factors by organizations, and re-scoring the CVSS score(Fruhwirth & Mannisto, 2009). This meant that even if vulnerability had a high confidentiality impact, the 'confidentiality' factor had a relatively low value in terms of its overall importance to the organization, which was why the rank of that risk was low.

Brian derived the metric for measuring security during software development lifecycle using automated source code analysis techniques (Brian, 2005). As per the author, detecting vulnerability could indicate any improvement in the software security through measuring metrics, such as "vulnerability dwell, total number of issues, number of issues reviewed, and number of vulnerabilities identified as a function of time"(Brian, 2005). Wang et al. provided a procedure for evaluating the security risk score of software products like browser, web server, database server etc. using NVD information (Wang, Wang, Guo, & Xia, 2009). Ryan et al. developed a probabilistic model to determine successful attacks during security incidents using the method of expert judgment elicitation. In this research, the quantitative information regarding security risk values was obtained using survey instruments and expert judgment elicitations (Ryan et al., 2012). In general it was noted through literature review that in the area of quantitative security risk assessment however, the challenge lies in the identification of the metrics or attributes for risk measurement, defining relevance to the context, and proving validity in real case scenario(Verendel, 2009).

3.2.4 Identifying Security Risk Groups

Standardizing vulnerability from historic data and across web application-scanner categories (Venter et al., 2008) has multiple advantages, one of which is of interest to us:

comparing software products by measuring the output of the vulnerability scanner with the quantitative risk assessment model. Creating such a model from historic data provides a better estimate of risk probability and impact.

The first task in building a security risk model is to classify or cluster the data obtained from the vulnerability database. The literature review demonstrates several methods using supervised and unsupervised machine learning algorithms. Many of the studies in the field of software vulnerabilities performed their analysis to obtain semantic groups, using the 'summary' field from each entry of the NVD data repository. We used the same for our text-mining process, along with other non-textual fields.

Mkpong-Ruffin et al.(Mkpong-Ruffin, 2009) (Mkpong-Ruffin et al., 2007) performed clustering of NVD data to create quantitative security risk assessment model, using K-means and the Expected Maximization algorithm. The clusters in this model were found to be statistically significant for vulnerability type, loss type, and exploit range. Their research used the quantitative model to analyze the output of Microsoft Threat Analysis & Modeling tool (Microsoft, 2012) and evaluate the security risk of the software product, during the design phase of software development lifecycle.

Chen et al. categorized the CVEs using the Support Vector Machine (SVM) algorithm, and used the summary field of the NVD for text mining (Zhongqiang Chen et al., 2010). The purpose of this research was to build a CVE classifier for automatic classification and generalization of CVE data. The taxonomic features used for classification were vulnerability cause, impact, exploit location (local/network), and severity. The categories for vulnerability impact were derived from Secunia and X-force as shown in Table 3-1 below.

Table 3-1. Impact categories - Secunia & X-force

Secunia	X-Force
Brute force Cross-Site Scripting DoS (Denial of Service) Exposure of sensitive information Exposure of system information Hijacking Manipulation of data Privilege escalation Security Bypass Spoofing System access Unknown	Gain Access, Gain Privileges, Bypass Security, File Manipulation, Data Manipulation, Obtain Information, Denial of Service, Configuration, Informational, Other, None

The categories for 'vulnerability cause' were derived from 'class' attribute of vulnerabilities reports in BID[13] as shown below.

Table 3-2. Vulnerability Cause categories - BID

"Configuration Error, Boundary Condition Error, Environment Error, Input Validation Error, Design Error, Race Condition Error, Origin Validation Error, Access Validation Error, Failure to Handle Exceptional Conditions, Atomicity Error and the Unknown" (Tripathi & Singh, 2011).

SVM is a supervised learning algorithm in which the model is trained to classify data based on set of predefined groups and then, used to classify new data into those groups. The advantage of using supervised learning algorithm is that it can be easily used for pattern classification based on known labels. However, this makes the grouping rigid,

[13] http://www.securityfocus.com/bid/

since data is forcibly classified into predefined categories. In addition, the determination of labels is important and in a large dataset, this might be prohibitive. It does not allow natural grouping to be discovered and its application is limited in areas where groups of data can belong to multiple categories. However, precision and recall rates have been found to be good in some cases.

In contrast, unsupervised learning algorithm like SOM, are used to identify hidden groups in the data. Venter et al. proposed self organizing maps(SOM) to standardize vulnerabilities, using historical data from the national vulnerability database (Venter et al., 2008). The 'summary text' field from NVD is converted into numeric vectors and fed into the SOM algorithm for clustering. A prototype was shown to demonstrate the use of the algorithm. However, the research did not provide the methods used to verify the precision and recall rates of the results of the clustering algorithm.

The advantage of using unsupervised machine learning algorithm is that the hidden patterns can be discovered from a large amount of data. The disadvantage of unsupervised machine learning algorithm is that the groups are sometimes too granular and not mutually exclusive. In addition, labeling the clusters found is a time consuming and expert-oriented activity.

Shuguang et al. proposed a custom text-based clustering research using NVD data for vulnerability categorization and taxonomy development (Shuguang, Heping, Min, & Jie, 2010). They initially evaluated text-based clustering algorithms, such as "Simplekmean, BisectingKMeans, and BatchSom" using the DIAsDEM Workbench (Shuguang et al., 2010) to classify NVD data into 45 clusters. Having discovered the disadvantages in these traditional clustering algorithms, they developed their own text-mining and

clustering processes. Their goal was to avoid overlaps and they proposed metrics to choose the dominant cluster based on 'cluster quality,' which was calculated using several parameters proposed by the author, such as descriptor dominance, descriptor coverage, and relative size of cluster.

Liu et al. proposed an automated architecture based on text-mining and analysis of discovered network vulnerabilities (C. Liu, Li, & Chen, 2012). For text-mining they suggested an ensemble of classifiers, such as K-Nearest Neighbor, Naïve Bayes, Support Vector Machine, and Keyword Search.

3.3 Software Selection

The objective of this section of the literature review is to gather security metrics suggested in the existing literature for the evaluation and the selection of OSS software. The purpose is to investigate how the existing metrics support our goal of creating a quantitative security risk assessment framework.

Open Source System (OSS) has many similarities to commercial-off-the-shelf (COTS) as well as significant differences (Wheeler). Even though acquisition of COTS has been studied widely in system engineering, similar studies related to OSS are few. Acquisition of OSS comes with its unique challenges due to lack of vendor support for demonstration, training, support etc. These challenges make it even more important to have models and frameworks to enable system engineers assess OSS for success implementation. Authors Salter et al. have put forth a lifecycle based method for commercial-off-the-shelf (COTS) acquisition (Salter & Buede, 2001). However in our review of prior works in the field of system engineering (SE), the application of SE in open source system (OSS) selection, evaluation and integration is lacking. Each of these

areas has been studied in other fields such as information technology, computer science, software engineering etc. Acquisition and Integration of COTS has been studied in many of the SE papers and are applicable in OSS as well. However OSS has significant differences that provide benefits as well as disadvantages. Thus a gap exists in the study of acquisition and integration of OSS based on SE paradigm.

A review of prior works in the fields of COTS and OSS selections was conducted to find methods, frameworks, and metrics created for quantitative risk assessment. It was found that most of the COTS selection methods were based on the requirement engineering process and methods to evaluate the COTS based on the requirement criteria generated from that process. Both, COTS and OSS selection studies have mostly focused around requirements criteria, quality criteria, and decision-making processes around these criteria, such as AHP. Al-Qutaish et al. demonstrated the use of AHP to select OSS (Al-Qutaish, Muhairat, & Al-Kasasbeh, 2009). Hauge et al. point out several reasons why COTS selection criteria may not be sufficient for OSS, essentially because of the lack of marketing entity, lack of information, lack of backing from a company etc. (Hauge, Osterlie, Sorensen, & Gerea, 2009). In the industry environment, Hauge et al. found through an empirical study that, the project-specific constraints and developers' best fit analysis was often used for OSS selection (Hauge et al., 2009). Lee et al. provided an OSS selection process consisting of 5 steps and 19 activities (Lee, Kim, Choi, & Rhew, 2007) which started with gathering user requirements, selecting products based on keywords, analyzing them based on assessment criteria, and then, finally, undertake selection based on weight assignment. This structured process provided a general guideline for OSS selection with each activity having a defined output. Through a

systematic literature review of literature related to software selection, Jadhav et al. accumulated several criteria utilized for this purpose. In this study, of the area of security assessment, the attributes identified were: user identification, data encryption, and auditing (A. S. Jadhav & Sonar, 2011). The authors did not seem to notice vulnerability or security risk levels analysis as criteria for selection, and none of these were quantitative in nature. In another paper, Kennedy mentioned several criteria for evaluating open source products, such as a method of controlling releases, the maturity and size of the open source community, requirements to modify the code, possibility of Maintenance and Support etc. (Kennedy, 2010) . In a similar research, the authors focused on those criteria that were easily available from open source portals or the product's home page when evaluating an OSS product (Polancic, Horvat, & Rozman, 2004). For example, user rating, vitality (last modified date), popularity (Google hits+number of times last viewed), lifespan/duration of the project, URL hits, subscribers, number of developers, number of times downloaded, bug ratio, number of OS supported, status of the project etc. Author Spinellis et al. present a framework for evaluating the quality of OSS, termed SQO-OSS, which includes the criteria for both, the product- and the process-related areas (Spinellis et al., 2009). Security is often mentioned as a quality attribute for OSS selection (Sung et al., 2007). However, many of these models do not provide practical guidance on how to measure security quantitatively or use it for the selection of software. Lack of a systematic procedure and handling of non-functional requirements, such as security, has been cited as common issues found through survey of industry for the COTS selection process (Tarawneh, Baharom, Yahaya, & Zainol, 2011).

Thus, through our literature review, we see a gap in using security risk assessments

for OSS selection. In addition, very few studies focus on the operational and implementation dimensions of COTs or OSS products' evaluations. Insecure products may reduce security levels of the whole system. Morasca et al. evaluated issues surrounding the testing process for OSS and proposed a checklist that could be used for the same (Morasca, Taibi, & Tosi, 2009). The authors emphasized the security testing as paramount for OSS due to the easy availability of the source code. Lavazza et al. analyzed the relationship between perceived reliability and code characteristics for a java program (Lavazza, Morasca, Taibi, & Tosi, 2012). They found that there was a significant correlation between OSS stakeholders' perceptions of OSS reliability and measurable code properties. Even though the emphasis on security assessment existed, and correlation has been demonstrated, the methods to evaluate this using empirical data were found lacking.

Groven et al. compared two quality assessment models for OSS with focus on security measurement. The security metrics from this paper have been repeated here for reference information (Groven, Haaland, Glott, & Tannenberg, 2010). In the OpenBRR framework, there were three security metrics for open source software evaluation as follows (Groven et al., 2010):

1. "Number of security vulnerabilities in the last 6 months that were moderately-to-extremely critical" (Groven et al., 2010)

2. "Number of security vulnerabilities still open (unpatched)" (Groven et al., 2010)

3." Is there dedicated information (web page, wiki, etc) for security?" (Groven et al., 2010)

Each of these metrics had a numeric score assign by detecting the number of vulnerabilities. For example, for metric 1 above, the scoring scale was 5: 0 were found, 4: 1-2 were found, 3: 3-4 were found, 2: 5-6 were found, 1: more than 6 was found.

Another open source quality evaluation framework known as the QualOSS used NVD to define the 9 security indicators(Groven et al., 2010) mentioned below:

"1. Global track record of NVD Entries over time

2. Global track record of High severity NVD Entries over time

3. Global track record of Medium severity NVD Entries over time

4. Predictability of yearly trend of NVD Entries over major releases

5. Predictability of yearly trend of High severity NVD Entries over major releases

6. NVD Entry Status of selected release

7. High severity NVD Entry Status of selected release

8. Track record of NVD Entries for selected minor over time

9. Track record of High severity NVD Entries for selected minor over time" (Groven et al., 2010)

However, these measurements were focused on the product itself rather than the entire system. It did not take into account the potential security risk that might exist within the product due to its design and which may manifest itself during the operations. In addition, if the vulnerability was not reported in the NVD, then, chances were that these measurements would not be able to function.

In summary, evaluation and selection frameworks for COTS or OSS have been

extensively researched over the past few years, but gaps exist in the area of security assessments. Since selection of a COTS or OSS is a multi-criteria decision-making process, methods, such as "Analytic Hierarchy Process (AHP), Weighted Scoring Method (WSM), Hybrid knowledge-based system (HKBS)" (A. Jadhav & Sonar, 2009) etc. have been investigated in prior research. In terms of software product selection procedures, the requirement engineering frameworks, such as Goal-Oriented Requirement Engineering (GORE) (Alves & Finkelstein, 2002) and Procurement-Oriented Requirements Engineering Method (PORE) (Hauge et al., 2009) are predominant. In the area of the OSS, selection frameworks, such as openBRR, QSOS, and QualOSS have also been proposed. The common denominator in these procedures and frameworks is defining requirements, establishing evaluation criteria, and assessing the software products by assigning weighted scores to them. However, not much work related to security evaluation of OSS or COTS, using black-box testing methods, such as vulnerability scanners was found in existing literature. Code metrics have been investigated for OSS, but that is not applicable to a COTS product whose code is not available. Therefore, we proposed an extension to these procedures that focused only on the security risk assessments, during the early stages of software selection.

3.4 Gaps in Research

Recently, scholars have proposed several quantitative security risk assessment models based on vulnerability data extracted from public databases, such as NVD, OSVBD etc. "Even though vulnerability data has been used for predicting exploits, manage patching, evaluate network security risks, it has rarely been applied to software selection. Since vulnerability is a type of defect that reduces the reliability of the system, filling this gap

would help evaluate and select COTS products or custom software systems"(Ruma Das, 2012). Based on our literature review, we find that the existing software selection methods proposed for COTS and OSS selection fall short by omitting steps that are essential for conducting security risk assessments, quantitatively and in operational settings. In addition, we did not find a study that proposed a model for providing security risk scores for a software product based on dynamic analysis and data mining.

3.5 Needs for a New Study

This literature review has investigated prior research in the fields of quantitative risk analysis, quantitative security assessment, and software selection. There was substantial research in the field of quantitative risk analysis and several quantitative security risk models have been proposed (Wang, Guo, et al., 2009), (Mkpong-Ruffin et al., 2007), (Houmb et al., 2010), (Chenmeng Sui, 2012). We did not find a model to perform quantitative security risk assessment during software selection. Hence a new study was needed to extend current research in this field and determine how empirical data available from public databases could help quantitative evaluation of risks for software products during the selection phase.

In the next few paragraphs, using information from our literature review, we compare closely related prior research and indicate how our research is different or how our research extends the other studies.

Wang et al. demonstrated an ontology based security assessment of the software products (Wang, Guo, et al., 2009) such as 'Mozilla Firefox 3' and 'Internet explorer 7'. The goal of their research was to measure the trustworthiness of software and compare similar products based on security metrics. "The questions that we propose in this

research are similar but the software products are different. The software products we plan to evaluate are custom web-based database-driven software applications. The security scores for these products are not likely to be present in CVSS and need to be derived based on categories of vulnerability found. This research plans to fill this gap by proposing a framework for calculating the score of a custom database-driven software system, like a Health management system, Conference management system etc"(Ruma Das, 2012).

In terms of Quantitative Security Assessment framework, our research is similar to model developed during the Ph.D. dissertation by Mkpong-Ruffin et al. (2007), where they proposed predictions of loss expectancy during design time using modeling tool. "The goal of our research is also to build such a quantitative security risk model; but instead of utilization in design phase, we propose a procedure that can be utilized to perform quantitative risk assessment during software product evaluation"(Das, Sarkani, & Mazzuchi, 2012). In addition in terms of using scanners, our research uses procedure similar to that used by Houmb et al., but instead of evaluation of security risk level assessment of a network, we apply it to a software product. Sui et al.,used the vulnerability scanner 'Nessus' to generate a list of vulnerabilities and compare these with their custom quantitative scoring criteria to calculate a security score (Chenmeng Sui, 2012). In our research, we also plan to use the vulnerability scanner, except that we will use the impact score and threat likelihood values derived from NVD data to generate security scores(Das et al., 2012).

Our research is similar to the study by Mendes et al. that proposed a method for selecting functionally similar components (web servers, database servers, OS) by

assessing the security risks based on known vulnerabilities (Mendes et al., 2011). Their research proposed a security benchmark procedure for web serving systems, consisting two parts: a static analysis and a dynamic experiment. The static analysis was undertaken to determine the type and version of the software components, which is then used to extract the impact and exploitability levels from the NVD and OSVBD databases. The dynamic experiment was performed to test the response of the system to various vulnerability related attacks. We proposed applying the results of the dynamic analysis to determine the overall security risks of a web-based database-driven application.

Our work builds on several existing research. In terms of clustering empirical data, we build on the work by Neuhaus et al. in using the topic modeling algorithm to study the security trends of vulnerabilities (Neuhaus & Zimmermann, 2010). In topic modeling, we have used the Latent Dirichlet Allocation (LDA) implementation to study bug localization (Lukins, Kraft, & Etzkorn, 2010), software evolution (Stephen W. Thomas, Adams, Hassan, & Blostein, 2012), and the categorizing of bug reports (Somasundaram & Murphy, 2012). The primary advantage of the LDA is that it is automated and discovers latent themes without any human intervention. The algorithm finds hidden themes in large text corpora and groups them into clusters based on the similarity of the words in the text. "The rationale behind choosing an unsupervised learning problem, such as the LDA is that not all security risk categories have been labeled as per the CWE (Common Weakness Enumeration) category in NVD. Using the LDA, our goal is to discover groups of similar vulnerabilities within the data by clustering algorithms as done by Neuhaus et al. The benefits of using topic modeling for creating a risk assessment framework is that it can be used to evaluate results from multiple different vulnerability

scanners, which produce textual output. Each scanner may have its own description of the vulnerabilities, which can be categorized into topics by using the topic modeling technique. These may then be assessed against the model created using the NVD entries" (Das et al., 2012; Ruma Das, 2012).

4 Research Framework

This chapter aims at describing the framework upon which the research will be founded.

4.1 Conceptual Framework

The conceptual framework of the study is graphically depicted in Figure 4-1.

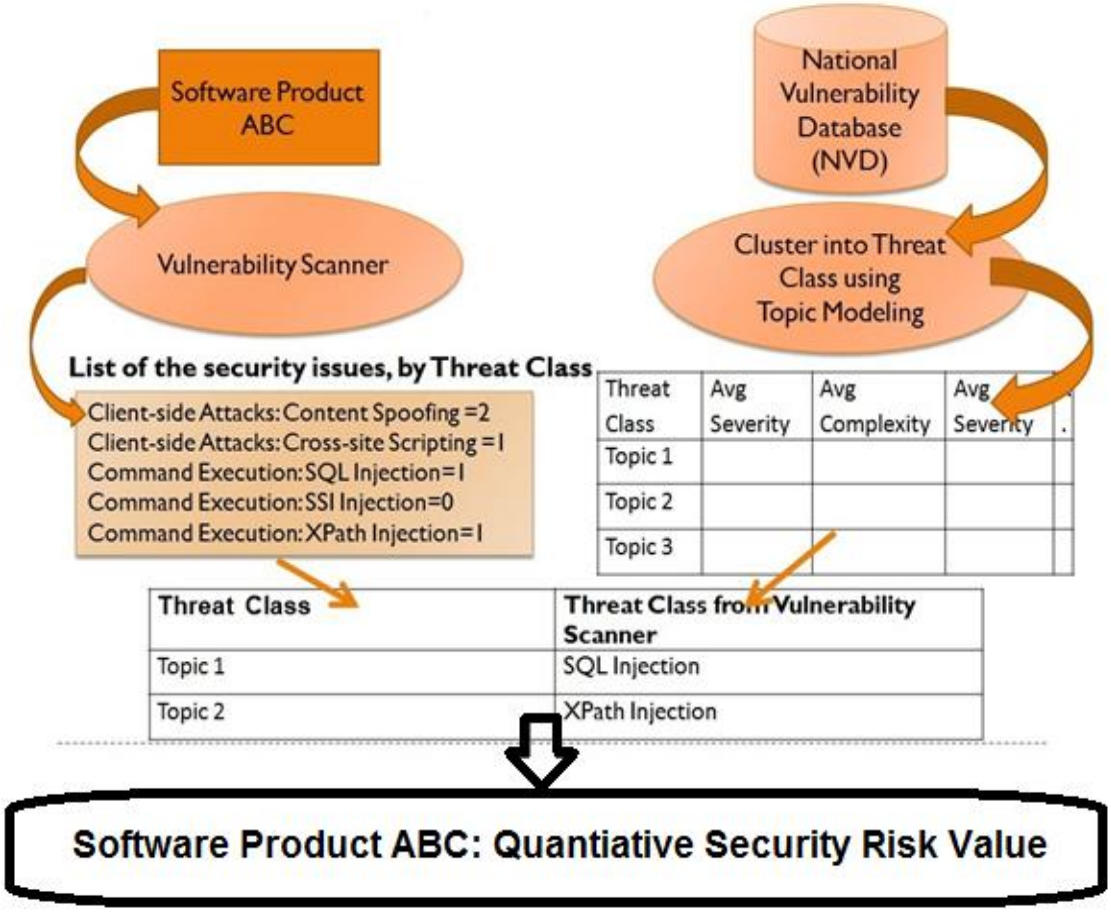


Figure 4-1. Conceptual Model

4.1.1 Hypothesis

The conceptual framework is based on following hypotheses:

Hypothesis 1: Historic vulnerability data can be used to discover software security risk groups through mining and clustering textual data

Hypothesis 2: Vulnerability categories from the scanner can be mapped to the discovered software security risk groups

Hypothesis 3: Risk groups found through clustering historic data provide good representation of the security risk score

4.2 Comparative Analysis of Related Research

4.2.1 Text Mining

Hypotheses 1 and 2 are based on prior work in vulnerability dataset clustering and classification as shown in Table 4-1 below:

Table 4-1. Prior work in Vulnerability Database Classification

Prior Work	Method used
(Neuhaus & Zimmermann, 2010)	Topic Modeling Algorithm
(Venter et al., 2008)	Self-Organizing Maps (SOM)
(Zhongqiang Chen et al., 2010)	Support Vector Machine (SVM)

Topic modeling algorithm: Neuhaus et al. performed security trend analysis using NVD data. Their research used the concept of 'topic model' to cluster the vulnerabilities into groups (Neuhaus & Zimmermann, 2010) that could be used to measure weighted percentage count change by year. After the groups were formed, they were annotated manually, by analyzing the terms in each group. The authors found that topics that were formed were similar to Common Weakness Enumeration (CWE) classification done by MITRE, thus indicating the viability of using topic models to group vulnerabilities.

Self-Organizing Maps (SOM): Venter et al. proposed self organizing maps(SOM) to standardize vulnerabilities, using historical data from the national vulnerability database (Venter et al., 2008). The 'summary text' field from NVD was converted into numeric vectors and fed into SOM algorithm for clustering. A prototype was shown to demonstrate the use of the algorithm.

Support Vector Machine (SVM): Chen et al. categorized the CVEs using the Support Vector Machine (SVM) algorithm, and used the summary field of NVD for text-mining (Zhongqiang Chen et al., 2010). The purpose of this research was to build CVE classifiers for automatic classification and generalization of CVE data. The taxonomic features used for classification were: vulnerability causes, impact, exploit location (local/network), and severity.

4.2.2 Security Metrics and Measurements

Five models were chosen from the previous research studies to design the calculations for our conceptual framework. Table 4-2 provides a brief summary of prior research on which we have based our calculations.

Table 4-2. Comparative Analysis of Related Models

	(Mkpong-Ruffin et al., 2007)	(Wang, Wang, et al., 2009)	(Houmb et al., 2010)	(Chenmeng Sui, 2012)	(Mendes et al., 2011)
Model Name	SSRAM (Software Security Risk Assessment Model	Security Metrics for Software System	Risk Level using CVSS estimates of Frequency and Impact	SSAS (Software Security Assessment System)	VEXA : Software Vulnerability Extractor and Analyzer
Data Source	NVD	NVD	NVD	NVD	OSVDB and NVD

	(Mkpong-Ruffin et al., 2007)	(Wang, Wang, et al., 2009)	(Houmb et al., 2010)	(Chenmeng Sui, 2012)	(Mendes et al., 2011)
Technique	K means Clustering	Custom Calculation based on CVSS BASE SCORE and Number of Months(Frequency)	Bayesian Belief Network (BBN) topology	Custom Calculation based on CVSS and VRSS	Sum the impact and exploitability score. Apply weights based on the relevance of components
Application	In Design Phase	Component Score:IE6, Mozilla etc.	Operation	Operation	Comparing Components(server, OS, database etc.) for web serving systems
Evaluation	Case study	Experiment	Case study	Experiment	Case Study
Quantitative Metrics	Predicted Loss Expectancy Predicted Impact Score Predicted Frequency of Occurrence	Base Impact, And Frequency	Frequency and impact score	Base Impact, Exploitability	Exploitability and Impact level from CVSS
Tools used	TAMT, SQL Server Integration services, MS Analysis studio	None	Hugins, Nessus,	Nessus	None

	(Mkpong-Ruffin et al., 2007)	(Wang, Wang, et al., 2009)	(Houmb et al., 2010)	(Chenmeng Sui, 2012)	(Mendes et al., 2011)
Artifact Produced	Model	Equations	Method, Equations,	Methods, Equations, Tool	VEXA tool

SSRAM (Software Security Risk Assessment Model) :This model was proposed in a doctoral thesis by Mkpong-Ruffin et al.(Mkpong-Ruffin et al., 2007). The model was designed using NVD data and different clustering algorithms, such as K-means and Expected Maximization algorithm. The applicability of the model was demonstrated by generating a list of potential vulnerabilities in an example 'voting' system using Microsoft's Threat Analysis and Modeling Tool (TAMT). These vulnerabilities were then evaluated against the model, to calculate the 'loss expectancy" value using the equation below.

Predicted Loss Expectancy (PLE) = Predicted Impact Score (PIS) X Predicted Frequency of Occurrence (PFO)

$$Predicted\ Impact\ Score = \bar{X}_{CVSS\ Score\ per\ period}\ \mathrm{X}\ (1 + \mu_{\text{growth rate per period}})$$

Equation 4-1. Predicted Loss Expectancy (PLE) (Mkpong-Ruffin et al., 2007)

Where,

Period indicates the evaluation time given,

x = the weighted mean, with the weight being the number of incidences reported

μ = the arithmetic mean

"Quantifying security risk level from CVSS estimates of frequency and

impact"(Houmb et al., 2010): Houmb et al. developed a quantitative risk level calculation using metrics from the CVSS found within the NVD dataset for each vulnerability. The calculation was done by feeding the attributes from NVD CVSS metrics to a Bayesian Belief Network (BBN) topology to get the final risk level score. The potential vulnerabilities and threats, for a company's network, were identified by using a vulnerability scanner called 'Nessus.' The frequency and impact values of those vulnerabilities were obtained using the CVSS metrics. These were fed into the model to obtain the risk level score. The Bayesian Belief Network Topology design by Houmb for risk level calculation is shown in Figure 4-2 below.

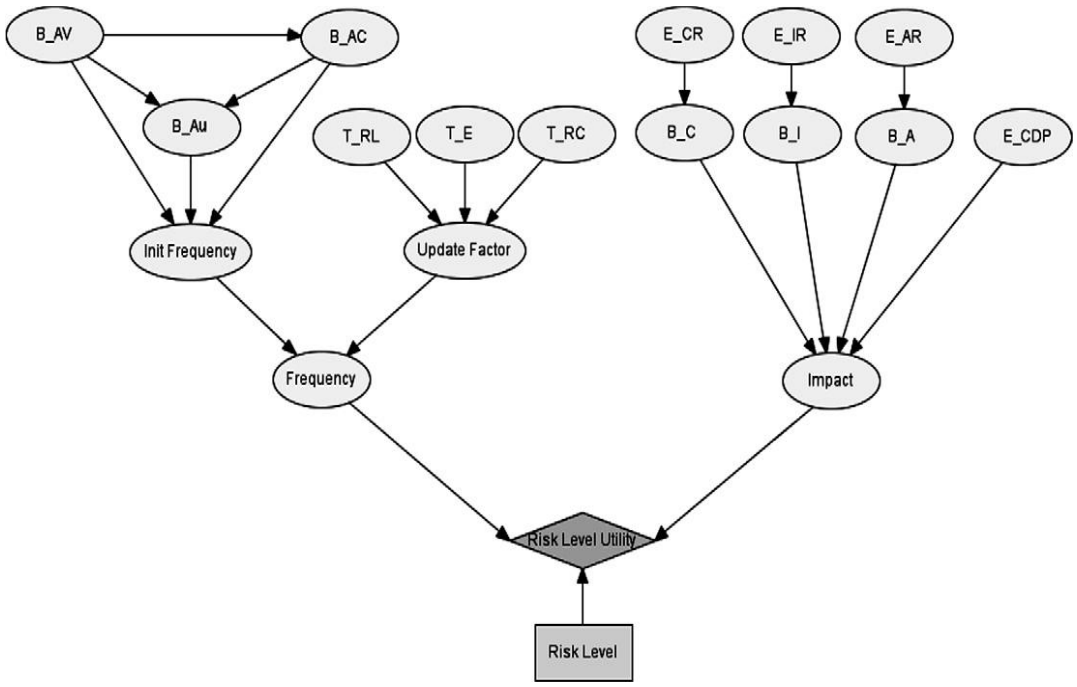


Figure 4-2. Quantitative Security Risk Level Estimation(Houmb et al., 2010)

SSAS (Software Security Assessment System) (Chenmeng Sui, 2012): Sui et al. established a procedure to calculate the security risk score of a software system. In this

procedure, the lists of vulnerabilities were first found by dynamic analysis using Nessus. Next a 'Base impact score' (BI) and the 'Exploitability Score (EX)' for the vulnerability area was calculated using custom equations.

Final impact score (IS) = BI x EX , where

EX= AV × AC × AU and BI= Base Impact = $I_x \times W_I + A_y \times W_A + C_x \times W_C$ Value for Integrity(I), Availability(A) and Confidentiality(C) were custom-defined and weighted(W) based on security requirements. Then, the final Impact Sore and Security Risk Score were calculated as shown in

Equation 4-2:

$$SRS = \sum_{j=1}^{3} \sum_{i=1}^{n_j} \left(\frac{IS_i}{n_j} * W_j \right)$$

Equation 4-2. SSAS (Chenmeng Sui, 2012)

Where SRS= Security Risk Score, j=risk level (low=1, medium=2,High=3), n=number of vulnerability found by scanner at a specific level, IS=Impact Score

Security Metrics for Software System (Wang, Wang, et al., 2009): Wang et al. proposed a procedure for calculating security metrics for a software system using an aggregation of vulnerabilities, their impact scores, and their frequencies in the NVD dataset. The steps in this procedure were as follows: For a given component (IE, Firefox browser etc), find the weakness and their counts (m). Then, for each weakness (CWE), aggregate the CVSS base score, and find the frequency of occurrence over the span of

time it was reported. The count and the frequency of the occurrence value was used to derive the average base score (W_n) and the probability of occurrence (P_n). The final security metric score SM(s) was an aggregation of the average base score and the probability of occurrences for all weaknesses.

$$SM(s) = \sum_{n=i}^{m} P_n X W_n$$

Equation 4-3. Security Metrics for Software System (Wang, Wang, et al., 2009)

VEXA(Software Vulnerability Extractor and Analyzer) (Mendes et al., 2011): Mendes et al. proposed a benchmarking methodology for evaluating the risk levels of the web serving systems based on static analysis of the components and assessment using attributes from OSVBD data. They extracted the impact and the exploitability score for the vulnerabilities found in the components and summed them to get the total security risk levels of the component. The total security risk level of the system was then calculated by deriving the weighted sum of the component security risk level.

$$CR = \sum_{i=1}^{n} (VRi = Ii * Ei)$$

$$FSR = \sum_{i=1}^{n} (CRi * Fi)$$

I : Impact Score, E : Exploitability Score , i=number of vulnerability, VR: vulnerability risk

"FSR: total (final) security risk of the entire (web serving) system" (Mendes et al., 2011)

Equation 4-4. VEXA (Mendes et al., 2011)

4.3 Research Framework Summary

The research framework has been based on text-mining procedures from several of the prior work in quantitative security risk assessment. We plan to adopt proven equations and measures from prior research to help build our model. Adopting a system level view, we establish the levels and dimensions for security risk level calculation for our research. Based on the goals of the research, we plan to implement the research framework using on the Design Science Research methodology, based on the guidelines provided by Hevner et al. (Hevner et al., 2004) and the research process proposed by Peffers et al. (K. Peffers et al., 2008). This methodology and its application have been described in the next chapter.

The main steps of this research are:

Security risk groups mining from historic data

Impact Score, Probability of Occurrence, and Probability of Impact Risk calculation

Validation and Verification of the Quantitative Model

Mapping the Cluster to Vulnerability Scanner categories

Experimentation case study demonstration

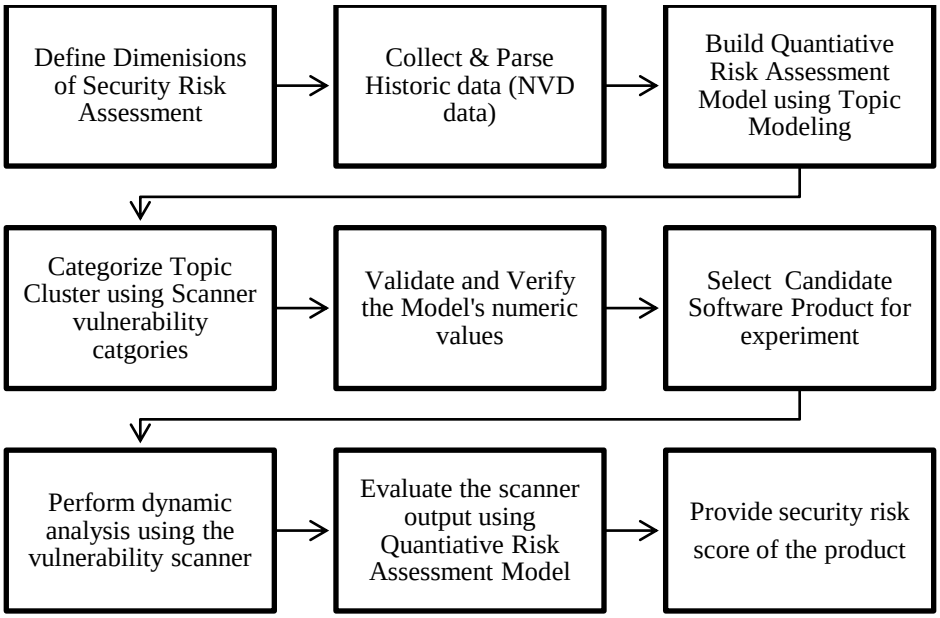


Figure 4-3. Research Framework

5 Research Methodology

This chapter describes the research methodology proposed to answer the research questions. The research methodology has been selected based on the goal of proposing an artifact, to solve the problem of quantitative security risk assessment of software product before its adoption. The rational for the selected research methodology is first presented, and then, the subsequent subsections describe the procedure, data collection technique, measurement instrument, experiment, and overall considerations of validity and evaluation.

5.1 Design Science Research Methodology (DSRM)

"Our research is carried out using guidelines provided by Hevner et al. (Hevner et al., 2004) and presented using the steps proposed by Peffers et al. (K. Peffers et al., 2008) for this methodology. Design Science (DS) research methodology is common in the field of information systems. Hevner et al. categorized the artifacts created through the DS methodology into constructs, models, methods, and instantiation"(Ruma Das, 2012). Table 5-1 provides examples of each of these artifact categories.

Table 5-1. Artifacts for Design Science Research (Hevner et al., 2004)

Artifacts	Example
Constructs	vocabulary and symbols
Models	abstractions and representations
Methods	algorithms and practices
Instantiations	implemented and prototype systems

The methodology for our research based on DSRM has been shown in

Figure 5-1 below.

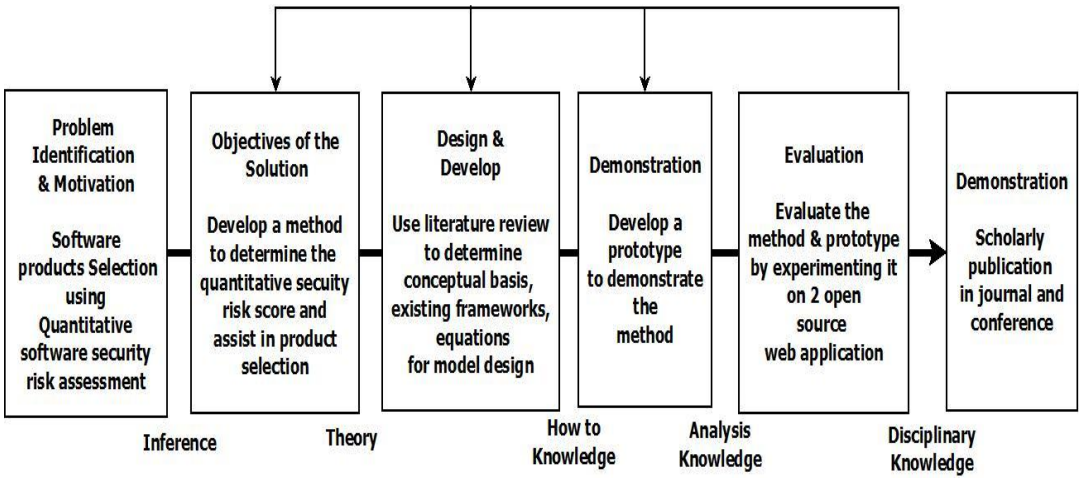


Figure 5-1. DSRM adapted from (Peffers et al., 2008)

5.2 Rationale for DSRM

Design science research is very similar to action research since both, attempt to identify and solve a problem. However, action research is a change-oriented approach that incorporates close collaboration between the researcher and practitioners (Juhani Iivari 2009). Design science research does not assume such collaboration. In fact, here, the researcher identifies the issue, develops the theory, and finally, produces an artifact using the theory to solve the problem. Unlike DSRM, the initiators of the study in action research are often the practitioners. Our research is based on solving a problem by developing a procedure. The industry practitioners are not involved and therefore, DSRM is an appropriate methodology to follow in research such as ours.

As pointed out by Peffers et al. "Design Science is of importance in a discipline oriented to the creation of successful artifacts"(K. E. N. Peffers et al., 2007). As mentioned by Hevner et al, the primary purpose of artifacts created using design science is to "address unsolved problems" and the value is based on the "utility provided in solving those problems"(Hevner et al., 2004). As pointed out in our publication, our

rationale for using DSRM can be described as follows: “Since the goal of our research is to create a method to select between multiple software products based on quantitative software security risk assessment, DSRM is an ideal methodology to follow. Finally, design science is a problem-solving paradigm and thus, supports our goal of solving the issue of evaluating software product’s security risk, quantitatively, during the early stages, without requiring a large amount of resources, and making a comparative analysis for selection.”(Ruma Das, 2012) We present our research below, using the language of DRSM.

5.3 Problem Identification and Rationale

This section describes the problem that we attempt to solve and the rationale for undertaking the study. In our paper we have described this as follows: “Recently several quantitative security risk assessment models have been proposed by many scholars, many of which use vulnerability data from public databases, such as NVD, OSVBD, etc. Even though vulnerability data has been used for predicting exploits, managing patching, and evaluating network security risks, it has rarely been applied to software selection. Since vulnerability is a type of defect that reduces the reliability of the system, filling this gap would help in the evaluation and selection of custom or open source software systems”(Ruma Das, 2012).

In sum the gap that we find is that “existing quantitative software security risk models fall short of fulfilling above intention, since they evaluate individual COTS product and topology made of COTS, like servers, databases, etc”(Ruma Das, 2012). OSS selection frameworks such as OpenBRR and QUALOSS, have addressed few quantitative measures, but they require the software product vulnerabilities to be reported to NVD for

the metrics to be calculated, and thus, do not address latent vulnerabilities that might manifest themselves in operational settings. Thus the research gap is that, "we have not found any instrument providing objective measurements to help with the assessment of candidate software products, such as web-based applications, not developed in-house"(Ruma Das, 2012). For products, not developed in-house, we can use the vulnerability scanner to identify potential security risk. In our review of prior works, we have not seen a procedure to utilize this and evaluate the impact and probability of discovered security risks in operational settings. In many situations, this is a manual process, and thus the values are based on subjective interpretation. However, in today's "data-rich" world, information about discovered vulnerabilities and their characteristics can be found in data sources, such as the NVD, OSVD, and Secunia etc. The problem is that the information content is mostly in text format. Therefore, the challenge is to mine the security risk groups from these textual data sources first, and then, develop a model that can be used to score the impact and probability values of scanner vulnerability categories.

"The rationale for this research is to provide an artifact that can assist practitioners in making informed decisions based on empirical data and objective value"(Ruma Das, 2012). The primary research questions (RQ) evaluated was:

"RQ1: What are the essential dimensions for evaluating a candidate software product before selection?

RQ2: How do we analyze the security risk level of a potential software product?

RQ3: Can we use a quantitative security risk assessment model to compare

and select between custom software products?"(Ruma Das, 2012)

5.4 Objective of the Solution

"The objective of the solution is to design a quantitative security risk assessment model, and demonstrate its use as a measuring instrument that can provide quantitative value for objective decision-making during software selection" (Ruma Das, 2012).

5.5 Model Design and Development

5.5.1 Proposed Model

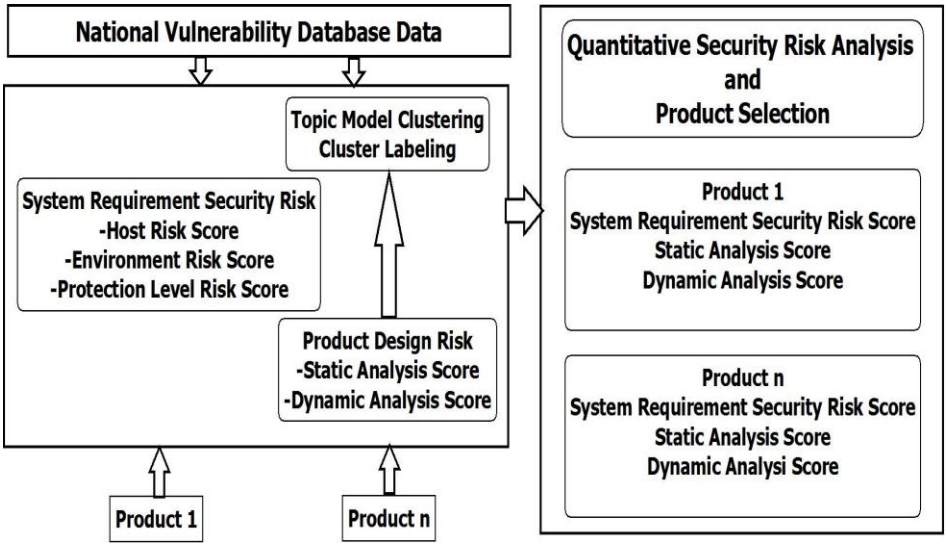


Figure 5-2. Model for OSS selection using Quantitative Security Risk Analysis(Ruma Das, 2012)

Figure 5-2 shows the design of the model for OSS selection using quantitative security risk analysis.

The procedure for building this model consists of the following steps:

Determine dimensions for evaluation(Ruma Das, 2012)

Collect NVD data and cluster using topic modeling methods (Ruma Das, 2012)

Obtain the categories of vulnerability that the vulnerability scanner tool can report(Ruma Das, 2012)

Label the topics found in step 2, using the categories from the vulnerability scanner(Ruma Das, 2012).

Calculate and store the topic's average impact score, attack vector probability, and CIA metric probabilities. These numeric values with topic model developed in step 3 and 4, results in the quantitative security risk assessment model for our study(Ruma Das, 2012).

Inspect and find system requirements for the candidate software product. Identify the operating system, database servers, web servers and other components needed, their brand and version(Ruma Das, 2012).

Determine the security score of the system requirement components (Ruma Das, 2012).

Install the software product and perform dynamic analysis using web application scanner. Use the quantitative security risk assessment model to determine impact score and frequency(Ruma Das, 2012).

Repeat this for all candidate products and compare the results(Ruma Das, 2012).

"To design the procedure, we investigated the sources of security risk of a software product. A software product faces a security risk due to the code base, the COTS product (servers, OS, etc.) it is hosted on, and the environment it is installed in. Thus, we propose two primary dimensions for evaluating the security risk of the candidate software product: System Requirement Risk and Product Design Risk, as shown in Figure 5-3."

(Ruma Das, 2012)

5.5.2 Dimensions of Risk

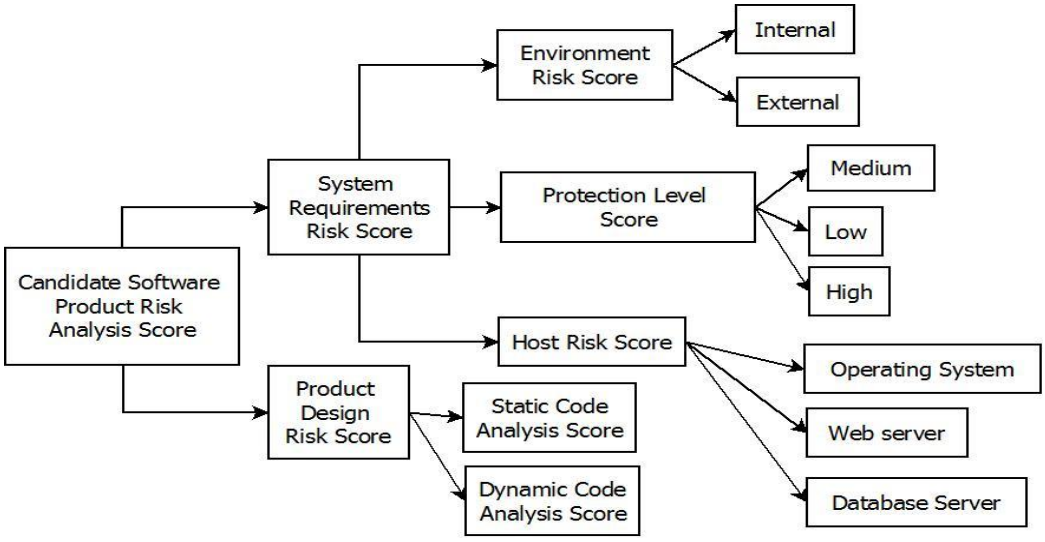


Figure 5-3. Risk assessment dimensions for the candidate product(Ruma Das, 2012)

5.5.2.1 System Requirement Risk

A software product comes with system requirements, which specify the operating system, database server, and web server on which it can operate (Ruma Das, 2012). These are sources of risk and can be further subdivided into:

Host Risk: This is the first sub-dimension and is related to the COTS product on which it is hosted. These are operating systems (OS), web servers, and database servers. Each of them is a source of security risk based on the number of vulnerabilities that reside within (Ruma Das, 2012).

Environment Risk: The next risk comes from this sub-dimension due to the environment in which the software product might be installed. A public-accessible server is more risky than one installed within a firewall and closed to public access(Ruma Das,

2012).

Protection Level Risk: The final risk sub-dimension is a subjective score that the practitioners can assign based on their knowledge about the protection controls that exists in terms of people, software, and process on the host(Ruma Das, 2012).

5.5.2.2 Product Design Risk

Another dimension of risk is attributed to the flaws in the software caused by programming errors, such as integer overflow or buffer overflow. These can be detected using static and dynamic analysis to identify a list of potential vulnerabilities (Ruma Das, 2012).

The next section describes the quantitative security risk assessment model development process.

5.5.3 Quantitative Security Risk Model Development

The first major step in this research is the development of the quantitative risk assessment model. Literature review indicates several methods used in prior research in the areas of vulnerability classification and clustering, such as K-mean, SOM, SVM etc. All of these methods have had successes and limitations. We decided to explore and evaluate ‘topic modeling’ algorithm for security risk-mining because most of the vulnerability reports were in the text format. Topic modeling algorithm creates topic groups based on the hidden themes within the text. These hidden themes from vulnerability reports indicate security risk groups. In addition, the output of the application scanners, that scan software products for vulnerabilities, have textual description of the vulnerability categories that help us map those categories to the hidden

themes derived through topic modeling. Models once trained can be repeatedly used to categorize other text from the same domain. Finally, topic modeling algorithm does not need specific fields to be selected for clustering. Any text field of the vulnerability report or an entire document of the report could be fed into the algorithm. Therefore, we were motivated to evaluate topic modeling for creation of the quantitative risk assessment model.

The quantitative risk assessment model development process consists of three primary tasks: data collection, data pre-processing, and topic model building.

5.5.3.1 Data Collection

The data collection process involved downloading the vulnerability data available in XML file format, from the National Vulnerability Database (NVD, 2012) website and loading to a Microsoft SQL server database. The data is available as one XML file for each year. The file is named as nvdcve-2.0-[year].xml, where the year indicates a number from 2002 to 2012. A Perl script was written to extract the essential attributes from the xml file and loaded the data into a table in the Microsoft SQL Server database (MS SQL server). The vulnerability data for all the years was loaded into one table called 'NVDcve', and related product information was loaded into another table. In subsequent steps, this database was also used to store and analyze data resulting from executing topic modeling algorithms for creation, training, and inference of the topic models.

5.5.3.2 Topic Modeling Tool Selection

There are several open source tool-boxes available for processing text using different types of topic modeling algorithm. We selected the Latent Dirichlet Algorithm (LDA),

for our purpose, due to its wide success noted in prior research, which helped us adopt and cross reference the results. Two open source tool-boxes were selected initially: Stanford Topic Modeling tool-box(Stanford, 2012) and Mallet (McCallum, 2002). For us, using the Stanford Topic Modeling tool-box required less pre-processing. The data from Microsoft SQL server could easily be exported into comma-separated values (CSV) file for feeding into the LDA algorithm implementation of the Stanford Topic Modeling tool-box. The output from the Stanford Topic Modeling tool-box was in CSV format and could be easily loaded into the MS SQL server database for further analysis.

5.5.3.3 Data Pre-processing

In case of pre-processing for the Stanford Topic Modeling toolbox, we extracted 2 fields, 'CVE_ID' and 'Summary text', from the NVDcve table, to create one *comma-separated values* (*CSV*) file for all the rows. This extracted NVD CSV file was then fed into the Stanford Topic Modeling toolbox program to create and train the topic model. Similarly, for analyzing vulnerability categories from the scanner, we created one CSV file with 2 columns the category id and their description, with one row per category as shown in Appendix A. Before creating the CSV file, we cleaned the 'summary text' data by removing, quotations, punctuations and numeric characters by replacing them with single empty space.

5.5.3.4 Model Creation

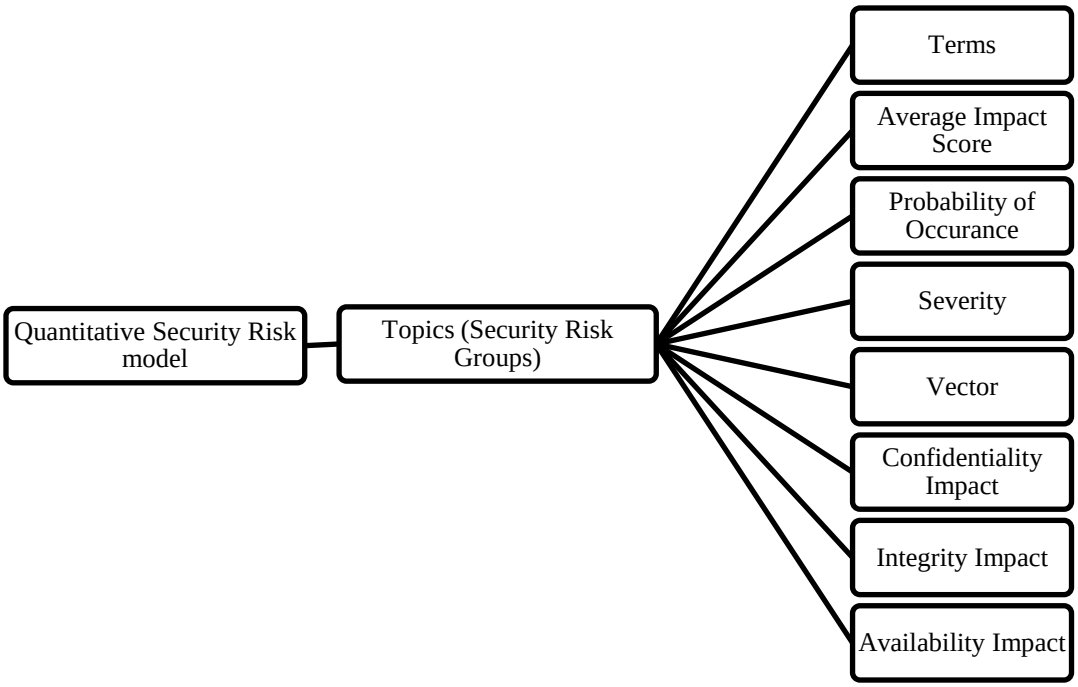


Figure 5-4. Quantitative Model Components

Our quantitative model consisted of 3 major components, 'topics' that represented software security risk groups, their average impact score, and their probability as shown in Figure 5-4. Related to the 'topics' were the secondary attributes that indicated the terms within each topic and their impact probability distribution. These 'topic terms' provided information about the types of security risks, which the topic represented and assisted in validating the topic labeling process. In addition, for each topic, we calculated the probability of the different levels(high, low , medium) of attributes, such as, severity, vector, complexity, confidentiality, integrity, and availability. The purpose of these metrics was to help determine, for a given security risk indicated by the topic, the likelihood of the specific level of risk impact. The conceptual model of this process is shown in Figure 5-5.

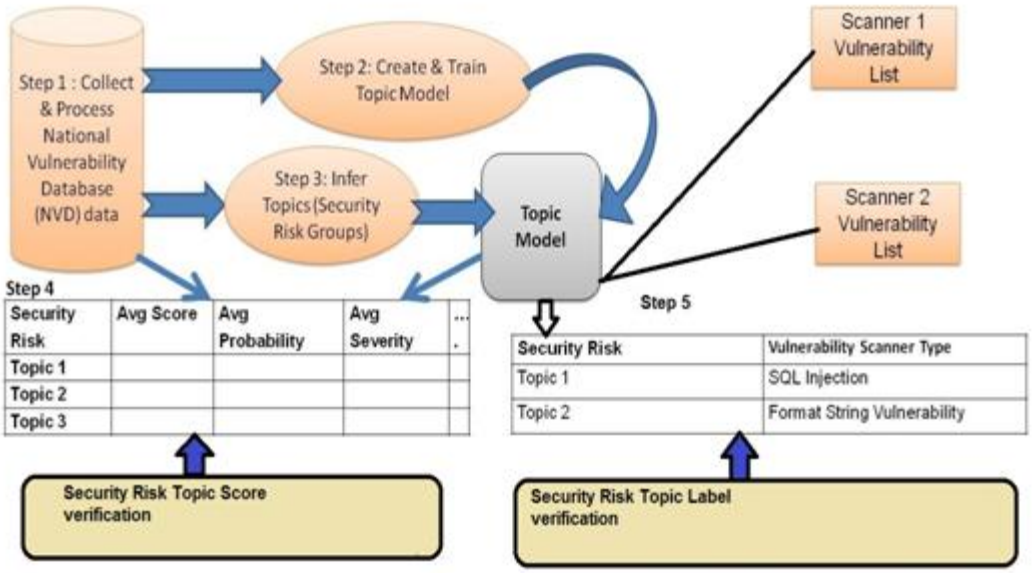


Figure 5-5. Quantitative Security Risk Assessment Model Creation Process

The preparation of the model consisted of the following steps:

Selecting the number of topics for the 'Topic Model'

Training the model

Inferring the topics using NVD data

Verifying the Topic cluster cohesion

Evaluating the model score values

Labeling the Topics

5.5.3.4.1 Topic Modeling

The topic modeling algorithm used for our experiment was "Latent Dirichlet Allocation (LDA)". The extracted NVD data, imported as CSV file was processed using the Stanford Topic Modeling Toolbox (Stanford, 2012). The toolbox enabled us to apply a simple English tokenizer and remove English stop words like 'the,' 'of,' etc. We set the

parameters such that summary text with less than 5 terms was removed and non-words characters were ignored.

In terms of estimating the 'number of topics' for the LDA algorithm, we followed the following process: "We ran the LDA model using the CVB0 algorithm, looping over the number of topics ranging from 10 to 100 and calculating the perplexity score. The model was trained using 80% of the data and tested using 20% of the data. Even though optimum number of topics from the graph in Figure 5-6 was initially determined to be 60, we finally chose 40, since it gave the least amount of overlaps and generic topics" (Ruma Das, 2012). Figure 5-6 shows the graph demonstrating the perplexity score versus the number of topics.

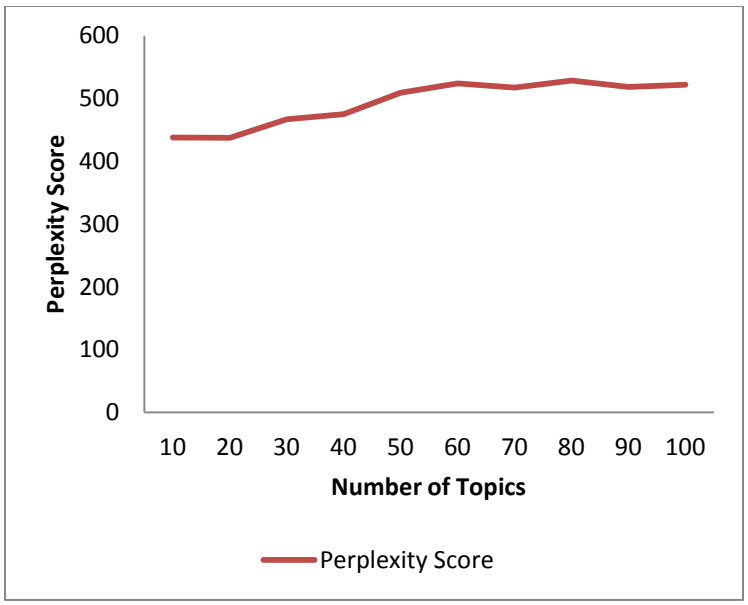


Figure 5-6. Topic Selection Evaluation(Ruma Das, 2012)

The output of topic modeling training and inferring process resulted in multiple files of which we primarily used two. The first file contained the topic groups and associated terms as shown in Table 5-2. The second file contained indicators showing association

between the CVE entry and the topic distribution as shown in Figure 5-7. nvddata-document-topic-distributions outputFigure 5-7.

Table 5-2. Topics and Top 10 Related Terms Output From LDA toolbox

Topic	Terms
Topic 24	string, function, arbitrary, execute, allows, attackers, format, remote, commands, shell, vulnerability, code, metacharacters, properly, handled, command, variable, argument, program, functions, earlier, demonstrated, specifiers, using, cgi, context-dependent, variables, query, php, certain, strings, arguments, allow, line, option, expression, used, perl, regular, possibly
Topic 26	server, web, remote, attackers, allows, http, request, url, apache, source, redirect, determine, iis, manager, requests, sites, page, application, open, uri, default, proxy, network, node, tomcat, jsp,mail, interfacepages, administration, sap, scripts, coldfusion, existence, error, contains
Topic 23	script, scripting, web, xss, cross-site, html, inject, arbitrary, attackers, remote, vulnerability, allows, parameter, earlier, index.php, search, page, action, parameters, cms, query, search.php, uri, error, title, _info, msg, path, portal, different, insert, login, theme, keyword

	CVE_ID	Topic 00	Topic 01	Topic 02	Topic 03	Topic 04	Topic 05	Topic 0
1	CVE-2012-0001	3.09E-05	6.63E-06	7.88E-04	4.27E-04	2.38E-04	2.58E-05	0.0583
2	CVE-2012-0002	2.33E-06	5.90E-06	2.31E-04	2.01E-05	7.22E-04	9.44E-05	0.0018
3	CVE-2012-0003	3.52E-06	5.01E-04	6.27E-06	3.03E-04	1.03E-04	8.32E-06	9.43E-0
4	CVE-2012-0004	0.001399518	3.25E-04	2.04E-04	1.26E-04	2.38E-06	1.10E-04	1.82E-0

Figure 5-7. nvddata-document-topic-distributions output

The database server used for data storage and manipulation was “Microsoft SQL Server Express 2008 R2”. The database was used to store the downloaded NVD data, and also the output from topic model algorithm processing using the LDA toolbox. This topic modeling algorithm output data was parsed and loaded into the database, resulting in a table having the attributes CVE ID, the Topic Name, and the associated probability distributions. Since the topic modeling algorithm assigns multiple topics per CVE, we

assumed that for each CVE, the most relevant topic was the topic with the highest probability distribution. The final result was a table where each row indicated the vulnerability ID (CVE ID), its most relevant Topic Name, Score, and maximum probability distribution as shown in Table 5-3.

Table 5-3. Vulnerability and Topic with maximum probability distribution

cve_id	Topic	Score	Max Probability
CVE-2008-2012	Topic 02	7.5	0.83
CVE-2012-0005	Topic 28	6.9	0.82
CVE-2012-0011	Topic 08	9.3	0.87
CVE-2012-0018	Topic 08	9.3	0.93
CVE-2012-0019	Topic 08	9.3	0.96
CVE-2012-0020	Topic 08	9.3	0.96
CVE-2012-0056	Topic 19	6.9	0.83
CVE-2012-0069	Topic 02	7.5	0.98
CVE-2012-0072	Topic 27	5	0.99

5.5.3.4.2 Topic Cluster Labeling

Topic labeling needed human intervention as noted in prior research (Neuhaus & Zimmermann, 2010; Venter et al., 2008). In a prior work in the area of Topic labeling, Hindle et al. generated the word-list with category names from ISO9126 and ontology developed by prior research pertaining to non-functional requirements. Then, this word-list was used to label the topics generated from the description text of the software maintenance activities. The purpose was to assign a label to a topic if any of the words from the word-list categories matched the topic terms (Hindle et al., 2011). We attempted to label our topics using categories from the vulnerability scanner. Our expectation was that the anchor words, such as ‘buffer overflow’ or ‘SQL injection’ would lead to matching the topic to the right vulnerability categories. We used the topic model trained with NVD data and provided the vulnerability scanner categories as input. The output of

this process was expected to be a mapping between the vulnerability scanner categories and topics. Due to a large number of frequently occurring common words and short scanner categories text, we found that scanner categories got associated with the multiple topics as shown in Table 5-4. Also, as noted by Neuhaus et al. the topic groups needed to be merged to form unique categories.

Table 5-4. Scanner Category Topic Mapping using AppScan categories

Topic	Scanner Category	Top Terms
Topic 00	Abuse of Functionality	integ,function,valu,craft,overflow,imag,code,trigger,context, depend,larg,memori,heap,length,error,size,earlier,relat,data, arrai,lead,multipl,neg,buffer,librari,font,possibl,product,read ,decod,pdf,png,alloc,number,signed,gif,jpeg,contain,certain, field
Topic 01	SSI Injection	cgi,command,build,shell,metacharact,program,vmware,host, earlier,server,script,workstat,virtual,argument,bugzilla,guest ,filenam,version,player,tool,process,setup,configur,esx,hotfi x,net,cmd,function,ecom,possibl,bin,packag,solut,exec,user, includ,samba,awstat,read,esxi
Topic 02	Format String Attack	string,function,format,charact,code,variabl,specifi,earlier,de monstr,queri,contain,possibl,certain,encod,sequenc,mysql,ar gument,input,log,express,tabl,command,filter,escap,relat,file nam,data,version,insert,regular,statement,utf,perl,termin,eva l,quot,modifi,unicod,inject,doubl
Topic 02	XPath Injection	string,function,format,charact,code,variabl,specifi,earlier,de monstr,queri,contain,possibl,certain,encod,sequenc,mysql,ar gument,input,log,express,tabl,command,filter,escap,relat,file nam,data,version,insert,regular,statement,utf,perl,termin,eva l,quot,modifi,unicod,inject,doubl
Topic 04	Information Leakage	inform,obtain,sensit,password,store,access,web,control,root, databas,request,direct,insuffici,read,cleartext,download,encr ypt,configur,data,credenti,plaintext,mdb,potenti,usernam,use r,log,document,contain,panel,earlier,weak,hash,txt,includ,sn if,kei,decrypt,phpinfo,backup,plesk
Topic 05	SQL Injection	sql,inject,command,index,earlier,action,multipl,com,joomla, compon,_id,magic,disabl,_quotes_gpc,usernam,cat,new,asp, view,page,categori,aka,catid,cid,beta,portal,search,mambo,b oard,articl,possibl,book,link,list,classifi,login,blog,pid,pro,m odul
Topic	Denial of	servic,caus,denial,crash,possibl,applic,memori,corrupt,long,

Topic	Scanner Category	Top Terms
06	Service	code,craft,hang,daemon,free,trigger,unspecifi,version,earlier,bound,engin,multipl,argument,doubl,oper,string,shutdown,empti,platform,devic,nicknam,ircd,audio,persist,unknown,openttd,alpha,seri,instabl,thread,counter
Topic 08	Session Fixation	password,authent,upload,administr,access,cooki,login,usernam,session,admin,bypass,account,set,user,gain,direct,earlier,default,request,extens,code,unrestrict,requir,doe,modifi,chang,physic,privileg,valu,field,screen,lock,hijack,auth,possibl,log,configur,featur,proxim,new
Topic 12	Content Spoofing	user,content,domain,type,doe,spoof,open,url,assist,origin,displai,site,bypass,frame,download,address,target,window,link,dialog,polici,secur,opera,trust,conduct,bar,check,prevent,trick,event,set,featur,verifi,malici,properli,possibl,box,phish,read,save
Topic 13	Cross-site Scripting	script,site,web,html,cross,xss,inject,multipl,earlier,index,search,page,field,action,possibl,login,messag,guestbook,version,email,queri,uri,titl,beta,view,comment,text,portal,url,msg,theme,insert,jsp,usernam,link,keyword,new,result,cfm,templat
Topic 17	Buffer Overflow	overflow,buffer,code,long,base,stack,heap,multipl,function,earlier,craft,user,possibl,assist,string,argument,command,version,filenam,dll,field,playlist,mai,line,trigger,contain,cpp,player,request,pars,malform,pro,hostnam,librari,entri,winamp,build,tftp,beta,header
Topic 20	Predictable Resource Location	servic,caus,denial,port,packet,consumpt,connect,larg,tcp,number,memori,cpu,send,earlier,udp,network,leak,resourc,data,server,exhaust,request,mani,fragment,address,flood,malform,hang,multipl,traffic,firewal,icmp,ipv,protocol,version,sourc,rout,process,certain,close
Topic 23	Brute Force	doe,bypass,restrict,access,properli,make,intend,easier,check,kei,secur,protect,conduct,forc,function,certain,configur,authent,valid,gener,valu,disabl,implement,mechan,brute,time,set,perform,session,requir,leverag,modifi,provid,enabl,guess,return,acl,random,polici,signatur
Topic 23	Insufficient Authentication	doe,bypass,restrict,access,properli,make,intend,easier,check,kei,secur,protect,conduct,forc,function,certain,configur,authent,valid,gener,valu,disabl,implement,mechan,brute,time,set,perform,session,requir,leverag,modifi,provid,enabl,guess,return,acl,random,polici,signatur
Topic 23	Insufficient Authorization	doe,bypass,restrict,access,properli,make,intend,easier,check,kei,secur,protect,conduct,forc,function,certain,configur,authent,valid,gener,valu,disabl,implement,mechan,brute,time,set,perform,session,requir,leverag,modifi,provid,enabl,guess,return,acl,random,polici,signatur

Topic	Scanner Category	Top Terms
Topic 23	Insufficient Session Expiration	doe,bypass,restrict,access,properli,make,intend,easier,check,kei,secur,protect,conduct,forc,function,certain,configur,authent,valid,gener,valu,disabl,implement,mechan,brute,time,set,perform,session,requir,leverag,modifi,provid,enabl,guess,return,acl,random,polici,signatur
Topic 37	Credential/Session Prediction	user,local,creat,overwrit,permiss,symlink,temporari,log,modifi,instal,directori,read,packag,world,earlier,insecur,linux,configur,write,script,delet,tmp,possibl,oper,default,readabl,set,suse,pam,writabl,debian,link,version,util,replac,run,group,home,gnu/linux,weak
Topic 38	Directory Indexing	directori,dot,travers,read,includ,sequenc,local,index,earlier,filenam,list,multipl,pathnam,compon,disabl,possibl,languag,com,absolut,magic,joomla,backslash,_quotes_gpc,lang,write,download,control,page,slash,overwrit,creat,templat,beta,action,dir,view,encod,uri,path,_file
Topic 38	Path Traversal	directori,dot,travers,read,includ,sequenc,local,index,earlier,filenam,list,multipl,pathnam,compon,disabl,possibl,languag,com,absolut,magic,joomla,backslash,_quotes_gpc,lang,write,download,control,page,slash,overwrit,creat,templat,beta,action,dir,view,encod,uri,path,_file

We obtained additional vulnerability categories text from other scanners, and ran them against the topic model, to map them to our topics groups. If the topic was not already present in our topic-label table, we added it to our table. If the topic was present, then, we appended the new label to our previous labels for that topic. Manual analysis was performed to detect whether the topic labels indicated the same vulnerability or were wrongly labeled. Unlabeled topics remained with unique numbers assigned to them and were labeled ‘Unlabeled Topic x’ where x was a number.

5.5.4 Verification and Validation

We evaluated the quantitative risk assessment model using visual as well as statistical methods to meet the following criteria:

Topics in the model are relevant to the domain and indicate specific security risk

Data points in the Topic clusters are cohesive

Numerical values of the model are good representatives of the specific security risk group

5.5.4.1 Topic Cluster Verification

Topic clusters are statistical and probabilistic clustering mechanisms. Therefore, verifying the cluster for validity within a domain is essential. Finding the ground truths for topic model verification is challenging (Ramirez, Brena, Magatti, & Stella, 2012). In the NVD data, some vulnerabilities were categorized using the Common Weakness Enumeration Specification(CWE) types (MITRE). We assumed CWE type categorization of NVD data to be the ground truth for verification. In this process, we attempted to label a topic with a CWE category, if the topic had maximum vulnerabilities of that CWE type as shown in Table 5-5.

Table 5-5. Topic CWE verification

Topic	cwe_id	Name
Topic 17	CWE-119	Buffer Errors
Topic 02	CWE-134	Format String Vulnerability
Topic 21	CWE-16	Configuration
Topic 23	CWE-16	Configuration
Topic 00	CWE-189	Numeric Errors
Topic 11	CWE-20	Input Validation
Topic 34	CWE-200	Information Leak / Disclosure
Topic 38	CWE-22	Path Traversal
Topic 04	CWE-255	Credentials Management
Topic 23	CWE-264	Permissions, Privileges, and Access Control
Topic 08	CWE-287	Authentication Issues
Topic 23	CWE-310	Cryptographic Issues

Topic	cwe_id	Name
Topic 29	CWE-352	Cross-Site Request Forgery (CSRF)
Topic 32	CWE-362	Race Conditions
Topic 20	CWE-399	Resource Management Errors
Topic 37	CWE-59	Link Following
Topic 01	CWE-78	OS Command Injections
Topic 13	CWE-79	Cross-Site Scripting (XSS)
Topic 05	CWE-89	SQL Injection
Topic 27	CWE-94	Code Injection

Most classification algorithms' correctness and effectiveness is verified using the precision and recall measures. Even though in topic modeling, a CVE can be associated with multiple topics, we used the same measures and evaluated the topic clusters. Precision is defined as the number of correctly predicted data to the total amount of data. Recall, which is the measure of effectiveness of the algorithm, is defined as the ratio of the number of correctly predicted data to the actual number of correct data records (Chowdhury & Zulkernine, 2011).

$$\text{Precision} = \frac{True\ Positives}{Trur\ Positives+False\ Positives}$$

$$\text{Recall} = \frac{True\ Positives}{True\ Positives+False\ Negatives}$$

We tested the precision and recall using the CWE types and the comparable topics groups. The results are shown in Table 5-6. We believe for a quantitative model based on historic data, the recall measure is more important, since we want our numeric score to be a good representation of the whole security risk group and the probability of the impact to reflect the real risk posture. Finally, the F-measure, which can be defined as "the harmonic mean of precision and recall" (Chowdhury & Zulkernine, 2011), was calculated. We used the equation given below, with $\beta=1$. Since we need the quantitative

model to return good score, we believed recall must be given more weight than precision to evaluate the cluster and thus, we also calculated the F_2-measure where β=2, gave more weight to recall.

$$\text{F-Measure} = \frac{(1+\beta^2)X\ (\ Precision\ X\ Recall)}{(\beta^2\ X\ Precision)+Recall}$$

Table 5-6. Precision-Recall between Topic Groups and CWE Types

CWE	Topic	Total CWE (X)	Total Mapped Topic Found (Y)	Total Mapped Topic Within CWE (Z)	Precision (Z/Y)	Recall (Z/X)	F-Measure	F 2 - Measure
CWE-189	Topic 00	791	1196	318	26.59%	40.20%	32.01%	33.37 %
CWE-78	Topic 01	59	666	13	1.95%	22.03%	3.59%	4.31%
CWE-134	Topic 02	135	1047	88	8.40%	65.19%	14.89%	17.61 %
CWE-255	Topic 04	274	1235	66	5.34%	24.09%	8.75%	10.02 %
CWE-89	Topic 05	3311	4320	2788	64.54%	84.20%	73.07%	75.06 %
CWE-287	Topic 08	611	1471	248	16.86%	40.59%	23.82%	25.97 %
CWE-20	Topic 11	1836	1352	174	12.87%	9.48%	10.92%	10.59 %
CWE-79	Topic 13	3391	5594	2905	51.93%	85.67%	64.66%	68.00 %
CWE-119	Topic 17	3372	2203	955	43.35%	28.32%	34.26%	32.88 %
CWE-399	Topic 20	1521	1561	185	11.85%	12.16%	12.01%	12.04 %
CWE-16	Topic 21	211	1528	29	1.90%	13.74%	3.34%	3.93%
CWE-16	Topic 23	2085	1528	29	1.90%	1.39%	1.61%	1.56%
CWE-310	Topic 23	343	1528	134	8.77%	39.07%	14.32%	16.40 %

CWE	Topic	Total CWE (X)	Total Mapped Topic Found (Y)	Total Mapped Topic Within CWE (Z)	Precision (Z/Y)	Recall (Z/X)	F-Measure	F 2 - Measure
CWE-264	Topic 23	2085	1528	286	18.72%	13.72%	15.83%	15.36 %
CWE-94	Topic 27	1589	1975	673	34.08%	42.35%	37.77%	38.60 %
CWE-352	Topic 29	401	910	363	39.89%	90.52%	55.38%	60.04 %
CWE-362	Topic 32	137	1418	43	3.03%	31.39%	5.53%	6.62%
CWE-200	Topic 34	1036	869	265	30.49%	25.58%	27.82%	27.34 %
CWE-59	Topic 37	308	1456	242	16.62%	78.57%	27.44%	31.54 %
CWE-22	Topic 38	1323	2035	1057	51.94%	79.89%	62.95%	65.74 %

This gave us fairly good results for common security risks like SQL injection, however, since precision was not perfect in all cases, the topic labels had to be manually verified and fixed. The reasons for this have been described in the next section 5.5.4.2.

5.5.4.2 Topic Labeling Results

We obtained a sample of vulnerability categories from Appscan tool and Skipfish tools (Appendix A). We ran both against our topic model and mapped the categories to our topic groups. This along with CWE categories helped us to label 23 of the 40 topics automatically as shown in Table 5-8. The similar categories from different scanners being assigned to same topic, helped justify the usefulness of the model. The multiple labeling processes showed that topic modeling algorithms can be used to standardize scanner categories and map to the quantitative model. The sample result of the labeling process is

shown in Table 5-7. Multi-label check.

Table 5-7. Multi-label check

TopicName	Category Description	Scanner Name
Topic 05	SQL Injection	AppScan
Topic 05	Server-side SQL injection (including blind vectors, numerical parameters).	SkipFish
Topic 05	SQL Injection	None (CWE) *
Topic 02	Format String Attack	AppScan
Topic 02	Format string vulnerabilities.	SkipFish
Topic 02	Format String Vulnerability	None (CWE)
Topic 13	Cross-site Scripting	AppScan
Topic 13	Cross-Site Scripting (XSS)	None-CWE
Topic 13	Stored and reflected XSS vectors in document body (minimal JS XSS support present).	SkipFish

None (CWE) * : categorized by max CWE type verification

Combining the results from these different procedures of labeling, we created the final topic labels as shown in Table 5-8 below:

Table 5-8. Final Topic Labels

Topic	Risk
Topic 00	Abuse of Functionality,Integer overflow vulnerabilities.,Numeric Errors
Topic 01	SSI Injection,OS Command Injections ,Server-side shell command injection (including blind vectors).
Topic 02	Format String Attack, XPath Injection,Format string vulnerabilities.,Format String Vulnerability
Topic 04	Information Leakage,Credentials Management
Topic 05	LDAP Injection, OS Commanding, SQL Injection,Server-side SQL injection (including blind vectors, numerical parameters).,SQL Injection
Topic 06	Denial of Service
Topic 08	Session Fixation,Authentication Issues
Topic 09	Attacker-supplied embedded content (stored and reflected).,Redirection to attacker-supplied URLs (stored and reflected).
Topic 11	Input Validation
Topic 12	Content Spoofing,Conflicting MIME / charset info on renderables.,Generic MIME types on renderables.,Incorrect or missing charsets on renderables.,Incorrect or missing MIME types on renderables.
Topic 13	Cross-site Scripting,Cross-Site Scripting (XSS) ,HTML forms with no XSRF protection.,Stored and reflected XSS vectors in document body (minimal JS XSS support present).

Topic 15	Expired or not-yet-valid SSL certificates.,Self-signed SSL certificates.
Topic 16	Bad caching directives on cookie setting responses.,Bad caching directives on less sensitive content,HTTP credentials in URLs.,Mixed content on non-scriptable subresources (optional).,Stored and reflected XSS vectors via HTTP header splitting.
Topic 17	Buffer Overflow,Buffer Errors
Topic 18	Attacker-supplied script and CSS inclusion vectors (stored and reflected).,Stored and reflected XSS vectors via HTTP redirects.
Topic 20	Predictable Resource Location,Resource Management Errors
Topic 21	Configuration
Topic 23	Brute Force, Insufficient Authentication, Insufficient Authorization, Insufficient Session Expiration,Configuration ,Cryptographic Issues ,Permissions, Privileges, and Access Control ,SSL certificate host name mismatches.
Topic 27	Code Injection
Topic 29	Cross-Site Request Forgery (CSRF)
Topic 32	Race Conditions
Topic 34	Explicit SQL-like syntax in GET or POST parameters.,Information Leak / Disclosure
Topic 37	Credential/Session Prediction,Assorted file POIs (server-side sources, configs, etc).,Link Following
Topic 38	Directory Indexing, Path Traversal,Directory listing bypass vectors.,Directory traversal (including constrained vectors).,Path Traversal
Topic 39	External untrusted embedded content.,External untrusted script and CSS inclusion vectors.,Mixed content problems on script and CSS resources (optional).,Server-side XML / XPath injection (including blind vectors).

5.5.4.3 Topic Terms Verification

We used different visual presentations to verify that the topics had the least overlap and that the terms were relevant within the topic. To validate that the number of topics selected, gave the least amount of overlaps and generic topics, we generated a crosstab visualization of topic-terms output file, using structured query language (SQL) script, executed on the topic-terms output file loaded into the MS SQL server, as shown in Figure 5-8 below.

A	B	C	D	E	F	G	H	I	J	K	L
TermValue	Topic-0	Topic-1	Topic-2	Topic-3	Topic-4	Topic-5	Topic-6	Topic-7	Topic-8	Topic-9	Topic-10
applic	0	0	0	0	0	0	1	1	0	1	1
archiv	0	0	0	0	0	0	0	0	0	0	0
argument	0	1	1	1	0	0	1	0	0	0	0
arrai	1	0	0	0	0	0	0	0	0	0	0
articl	0	0	0	0	0	1	0	0	0	0	0
bin	0	1	0	0	0	0	0	0	0	0	0
bit	0	0	0	0	0	0	0	0	0	0	0
blackberri	0	0	0	0	0	0	0	0	0	0	0
blog	0	0	0	0	0	1	0	0	0	0	0
board	0	0	0	0	0	1	0	0	0	0	0
book	0	0	0	0	0	1	0	0	0	0	0
bound	0	0	0	0	0	0	1	0	0	0	0
box	0	0	0	0	0	0	0	0	0	0	0
brows	0	0	0	0	0	0	0	0	0	0	0
browser	0	0	0	0	0	0	0	0	0	1	0
brute	0	0	0	0	0	0	0	0	0	0	0
buffer	1	0	0	0	0	0	0	0	0	0	0
bug	0	0	0	0	0	0	0	0	0	0	0
bugzilla	0	1	0	0	0	0	0	0	0	0	0
build	0	1	0	0	0	0	0	0	0	0	0
bulletin	0	0	0	0	0	0	0	0	0	0	0
bundl	0	0	0	0	0	0	0	1	0	0	0
busi	0	0	0	0	0	0	0	1	0	0	0
bypass	0	0	0	0	0	0	0	0	1	0	0

Figure 5-8. Crosstab visualization verifying topic terms overlap

The crosstab visualizations enabled us to discover common terms between the topics that were frequent but did not help identify the topic theme. For example, in Figure 5-8, the word ‘applic’ is seen to repeat between multiple topic groups but this does not help identify any themes related to the topic. We called these words ‘noise words’ and found that removing these words greatly helped bring forth those words that were useful in labeling the topic.

All clustering algorithms produce clusters, even if the items in the cluster may be irrelevant. It is therefore, imperative to analyze each cluster to verify whether the terms represent a specific topic. Therefore, as a second visual validation procedure, we evaluated the use of tag/word cloud to help visualize the terms in the topic cluster and verify that the topics in the model are relevant to the domain. We considered the number of word-instances of the term in the topic as the weight of that term in the topic. ‘Word

cloud' visualization enabled us to visually verify the term's importance in a topic by using the probability as the weighing factor to adjust the font size as shown in Figure 5-9 and thus, cross-check the topic label assigned. One of the key advantages of tag cloud visualization was the detection of words that were common in many of the topics and had high significant weight. We had initially set the algorithm to remove the most frequently occurring 10 terms. However, some terms were not caught through that setting. We attempted to estimate different values for common word removal threshold, but did not find an efficient method. Instead of determining the right number of terms to filter out, we proposed the development of a custom 'stop word' list with those words that were generic in nature and common due to the domain of analysis. In case of NVD data, the words were 'data,' 'earlier,' and 'cve'. As discovered earlier, in crosstab visualization, these words did not help identify the topic.

Topic 05

Term	Weight
sql	11096.47
inject	5853.65
command	5532.00
index	1464.68
earlier	988.39
action	952.65
multipl	753.49
com	470.04
compon	445.11
joomla	441.45

Topic 13

Term	Weight
script	13779.04
site	6994.38
web	6728.44
html	6592.45
cross	6549.13
xss	6480.85
inject	6020.12
multipl	1260.42
earlier	832.78
index	740.55

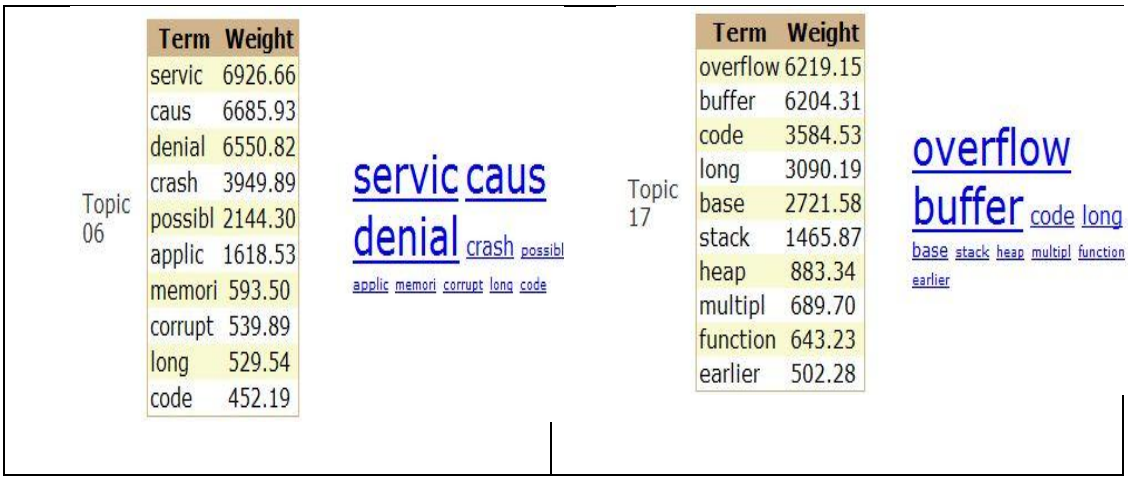


Figure 5-9. Sample Tag cloud of Topic Terms

5.5.4.4 Topic Cluster Cohesion Verification

Topics discovered by the topic modeling algorithm represent the security risk groups for the risk assessment model. The quantitative risk impact value for the model is derived by calculating the average of the CVSS score from the NVD data, for each topic group. To verify the cohesion between the vulnerabilities' data points in the cluster, we calculated the standard deviation for each topic group's average impact score as shown in Table 5-9. Even though the scores have wide ranging value; the standard deviation value is below 2.38, indicating that data points are fairly close in the topic cluster.

Table 5-9. Model Score Standard Deviation Verification

Topic	count	Standard Deviation	Avg. Score	Max Score	Min Score
Topic 05	4320	0.53	7.36	10	2.1
Topic 34	869	0.92	5.18	10	1.8
Topic 13	5594	0.97	4.52	9.4	1.2
Topic 27	1975	1.02	7.20	10	2.6
Topic 38	2035	1.44	5.99	10	1.2
Topic 25	552	1.45	6.70	10	2.6
Topic 29	910	1.53	6.26	10	2.1
Topic 18	588	1.53	8.60	10	2.1
Topic 20	1561	1.59	5.57	10	0
Topic 11	1352	1.63	5.50	10	1.2

Topic	count	Standard Deviation	Avg. Score	Max Score	Min Score
Topic 17	2203	1.66	8.05	10	1.9
Topic 16	884	1.66	5.90	10	0
Topic 08	1471	1.68	6.95	10	1.9
Topic 21	1118	1.71	7.11	10	1.5
Topic 24	808	1.72	6.34	10	1.7
Topic 35	580	1.72	8.32	10	2.1
Topic 36	382	1.73	5.87	10	2.1
Topic 22	446	1.75	6.99	10	1.2
Topic 03	782	1.76	8.17	10	2.1
Topic 04	1235	1.80	5.22	10	1.2
Topic 02	1047	1.83	6.78	10	1.9
Topic 01	666	1.84	7.43	10	1.9
Topic 23	1528	1.93	6.03	10	0
Topic 12	763	1.93	5.59	10	1.2
Topic 31	879	1.94	6.80	10	1.2
Topic 15	1087	1.94	6.24	10	0
Topic 19	1195	1.97	7.84	10	2.1
Topic 26	1737	1.98	6.35	10	0
Topic 00	1196	1.98	7.06	10	1.9
Topic 09	728	2.00	6.38	10	1.8
Topic 33	671	2.05	7.31	10	0
Topic 06	1333	2.07	6.30	10	1
Topic 28	678	2.07	6.53	10	1.9
Topic 14	877	2.11	7.26	10	0
Topic 32	1418	2.15	4.94	10	1.2
Topic 39	723	2.19	6.54	10	2.6
Topic 10	409	2.22	6.82	10	1.2
Topic 37	1456	2.28	4.64	10	0
Topic 30	966	2.32	6.82	10	1.2
Topic 07	1283	2.37	6.03	10	1

5.5.4.5 Impact Score Verification

Our quantitative model consists of the topic groups, its terms, the impact score, and the probability of occurrence, averaged over the number of months from the first published date to the last published date. A paired t-test was performed to determine whether the scores predicted by the model built were good representatives of the

vulnerability they represented.

Our model is based on the hypothesis that the average impact score of the topic model groups correctly predicts the impact score of a potential vulnerability. We validated this hypothesis by performing a t-test, by taking a subset of the NVD data for the year 2012.

H_0: The average impact score of the topic model cluster correctly predicts the impact score of a potential vulnerability. The mean predicted score of each cluster and the mean actual score of the reported vulnerabilities are same.

H_0: $\mu_d = 0 \Rightarrow (\mu_{actual\ score} - \mu_{predicted\ score}) = 0$

where $\mu_d = \mu_{actual\ score} - \mu_{predicted\ score}$

H_1: The average impact score of the topic model cluster does not correctly predicts the impact score of a potential vulnerability. Thus the mean predicted score of each cluster and the mean actual score of the reported vulnerability are not same.

Therefore H_1: $\mu_d > 0$ or $\mu_d < 0$

Table 5-10 shows the result of the t–test performed using a subset of NVD data.

Table 5-10. t-Test: Paired Two Sample for Means

t-Test: Paired Two Sample for Means		
	Actual Score	Predicted Score
Mean	6.47	6.47
Variance	5.34	1.22
Observations	1198	1198
Pearson Correlation	0.66	
Hypothesized Mean Difference	0	
df	1197	
t Stat	-0.0045	

P(T<=t) one-tail	0.49	
t Critical one-tail	1.65	
P(T<=t) two-tail	0.99	
t Critical two-tail	1.96	

The t-test was performed at a significance level of $\alpha=0.05$. We accept the Null hypothesis since the p-value is greater than 0.05 and the test statistic value is less than t-critical value. Thus statistical evidence indicates that the predicted scores are effective presentation of the security risk groups created through topic modeling.

5.5.4.6 Comparative Verification

In the final comparative experiment, we compared the clusters with predefined mapping of the data to CWE using precision and recall test (Table 5-6). Our results match with the findings of Neuhaus where the popular categories like cross-site scripting, Path Traversal , SQL injection did well (Neuhaus & Zimmermann, 2010), where as topics like 'Insufficient Authentication,' which were split between multiple CWE's did worse than popular categories in precision and recall test. 'Cross-Site Request Forgery (CSRF)' and 'Credential/Session Prediction' topic clusters had low precision but very high recall. The reasons for these have been articulated very well in a paper by Neuhaus et al. (Neuhaus & Zimmermann, 2010) . For example, the topic modeling algorithm assigns multiple topics per CVE, whereas there is only one CWE Type assigned per CVE. Second, if a CVE is about buffer over flow, and the text does not have the high proportion of words required, then, it will not be assigned the right topic. Also, sometimes, buffer overflow might cause 'denial of service' risk and may be assigned to that CWE category with NVD. Due to such arbitrary assessment of CWE types to the CVE and the requirement of anchor words, our topic groups and CWE categorization did

not match in few cases.

5.5.4.7 Topic Label Verification

Finally, we analyzed the vulnerability data in each topic cluster to validate that the cluster represented that data accurately. For example, we selected vulnerabilities from NVD, which were categorized into Topic 17, which is labeled to be 'buffer overflow' and examined the summary text. The results in the Table 5-11 showed that the 'Topic' group found by LDA algorithm did reflect the vulnerability reported. Such results had been obtained previously in a study by Neuhaus et al. and this confirmed the verifications of our topic groups (Neuhaus & Zimmermann, 2010).

Table 5-11. Topic CVE verification

Topic	cve_id	Summary
Topic 17	CVE-2002-0149	Buffer overflow in ASP Server-Side Include Function in IIS 4.0, 5.0 and 5.1 allows remote attackers to cause a denial of service and possibly execute arbitrary code via long file names.
Topic 17	CVE-2002-0154	Buffer overflows in extended stored procedures for Microsoft SQL Server 7.0 and 2000 allow remote attackers to cause a denial of service or execute arbitrary code via a database query with certain long arguments.
Topic 17	CVE-2002-0177	Buffer overflows in icecast 1.3.11 and earlier allows remote attackers to execute arbitrary code via a long HTTP GET request from an MP3 client.
Topic 17	CVE-2002-0179	Buffer overflow in xpilot-server for XPilot 4.5.0 and earlier allows remote attackers to execute arbitrary code.
Topic 17	CVE-2002-0198	Buffer overflow in plDaniels ripMime 1.2.6 and earlier, as used in other programs such as xamime and inflex, allows remote attackers to execute arbitrary code via an attachment in a long filename.
Topic 17	CVE-2002-0199	Buffer overflow in admin.cgi for Nullsoft Shoutcast Server 1.8.3 allows remote attackers to cause a denial of service and possibly execute arbitrary code via an argument with a large number of backslashes.
Topic 17	CVE-2002-0204	Buffer overflow in GNU Chess (gnuchess) 5.02 and earlier, if modified or used in a networked capacity contrary to its own design as a single-user application,

		may allow local or remote attackers to execute arbitrary code via a long command.
Topic 17	CVE-2002-0231	Buffer overflow in mIRC 5.91 and earlier allows a remote server to execute arbitrary code on the client via a long nickname.

In the second verification, we utilized string matching similarity algorithm. The topics created by the topic modeling algorithm were presented by a set of terms. We utilized string matching algorithms to compare the similarities between the topic cluster terms and the labels we assigned to the topics through earlier processes. Due to stemming, our words were partial instead of complete. So we did not use the 'Levenstein distance' that indicated the number of changes required to change from string one to string two. We selected the Q-grams algorithm to measure the similarities between the topic terms and topic labels assigned. Q-grams algorithm counted the number of common substrings shared by the two strings (Salmela, Tarhio, & Kytjoki, 2007). Our assumption was that there would be semantic similarity between the topic terms and the labels obtained from scanner documentation. We found that it helped to merge topics. We set the threshold that everything over .05 would be considered as good string matches. Table 5-12 shows the result of this process and indicates string matching algorithm as a good solution for Topic labeling if the wordlist can be generated from a particular domain.

Table 5-12. Topic Top Terms & Scanner Categories - String Match Q-Gram

Topic	Risk	Top Terms	scanner name	Score
Topic 05	Server-side SQL injection (including blind vectors,	sql , inject , command , index , earlier , action , multipl , com , joomla , compon , _id , magic , disabl , _quotes_gpc , usernam , cat	SkipFish	0.084

Topic	Risk	Top Terms	scanner name	Score
	numerical parameters).	, new , asp , view , page , categori , aka , catid , cid , beta , portal , search , mambo , board , articl , possibl , book , link , list , classifi , login , blog , pid , pro , modul		
Topic 05	SQL Injection	sql , inject , command , index , earlier , action , multipl , com , joomla , compon , _id , magic , disabl , _quotes_gpc , usernam , cat , new , asp , view , page , categori , aka , catid , cid , beta , portal , search , mambo , board , articl , possibl , book , link , list , classifi , login , blog , pid , pro , modul	None-CWE	0.051
Topic 06	Denial of Service	servic , caus , denial , crash , possibl , applic , memori , corrupt , long , code , craft , hang , daemon , free , trigger , unspecifi , version , earlier , bound , engin , multipl , argument , doubl , oper , string , shutdown , empti , platform , devic , nicknam , ircd , audio , persist , unknown , openttd , alpha , seri , instabl , thread , counter	AppScan	0.050
Topic 22	Permissions, Privileges, and Access Control	user , directori , search , path , local , librari , share , trojan , hors , untrust , current , folder , board , work , dll , privileg , gain , malici , code , load , point , program , possibl , power , locat , modifi , place , earlier , invis , pathnam , contain , python , variabl , technolog , demonstr , _library_path , bulletin , zero , unc , version	None-CWE-(manual)	0.059

5.5.4.8 Content Validity of Topics

Having clustered the vulnerability data into topics, we analyzed the properties of the topics with regards to severity, confidentiality, integrity etc. We verified the security

metrics for different security risk topics, to analyze if the topic's impact probabilities provided known facts and thus could be used to derive facts for the unknown topic categories. We calculated the probability of different levels for the metrics, severity, confidentiality, and integrity for each topic and selected the level with highest probability in each metrics group. For example, in our model, 'Topic 5' refers to the 'SQL injection' risk. We have the following probability dimensions for 'Topic 5' security groups based on the level that has the maximum likelihood, as shown in Table 5-13.

Table 5-13. Topic Probabilities

Topic	None Availability	Partial Availability	Complete Availability
Topic 05	0.025	0.9657	0.0092
Topic	Low Severity	Medium Severity	High Severity
Topic 05	0.0006	0.1303	0.8689
Topic	None Integrity	Partial Integrity	Complete Integrity
Topic 05	0.0027	0.9879	0.0092
Topic	Low Complexity	Medium Complexity	High Complexity
Topic 05	0.9129	0.0745	0.0125
Topic	None Confidentiality	Partial Confidentiality	Complete Confidentiality
Topic 05	0.0125	0.9780	0.0094
Topic	ADJACENT_ NETWORK Vector	NETWORK Vector	LOCAL Vector
Topic 05	0.0004	0.9986	0.0009
Topic	NONE Authentication	MULTIPLE_ INSTANCES Authentication	SINGLE_ INSTANCE Authentication
Topic 05	0.9717	0	0.0282

Based on this data, for the 'SQL injection' security risk, we could make the following statement using the CVSS guide (Peter Mell, 2012). In case of the 'SQL Injection risk,' the highest probability of impact on 'integrity' was 'partial' since some tables were exposed rather than the complete system. The probability for 'access complexity' needed

to exploit this risk was ‘Low’ signifying that no additional access or specialized circumstances were needed for the exploit to be successful. With a very high probability, this exploit was possible without requiring any authentication**.** The impact on availability was mostly ‘partial’, indicating it could result in reduced performance. Also, the results indicated that with a high probability, the vulnerability was exploitable locally as well as to a user within the network. Overall, the severity of attack was high for this risk category. These findings were well known facts for ‘SQL injection’ vulnerability and thus our model reflected the content it represented.

5.5.4.9 Topic Trend Analysis

The next verification was performed to check if the topic security risk groups demonstrated the trends noticed in prior research and industry findings with regard to vulnerability discovery. Table 5-14 shows the result of trend analysis of currently active risks. In terms of analyzing the five-year trend of the security risk topic groups, we find that large numbers of issues still belong to the same security groups, as noticed by prior research. Table 5-14 shows the change in top-10 risks and demonstrates changes through colors, if any, over the last 6 years. Security risks, such as Information Leakage, SQL Injection, Input Validation, Cross-site Scripting, Buffer Overflow, Resource Management Errors and Permissions, Privileges, and Access Control, have been dominant contributors to vulnerability databases. We also noticed some re-emerging risks in our top-10 list. For example, ‘Attacker-supplied script and CSS inclusion vectors (stored and reflected),’ and ‘Integer overflow’ risks have been steadily moving up in rank since 2009 based on number of issues reported and are now among the top-10.‘Code injection’ issues have dropped off as one of the top-10 issues. Thus, such models can be used to demonstrate

and flag re-emerging risk groups on a security dashboard.

Table 5-14. Top 10 Risks by Year

Topic	2006	Topic	2007	Topic	2008	Topic	2009	Topic	2010	Topic	2011
Topic 13	1231	Topic 23	979	Topic 05	1592	Topic 13	749	Topic 23	885	Topic 23	733
Topic 05	1064	Topic 13	844	Topic 13	1032	Topic 05	708	Topic 13	612	Topic 17	537
Topic 23	957	Topic 05	786	Topic 23	908	Topic 23	665	Topic 05	586	Topic 13	430
Topic 27	907	Topic 17	776	Topic 17	580	Topic 17	560	Topic 17	527	Topic 20	391
Topic 17	414	Topic 27	624	Topic 38	387	Topic 38	223	Topic 20	234	Topic 18	241
Topic 04	311	Topic 20	273	Topic 08	332	Topic 11	208	Topic 38	222	Topic 11	231
Topic 20	262	Topic 38	263	Topic 27	279	Topic 27	197	Topic 11	220	Topic 04	228
Topic 38	252	Topic 11	245	Topic 20	263	Topic 32	193	Topic 32	180	Topic 06	142
Topic 11	216	Topic 04	230	Topic 11	222	Topic 20	165	Topic 27	175	Topic 00	99
Topic 08	184	Topic 08	199	Topic 04	215	Topic 08	157	Topic 18	168	Topic 05	96

In addition to monitoring top-ten risks, the trend analysis compared the percentage change and provided an overview of issues that had changed in importance as shown in Table 5-15. Compared to Table 5-14, we see that even though ‘integer overflow’ is re-emerging as a top-ten risk, the trend is showing a reduction from the previous year and then, seems to have increased significantly in 2012. ‘Information leakage’ a member of the top-ten risk group has had a significant rise in 2011. In addition, comparison of the data in the Table 5-14 and Table 5-15 indicates that the ‘Topic 32’ must be monitored as it is merging in terms of frequency and impact. Finally ‘Denial of service,’ which is closely related to ‘resource management’ and is the result of many different security vulnerabilities seems to be rising in 2012, as reported by industry news article (Proofpoint - Enterprise Email Security, 2012). The unspecified topics 18 and 19, which were represented by the term ‘unknown’ with high probability, reduced from 2011 to

2012, and this may be attributed to those vulnerabilities being properly described as information is available and their subsequent categorization into related topics.

Table 5-15. Topic Vulnerability change from year 2007-2012

Risk	Topic	Change 2007	Change 2008	Change 2009	Change 2010	Change 2011	Change 2012
Integer Overflow	Topic 00	24.65%	-3.39%	-12.87%	-0.67%	-33.11%	91.92%
OS Command Injections	Topic 01	53.13%	34.69%	-31.82%	-26.67%	-18.18%	133.33 %
Format String Vulnerability	Topic 02	0.63%	-32.70%	-29.91%	-16.00%	-39.68%	23.68%
Credentials Management	Topic 04	-2.74%	20.42%	-48.54%	-21.59%	11.59%	16.88%
SQL Injection	Topic 05	-26.13%	102.54%	-55.53%	-17.23%	-83.62%	121.88 %
Denial Of Service	Topic 06	-3.23%	0.83%	9.09%	-17.42%	30.28%	7.04%
Authentication Issues	Topic 08	8.15%	66.83%	-52.71%	-67.52%	-13.73%	140.91 %
IE	Topic 09	-37.65%	-43.40%	50.00%	-11.11%	-2.50%	123.08 %
Input Validation	Topic 11	13.43%	-9.39%	-6.31%	5.77%	5.00%	6.93%
Content Spoofing	Topic 12	-7.69%	-15.48%	-25.35%	-7.55%	12.24%	-14.55%
Cross-Site Scripting (XSS)	Topic 13	-31.44%	22.27%	-27.42%	-18.29%	-29.74%	119.53 %
SSL certificates	Topic 15	-26.96%	0.00%	1.19%	-52.94%	0.00%	160.00 %
Bad caching directives	Topic 16	68.18%	-43.24%	-19.05%	-9.80%	-10.87%	9.76%
Buffer Errors	Topic 17	87.44%	-25.26%	-3.45%	-5.89%	1.90%	14.71%
Unspecified Topic 18	Topic 18	18.75%	-15.79%	312.50 %	154.55%	43.45%	-25.73%
Resource Management Errors	Topic 20	3.03%	-17.65%	-39.29%	5.88%	2.78%	44.59%
Permissions, Privileges, and Access Control	Topic 23	2.30%	-7.25%	-26.76%	33.08%	-17.18%	70.94%

Risk	Topic	Change 2007	Change 2008	Change 2009	Change 2010	Change 2011	Change 2012
Code Injection	Topic 27	-31.20%	-55.29%	-29.39%	-11.17%	-57.71%	29.73%
Race Conditions	Topic 32	25.00%	-15.68%	23.72%	-6.74%	-47.78%	14.89%
Information Leak / Disclosure	Topic 34	-46.67%	-50.00%	-18.18%	-52.78%	788.24 %	-80.79%
Link Following	Topic 37	-3.85%	99.00%	-67.84%	17.19%	1.33%	7.89%
Path Traversal	Topic 38	4.37%	47.15%	-42.38%	-0.45%	-71.62%	31.75%
FireFox	Topic 39	-27.27%	25.00%	15.56%	19.23%	-22.58%	86.46%
Unknown Topic 19	Topic 19	5.38%	10.22%	-35.76%	67.01%	95.68%	-11.67%

5.6 Demonstration

To demonstrate the procedure and usage of the measurement tool, we selected open source health record software *openEMR*[14] and *openMRS*[15] as candidate software products. *openEMR* is developed in PHP with MySQL as the backend. *openMRS* is developed in Java with MySQL as backend. The test environment consists of Windows 7 laptop with both Apache Http server and Apache Tomcat server installed. *openEMR* was hosted on an Apache Http server and *openMRS* was hosted on an Apache Tomcat server, as specified by their respective system requirements documentation. These systems were scanned for vulnerabilities using the open source web application scanner *skipfish (Michal Zalewski, 2012)*.

[14] http://www.oemr.org/
[15] http://openmrs.org/

5.6.1 System Requirements Risk Analysis

The steps followed here were:

"Step 1: Inspect system requirements and identify major COTS components, such as database server, web server, etc.

For openMRS, we identified the following system requirements to host the product: Apache Tomcat 6 and MySQL 6

Step 2: Find base score; exploit score and the number of vulnerabilities for each component "(Ruma Das, 2012)

An example for the component MySQL is given in Table 5-16.

Table 5-16. Sample CVE base score, exploit score(Ruma Das, 2012)

CVE ID	Base score	Exploit score
CVE-2004-0388	2.1	3.9
CVE-2004-0627	10	10
CVE-2004-0628	10	10
CVE-2005-1636	4.6	3.9
CVE-2005-2558	4.6	3.9
CVE-2005-2572	8.5	6.8
CVE-2005-2573	5	10
CVE-2006-0369	2.1	3.9
CVE-2006-0903	4.6	3.9

Step 3: Calculate the component security score as follows using the impact score formula proposed by Sui et al.(Chenmeng Sui, 2012) (Ruma Das, 2012)

$$\textbf{Component Security Risk Score} = \frac{\sum_{i=1}^{n} \boldsymbol{Base\ Score\ X\ Exploit\ Score}}{\boldsymbol{n}} \quad (\mathbf{1})$$

Equation 5-1.Formula proposed by Sui et al.(Chenmeng Sui, 2012)

Where n = number of vulnerabilities in the component

Step 4: Then, consider the 'exposure score.' An OSS product installed on a public facing server is more vulnerable than on a private server. Therefore, we assigned a weight to each based on exposure level (Ruma Das, 2012).

Step 5: Finally, we considered the protection levels of the environment where it would be installed. We provided three values, namely (Low, Medium, High), for practitioners to select from based on their professional judgment (Ruma Das, 2012). The final score for the System Requirement Risk dimension is was given by Equation 5-2 below:

Total System Requirement Security Risk score = Component Score * Exposure Weight * Protection Level

Equation 5-2. Total System Requirement Security Risk score(Ruma Das, 2012)

The calculation and final results of system requirements risk analysis are shown in Table 5-17 below for product openMRS.

Table 5-17. openMRS System Requirements Risk Score(Ruma Das, 2012)

Host Components	Security Risk Score	Exposure Weight	Protection level	
Apache Tomcat 6	C1 =41.63	External E1=2	Low P1=3	C1*E1*P1= 249.8
MySQL 6	C2=33.81	Internal E2=1	High P2=1	C2*E2*P2= 33.81
Apache Windows 7	C3=13.33	External E3=2	Medium P3=2	C3*E3*P3= 53.32
Tomcat Windows 7	C4=13.33	Internal E4=1	High P4=1	C3*E3*P3= 13.33
Total System Requirement Security Risk score				350.26

Similarly, all the five steps mentioned above for system requirements risk analysis

calculation, were undertaken for the next product openEMR as shown in Table 5-18 below.

Table 5-18. openEMR System Requirements Risk Score(Ruma Das, 2012)

Host	Component Score	Exposure Weight	Protection level	
Apache 2.2	37.59	External=2	Low=3	225.54
MySQL 6	33.81	Internal=1	High=1	33.81
Apache Windows 7	13.33	External=2	Medium=2	53.32
Tomcat Windows 7	13.33	Internal=1	High=1	13.33
Total Component Security Risk score				326.00

An extension of this research would be to develop a tool support for this procedure, which could assist the practitioners to perform an if-then analysis by changing the operating system type, exposure score, and protection levels based on their knowledge of the environment (Ruma Das, 2012).

5.6.2 Product Design Risk Analysis

We conducted dynamic analysis to find the security risks introduced into the product due to incorrect coding practices or design defects that were internal to the product.

5.6.2.1 Dynamic Analysis

The dynamic analysis was performed by installing and making the products operational on the test environment. We used the web application scanner ‘skipfish’ for scanning the candidate products. A list of representative vulnerabilities found by dynamic analysis of openMRS using SkipFish is provided below (Michal Zalewski, 2012).

Table 5-19. openMRS Dynamic Analysis(Ruma Das, 2012)

Name of the Representative Threat	Number of Vulnerabilities Found in Candidate 1 (openMRS)
Integer overflow vector	4
Incorrect or missing charset	7
Incorrect or missing MIME type	1
HTML form with no apparent XSRF protection	2

One of the challenges we addressed using topic modeling is the ability to map scanner categories to the security risk groups. The reason this was necessary is that some CVE's are assign CWE types but the scanner categories do not map to CWE type always. "Some of the scanner categories had direct mapping to CWE but others did not, for example, integer overflow maps to CWE-190 whose parent is CWE-189 - Numeric Errors. Certain vulnerabilities in NVD have been mapped to CWE-189 - Numeric Errors, while others such as 'Incorrect or missing charset' had no mapping in CWE used by NVD. However, using the previously trained topic model, we were able to assign a topic to each of the skipfish vulnerability types and calculate the score"(Ruma Das, 2012).

Table 5-20 Skipfish Vulnerability Types-Topic Term Assignment (Ruma Das, 2012)

Name of the Representative Threat(Michal Zalewski, 2012)	Topic
Integer overflow vector	Topic 00
Incorrect or missing charset	Topic 12
Incorrect or missing MIME type	Topic 12
HTML form with no apparent XSRF protection	Topic 13

The security score was calculated using the method proposed by Wang et al. (Wang,

Wang, et al., 2009). The process involved counting the number of vulnerabilities per topic and determining the frequency as shown in Table 5-21 and Table 5-22.

Table 5-21. Sample Topic vulnerability count and date range(Ruma Das, 2012)

Topic	Vulnerability Count	Min Date	Max date
Topic 00	1196	1995-08-31	2012-05-29
Topic 12	763	1997-01-01	1997-01-01
Topic 13	5594	1999- 12-31	2012-06-04

Table 5-22. Sample Topic vulnerability count and Frequency(Ruma Das, 2012)

Topic Name	Vulnerability Count V	Number of Months Min Date-Max Date M	Frequency Per Month V/M
Topic 00	1196	200	5
Topic 12	763	184	4
Topic 13	5594	149	37

Average Impact Score for each topic Cluster is calculated as:

$$\text{Average Score} = \sum_{i=1}^{n} base\ Score_i$$

Where n=number of vulnerability in the Topic.

Table 5-23. openMRS Topic Average score and Frequency (Ruma Das, 2012)

Topic Name	Average Score	Frequency Per Month
Topic 00	7.06	F1=5
Topic 12	5.59	F2=4
Topic 13	4.52	F3=2

Finally, it was normalized using a weight parameter, which was the ‘number of vulnerability found in Candidate 1’ for each topic(Ruma Das, 2012) as shown in Table

5-24.

Table 5-24. openMRS vulnerability weight analysis(Ruma Das, 2012)

Topic Name	Average Impact Score (S)	Probability P	Weight (W)
Topic 00	S1=7.06	P1=F1/F1+F2+F3=0.0869	W1=4
Topic 12	S2=5.76	P2=F2/F1+F2+F3=0.1086	W2=8
Topic 13	S3=4.53	P3=F3/F1+F2+F3=0.8043	W3=2

The final product design risk for openMRS is given as :

Product Design Risk Score= S1*P1*W1 + S2*P2*W2 = 14.24

5.6.2.2 openEMR

Next, we installed ‘openEMR’, the second candidate software product and performed scanning using skipfish scanner. Since this product was installed on a different web server than product one, we found a different system requirement risk score. The Table 5-25 showed a sample out obtained through dynamic scanning of openEMR.

Table 5-25. Sample openEMR Dynamic Analysis(Ruma Das, 2012)

Name of the Representative Threat	Number of Vulnerabilities Found in Candidate 1 (openEMR)
Integer overflow vector	4
Incorrect or missing charset	8
Incorrect or missing MIME type	1
HTML form with no apparent XSRF protection	2

In the case of openEMR, for product design risk analysis, the types of vulnerability categories reported in our experiment remained the same, the while number of

vulnerabilities (weight w) found was different (Ruma Das, 2012). .

Table 5-26. openEMR vulnerability weight analysis(Ruma Das, 2012)

Topic Name	Average Score (S)	Probability (P)	Weight (W)
Topic 00	S1=7.06	P1=F1/F1+F2+F3= 0.019	W1=4
Topic 12	S2=5.76	P2=F2/F1+F2+F3=0.015	W2=9
Topic 13	S3=4.53	P3=F3/F1+F2+F3=0.8043	W3=2

Product Design Risk Score = S1*P1*W1 + S2*P2*W2 = 14.73

The tables Table 5-27and Table 5-28 show the final result of implementing the procedure, on the two candidate products selected for comparison, using the quantitative security risk assessment model.

The scores are to be interpreted only to evaluate the security risk levels between the 2 products and for comparison between them. The scores do not have meaning by themselves and do not indicate the risk level of the product itself. The example below indicates that openMRS has higher system requirement risk but lower product design risk as compared to openEMR.

Table 5-27. Quantitative Security Risk evaluation of openMRS(Ruma Das, 2012)

System Requirement Risk	350.26
Product Design Risk	
Dynamic Analysis Risk	14.28

Table 5-28. Quantitative Security Risk evaluation openEMR(Ruma Das, 2012)

System Requirement Risk	326
Product Design Risk	
Dynamic Analysis Risk	14.74

"The scores of the individual dimensions were not aggregated since they could end up being the same on aggregation. Individual comparison of each dimension was nevertheless, suggested" (Ruma Das, 2012).

5.7 The Prototype

The prototype demonstrates the functionality of a tool that based on this procedure. The tool has two main parts: the backend and the frontend as shown in Figure 5-10.

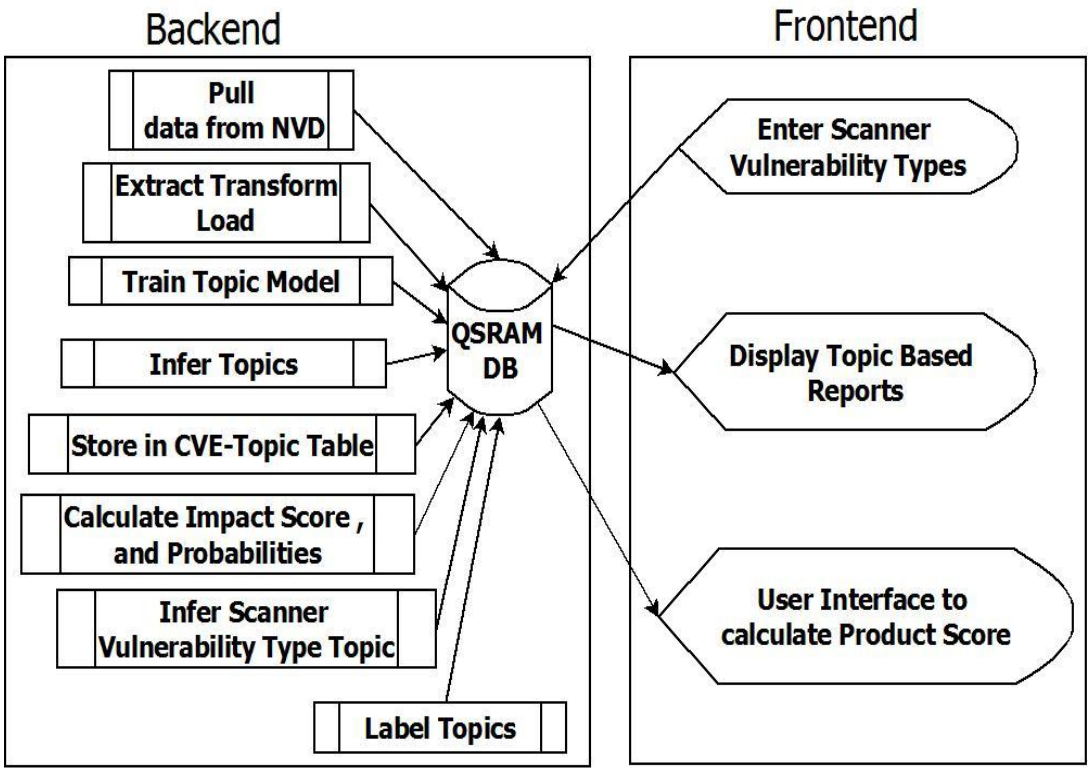


Figure 5-10. Prototype Design

The backend consist of database components and processes to:

Download and load from NVD XML file periodically

Perform Data processing(Extract, Transform and Load process)

Train the Model (one time)

Infer Topic (First time and for all new data)

Update and Maintain CVE Topic Mapping(CVE-TOPIC table)

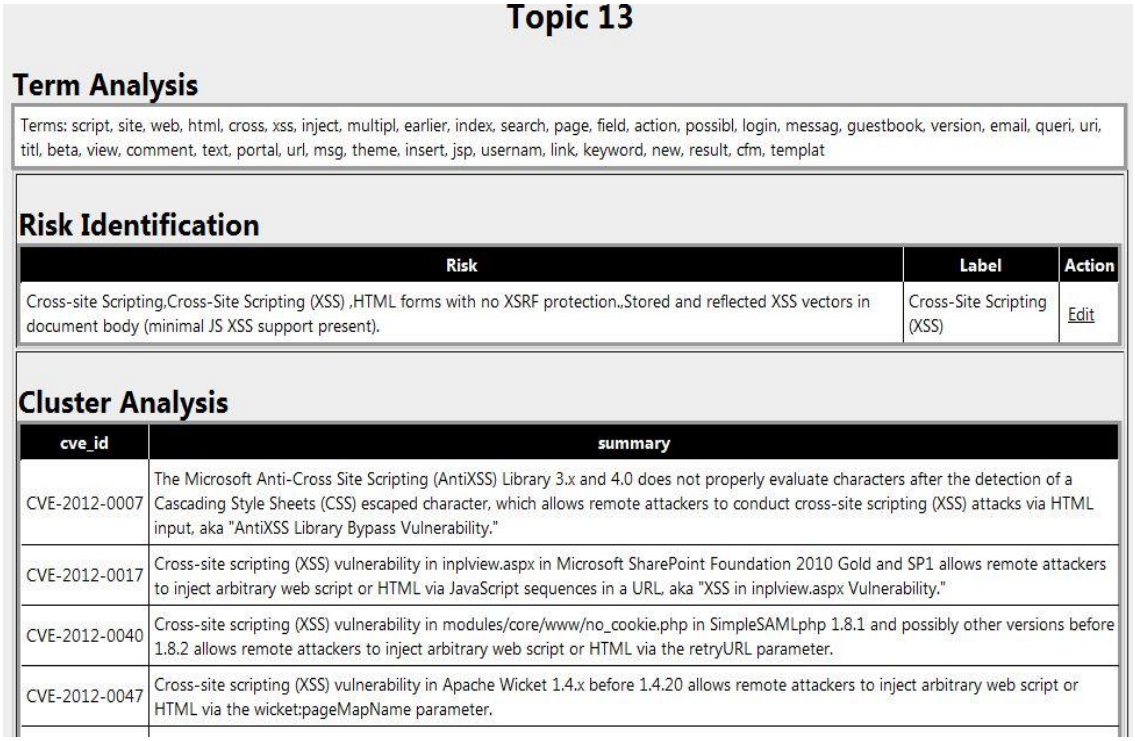
Infer Topic for Scanner Vulnerability Types

Label Topics based on scanner mapping

The frontend consist of the user interface that allows practitioners to analyze and utilize the data. It consists of an interface to enable practitioners to a perform topic-term analysis, topic cluster analysis, verification and correction of topic labels, validation of risks identified, and monitoring of risk impacts.

5.7.1 Topic Cluster Analysis

The purpose of this screen is to display the risks identified by mapping the scanner vulnerability types to the topic and allowing the practitioner to assign single labels. The screen (Figure 5-11) below shows ‘topic 13’ and the related terms identified by the topic modeling algorithm. It displays the CVEs located within that cluster to allow analysts to verify the topic label.



Topic 13

Term Analysis

Terms: script, site, web, html, cross, xss, inject, multipl, earlier, index, search, page, field, action, possibl, login, messag, guestbook, version, email, queri, uri, titl, beta, view, comment, text, portal, url, msg, theme, insert, jsp, usernam, link, keyword, new, result, cfm, templat

Risk Identification

Risk	Label	Action
Cross-site Scripting,Cross-Site Scripting (XSS) ,HTML forms with no XSRF protection.,Stored and reflected XSS vectors in document body (minimal JS XSS support present).	Cross-Site Scripting (XSS)	Edit

Cluster Analysis

cve_id	summary
CVE-2012-0007	The Microsoft Anti-Cross Site Scripting (AntiXSS) Library 3.x and 4.0 does not properly evaluate characters after the detection of a Cascading Style Sheets (CSS) escaped character, which allows remote attackers to conduct cross-site scripting (XSS) attacks via HTML input, aka "AntiXSS Library Bypass Vulnerability."
CVE-2012-0017	Cross-site scripting (XSS) vulnerability in inplview.aspx in Microsoft SharePoint Foundation 2010 Gold and SP1 allows remote attackers to inject arbitrary web script or HTML via JavaScript sequences in a URL, aka "XSS in inplview.aspx Vulnerability."
CVE-2012-0040	Cross-site scripting (XSS) vulnerability in modules/core/www/no_cookie.php in SimpleSAMLphp 1.8.1 and possibly other versions before 1.8.2 allows remote attackers to inject arbitrary web script or HTML via the retryURL parameter.
CVE-2012-0047	Cross-site scripting (XSS) vulnerability in Apache Wicket 1.4.x before 1.4.20 allows remote attackers to inject arbitrary web script or HTML via the wicket:pageMapName parameter.

Figure 5-11. Topic Cluster Analysis Screen

5.7.2 Term Analysis

The ‘term analysis’ screen (Figure 5-12) allows practitioners to correct the labels based on ‘word cloud’ visualization. It also displays the risks identified by mapping the scanner vulnerability types to the topic and allows the practitioner to assign single labels. Finally, it displays the impact on confidentiality, integrity, and availability (CIA) metrics based on calculated maximum probabilities for the three levels.

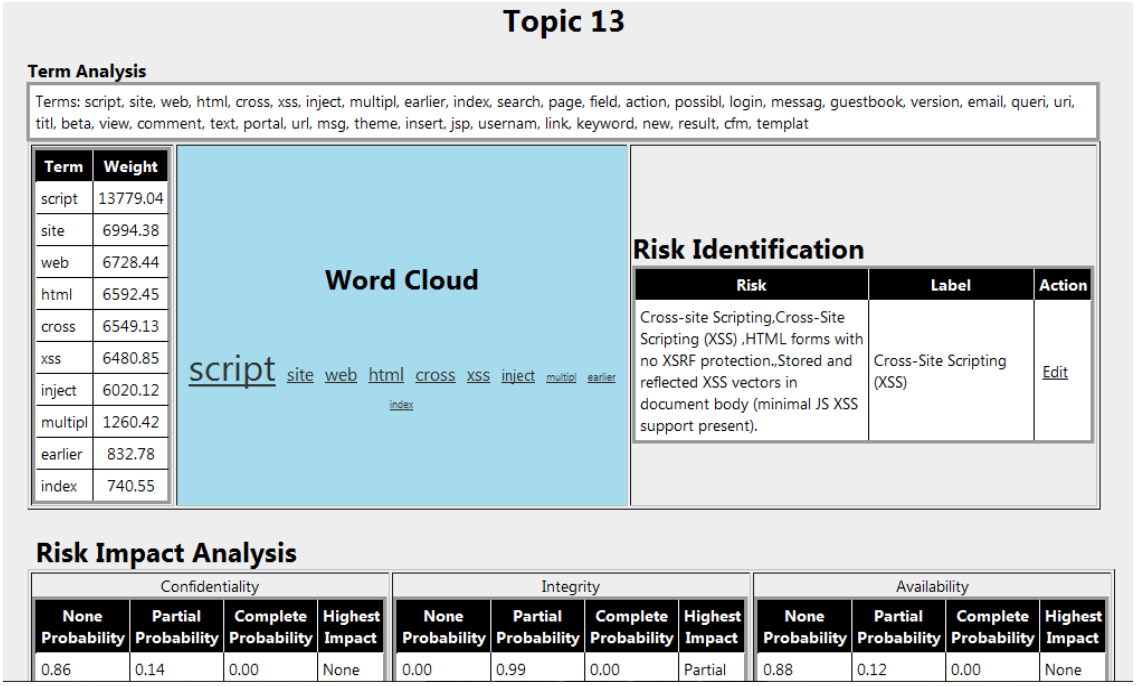


Figure 5-12. Term Analysis Screen

5.7.3 Comparative Analysis of CIA metrics

This screen (Figure 5-13) allows practitioners to view the CIA metrics for all the topics together to perform a comparative analysis.

Topic Confidentiality Probability	
Topic 00	Partial
Topic 01	Partial
Topic 02	Partial
Topic 03	Complete
Topic 04	Partial
Topic 05	Partial
Topic 06	None
Topic 07	Partial
Topic 08	Partial
Topic 09	Partial
Topic 10	Partial
Topic 11	None
Topic 12	Partial
Topic 13	None
Topic 14	Complete
Topic 15	Partial
Topic 16	Partial
Topic 17	Complete
Topic 18	Complete

Topic Integrity Probability	
Topic 00	Partial
Topic 01	Partial
Topic 02	Partial
Topic 03	Complete
Topic 04	None
Topic 05	Partial
Topic 06	None
Topic 07	Partial
Topic 08	Partial
Topic 09	None
Topic 10	Partial
Topic 11	None
Topic 12	Partial
Topic 13	Partial
Topic 14	Complete
Topic 15	Partial
Topic 16	None
Topic 17	Complete
Topic 18	Complete

Availability Probability	
Topic 00	Partial
Topic 01	Partial
Topic 02	Partial
Topic 03	Complete
Topic 04	None
Topic 05	Partial
Topic 06	Partial
Topic 07	None
Topic 08	Partial
Topic 09	Partial
Topic 10	Partial
Topic 11	Partial
Topic 12	None
Topic 13	None
Topic 14	Complete
Topic 15	Partial
Topic 16	Partial
Topic 17	Complete
Topic 18	Complete

Figure 5-13. Comparative Analysis of CIA metrics Screen

5.7.4 System Requirements Risk

This is the starting screen for calculating the software product score. In this screen, the user enters the system requirements and calculates the system requirement risk score, using the calculation shown in Table 5-17.

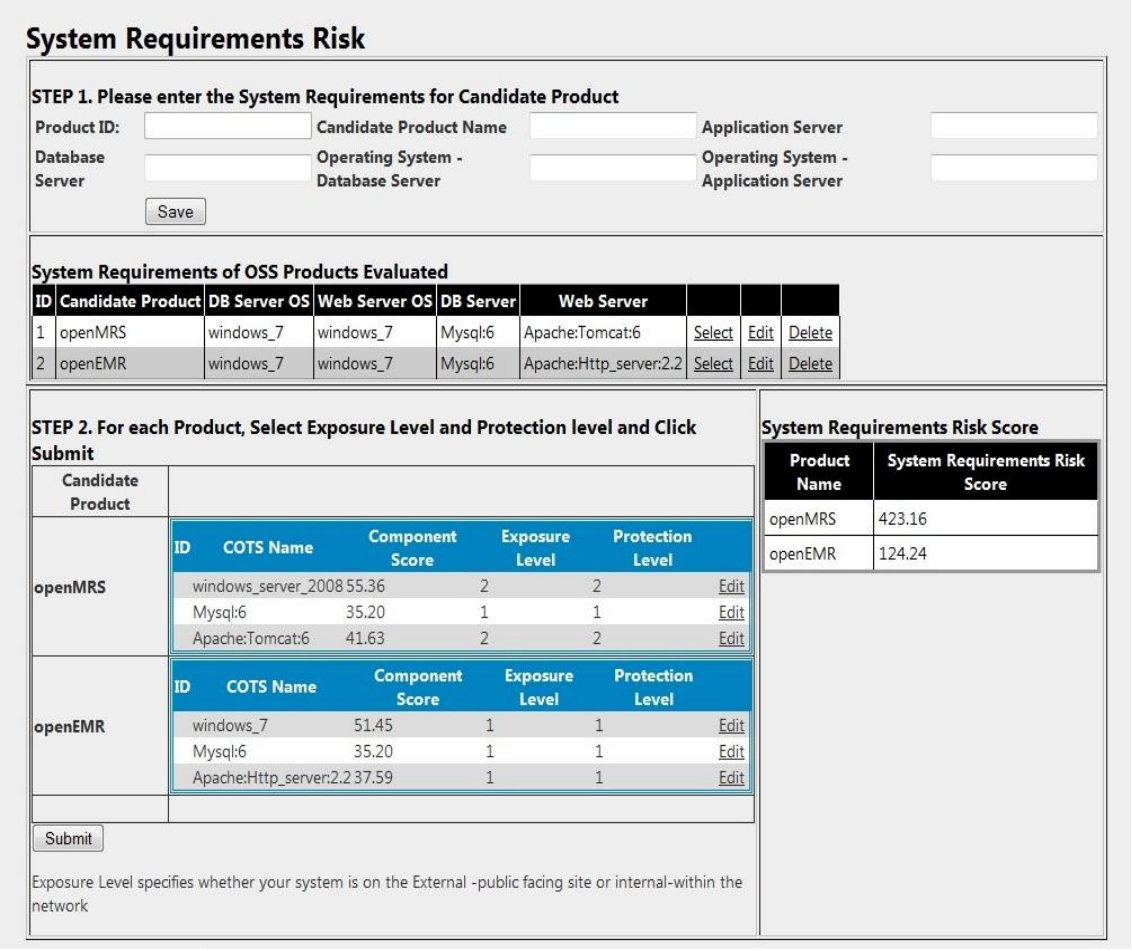


Figure 5-14. System Requirements Risk

In prototype Figure 5-14, we have attempted to show an example to demonstrate the difference in 'system requirement risk score', if the exposure level and protection level are different between the two products. Exposure Level=1 is for 'within firewall', and Exposure Level=2, is 'outside firewall'. For Protection Level, we have the following range: Low =3, Medium=2 and High=1. Thus the prototype indicates that suppose we choose to install one product (openEMR in example above) inside the firewall and have high protection levels, its security risk will be lower as compared to second product, which we choose to install outside the firewall.

5.7.5 Product Design Risk

This is the screen for calculating the product design risk score. In this screen, the user enters the vulnerabilities found using the scanner and calculates the product design risk score.

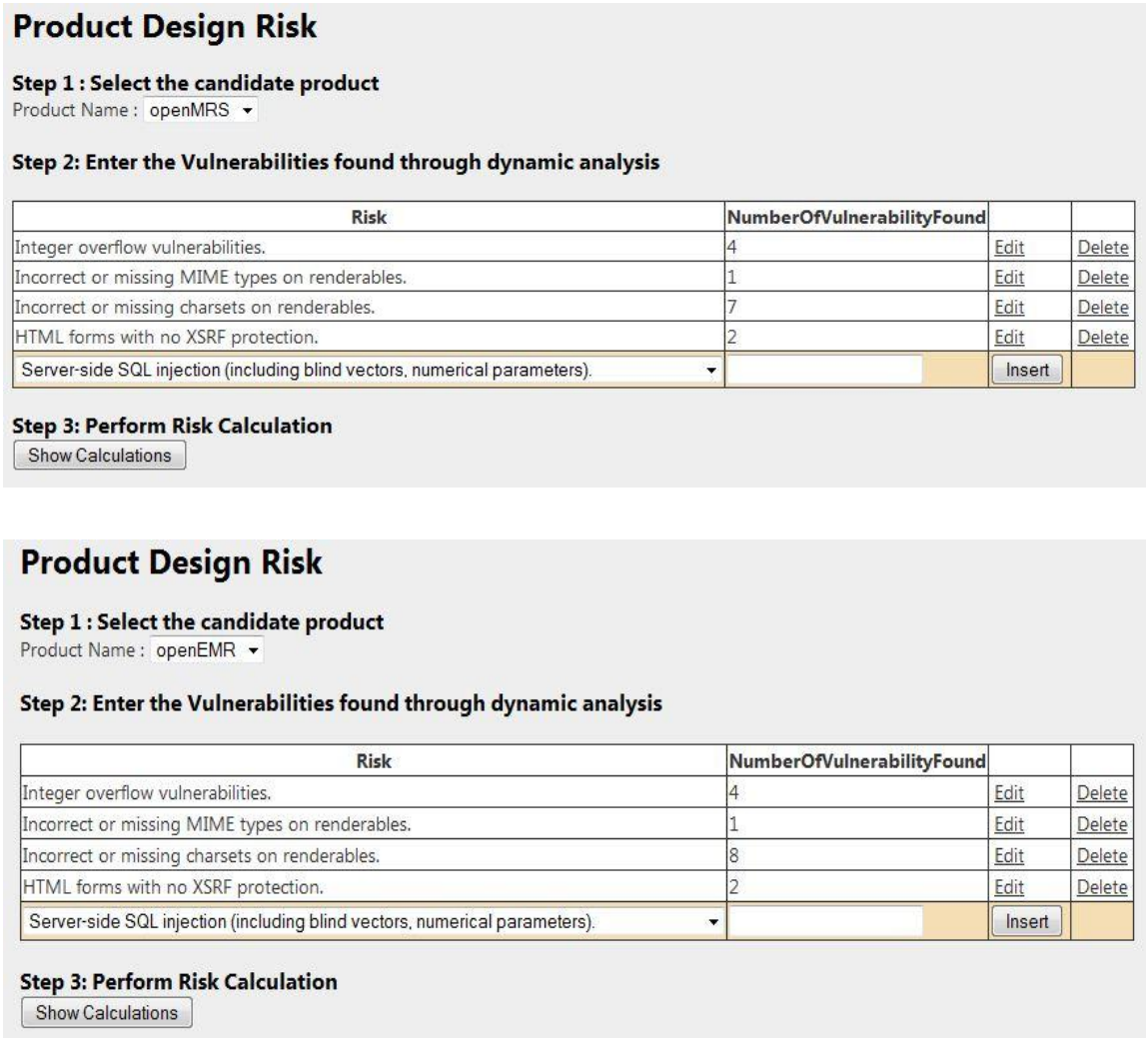
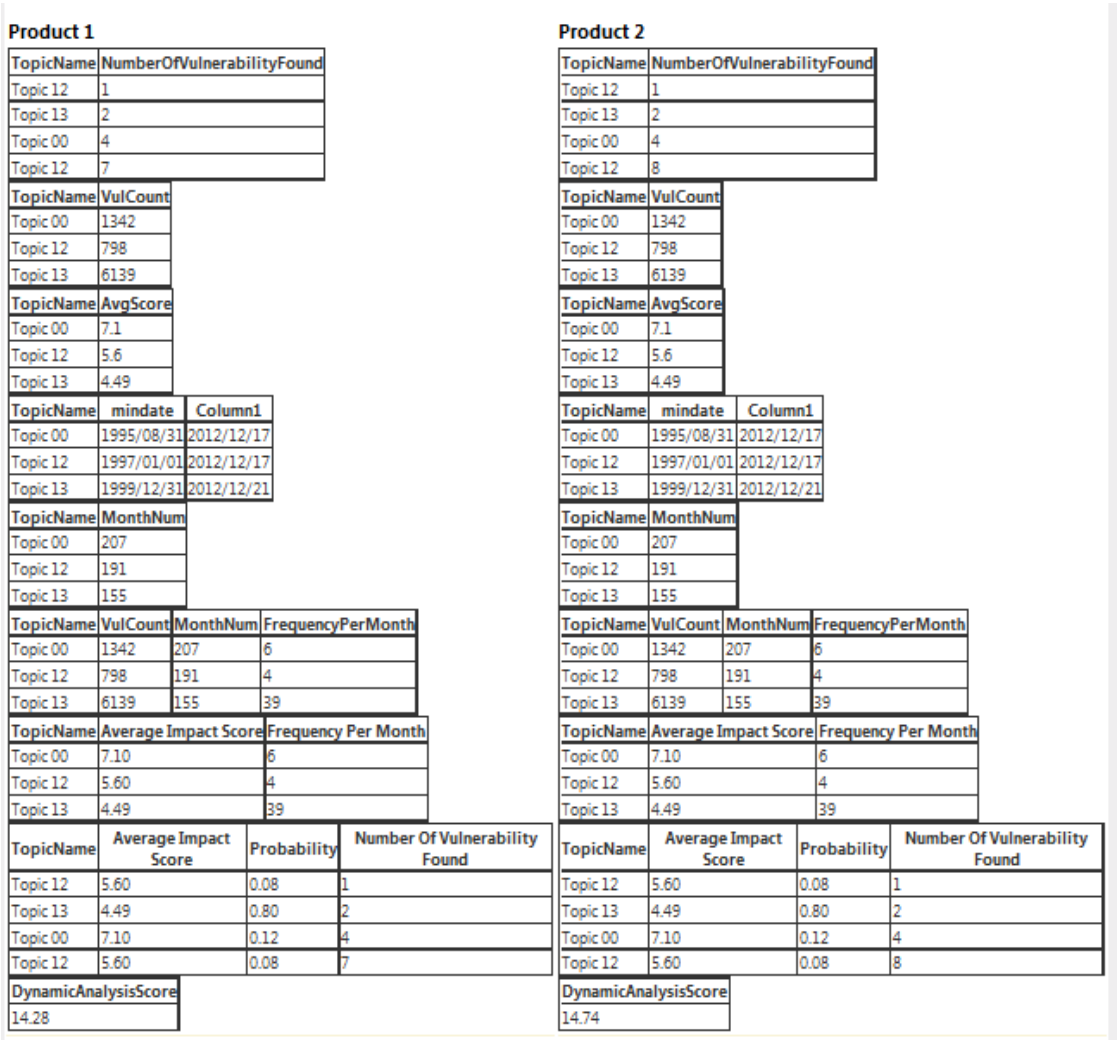


Figure 5-15. Product Design Risk Screen

5.7.6 Product Risk Score Calculation

The next screen demonstrates step-by-step calculation of the product design risk score

as explained in 5.6.2 Product Design Risk Analysis. In Figure 5-16 , Product 1 is ‘openMRS’ and Product 2 is ‘openEMR’.



Product 1

TopicName	NumberOfVulnerabilityFound
Topic 12	1
Topic 13	2
Topic 00	4
Topic 12	7

TopicName	VulCount
Topic 00	1342
Topic 12	798
Topic 13	6139

TopicName	AvgScore
Topic 00	7.1
Topic 12	5.6
Topic 13	4.49

TopicName	mindate	Column1
Topic 00	1995/08/31	2012/12/17
Topic 12	1997/01/01	2012/12/17
Topic 13	1999/12/31	2012/12/21

TopicName	MonthNum
Topic 00	207
Topic 12	191
Topic 13	155

TopicName	VulCount	MonthNum	FrequencyPerMonth
Topic 00	1342	207	6
Topic 12	798	191	4
Topic 13	6139	155	39

TopicName	Average Impact Score	Frequency Per Month
Topic 00	7.10	6
Topic 12	5.60	4
Topic 13	4.49	39

TopicName	Average Impact Score	Probability	Number Of Vulnerability Found
Topic 12	5.60	0.08	1
Topic 13	4.49	0.80	2
Topic 00	7.10	0.12	4
Topic 12	5.60	0.08	7

DynamicAnalysisScore
14.28

Product 2

TopicName	NumberOfVulnerabilityFound
Topic 12	1
Topic 13	2
Topic 00	4
Topic 12	8

TopicName	VulCount
Topic 00	1342
Topic 12	798
Topic 13	6139

TopicName	AvgScore
Topic 00	7.1
Topic 12	5.6
Topic 13	4.49

TopicName	mindate	Column1
Topic 00	1995/08/31	2012/12/17
Topic 12	1997/01/01	2012/12/17
Topic 13	1999/12/31	2012/12/21

TopicName	MonthNum
Topic 00	207
Topic 12	191
Topic 13	155

TopicName	VulCount	MonthNum	FrequencyPerMonth
Topic 00	1342	207	6
Topic 12	798	191	4
Topic 13	6139	155	39

TopicName	Average Impact Score	Frequency Per Month
Topic 00	7.10	6
Topic 12	5.60	4
Topic 13	4.49	39

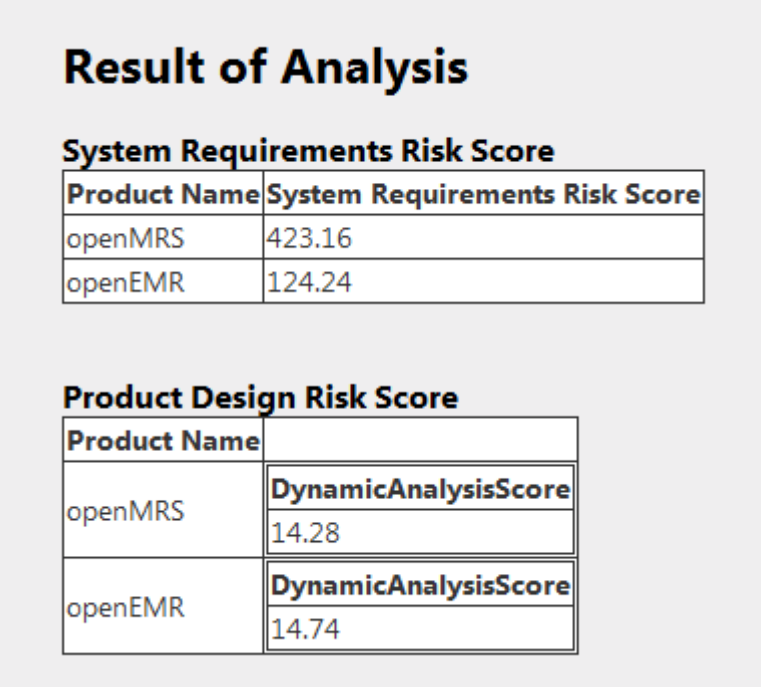
TopicName	Average Impact Score	Probability	Number Of Vulnerability Found
Topic 12	5.60	0.08	1
Topic 13	4.49	0.80	2
Topic 00	7.10	0.12	4
Topic 12	5.60	0.08	8

DynamicAnalysisScore
14.74

Figure 5-16. Product Risk Score Calculation Screen

5.7.7 Final Risk Score Display

The last screen displays the final security risk score identified for the products entered.



Result of Analysis

System Requirements Risk Score

Product Name	System Requirements Risk Score
openMRS	423.16
openEMR	124.24

Product Design Risk Score

Product Name	
openMRS	DynamicAnalysisScore: 14.28
openEMR	DynamicAnalysisScore: 14.74

Figure 5-17. Final Risk Score Display Screen

Figure 5-17 indicates the results of our prototype example. In the screens shown above, in a what-if analysis, the system engineer chooses to install one product (openEMR in example above) inside the firewall and the second product (openMRS), outside the firewall. One has high ‘protection level’ and the second one had low ‘protection level’. From the results one can interpret that system environment chosen for openEMR is approximately 3 times more secure than that chosen for openMRS. In other words, the results demonstrate that the product that is within the firewall and has high protection level will have lower system requirement risk as compared to the second product, which we choose to install outside the firewall.

6 Evaluation and Results

This chapter evaluates the procedure and the experiment, discusses the results, points out the verification of the hypothesis performed using DSRM, and finally, provides an overview of the limitations of this study.

6.1 Quantitative Security Risk Assessment Model Evaluation

Risk assessment is an essential step in any engineering activity that ensures that potential pitfalls have been evaluated in advance and action has been taken. The historical data repository provides a valuable resource for risk identification, prediction, and estimation. We use historical data from national vulnerability database (NVD) to identify known groups of security risk, their impact, and their probability. In this study, we explore the feasibility of 'Topic Modeling' algorithms for creating the quantitative security risk assessment model. The primary advantage of topic modeling algorithm is the automatic identification of latent themes in large collection of text. In addition, the topic model once trained can be utilized to infer topics for other text in the same domain. Most empirical data sources and vulnerability scanners produce output that has textual information describing the vulnerability. Thus categorizing based on text is a useful capability. Topic modeling is a valuable solution for categorizing vulnerability into security risk groups. Our test shows that the resulting groups demonstrate good cohesion when compared with their average impact scores. In addition, the consolidated security metrics for each topic reflects the posture of that risk group in the real world to a great extent. One of the limitations of the process is that we are dependent on the descriptions' text. As a result, if the reported vulnerability does not use common words to describe the vulnerability, it gets assigned to an unrelated group. Due to the same reason,

vulnerabilities related to newer technology do not get assigned to appropriate topics. The topics created by the topic modeling algorithm are not automatically labeled and thus, their interpretation is left to the user. We labeled our topics by feeding the vulnerability scanner category description from different scanners into our trained topic model, which categorized them into the same topics, which were created using NVD data. This provided the additional benefit of being able to scan different software products using those scanners and deriving a security score. The topic terms and their labels could be verified using tag clouds and different string similarity algorithms. In general, the topic and the term probability output from the topic modeling toolbox provided a very good indication of the topic when visualized using 'tag cloud' representation. In any domain, there will be large amount of noise words that are common but do not help to identify the topic. Thus, creating a domain specific stop-word-list is highly recommended to obtain a clean topic-terms list. Topic-terms lists are essential components for visual understanding and labeling of topics. The topics having a large representative in the vulnerability database had the highest percentage of being correctly labeled. As a future work, it may be beneficial to compare this simplistic type of labeling with implementation of automatic labeling of topic model described in existing research (Lau, Grieser, Newman, & Baldwin, 2011). In addition, implementation of hierarchical LDA to automatically learn the number of topics would also be a beneficial addition to this study.

We analyzed the vulnerabilities in those topic groups that had a high impact but low probability. We found that a few of these low probability topic groups were un-labeled. We analyzed the vulnerabilities within these unlabeled low probability, high impact groups and found that some of them belonged to the risk groups like buffer overflow,

input validation, code injection, and OS command injections. The reason we believe these did not get associated with the right topic was due to the summary text description not having the high probability anchor words. Further analysis of the unlabelled groups also revealed that some of these topics were related with newer technology such as 'Android.' On extracting the NVD records with the word 'android,' we found that a majority of the errors in the CVE for 'android' belonged to the topic groups '18' and '19' as shown below in Table 6-1.

Table 6-1. 'Android' term dataset topic analysis

Topic	Label	Low	Medium	High	Total
Topic 00	Numeric Errors	0	2	2	4
Topic 04	Credentials Management	4	3	1	8
Topic 06	Denial Of Service	0	2	0	2
Topic 11	Input Validation	0	1	0	1
Topic 12	Content Spoofing	0	1	1	2
Topic 13	Cross-Site Scripting (XSS)	0	2	0	2
Topic 15	SSL certificates	0	1	0	1
Topic 16	Bad caching directives	0	1	0	1
Topic 17	Buffer Errors	0	0	1	1
Topic 18	Unspecified Topic 18	0	5	67	72
Topic 19	Unknown Topic 19	0	1	44	45
Topic 20	Resource Management Errors	1	1	0	2
Topic 23	Permissions, Privileges, and Access Control	0	1	0	1
Topic 37	Link Following	0	1	0	1
Topic 39	Firefox	0	1	1	2

We found that some topics like topics 18 and 19 were not labeled because the summary text did not have incomplete description of the attack vector and were described with words such as 'unspecified vulnerability …' or 'unknown…'. In other cases, we found that they did apply to the vulnerability scanners we used, which were geared towards web applications.

Another interesting issue emerged through this analysis. Topics that had low

representation in the database showed up as having a low probability of occurrence. In case of newer technology, such as 'android' mobile applications, this did not indicate low probability of occurrence. For example, we found two topic groups that were un-labeled and had a low overall probability. However, when we filtered the data by the word 'android' and re-calculated the probability, we found that they had a reasonably high probability and high impact score as shown as Table 6-2. In addition, on checking the confidentiality, integrity, and availability (CIA) security metrics likelihood levels of these topics, we found that both these topics had the impact level of 'complete' with high probability. Thus, the first lesson learnt was that even though the process worked well for web applications, the model did not support assessment for newer technologies because the summary text was not well-defined. The second lesson learnt was that for optimal decision-making considering just impact and probability of occurrence was not enough. Each security risk group required analysis in terms of multiple dimensions and in the context of the type of component or technology assessed.

Table 6-2. Analysis of newer technology related topics

Topic	Impact Score	Probability of Occurrence
Overall		
Topic 19	7.85	3.12%
Topic 18	8.60	1.17%
For Android alone		
Topic 18	9.31	40.91%
Topic 19	9.75	25.57%

6.2 Procedure Evaluation

"In order to evaluate the artifact, we needed to detect the potential vulnerabilities in the candidate software product. However, this was challenging for multiple reasons. Much effort and skill was required for manual vulnerability detection. Automated

vulnerability testing was faster but suffered from problems related to false positives and missed issues. The two methods of automatic vulnerability detection are static analysis and dynamic analysis. Static analysis was performed by examining the source code or the binaries of the software, but no suitable method was found to map them to the topic groups. Dynamic analysis was performed on the application when it was running, using a vulnerability scanner. The challenge with vulnerability testing was that neither of the methods was comprehensive by itself, as pointed out by Austin et al. (Austin & Williams, 2011). In addition, the output of different scanners produced different results and often missed some vulnerability (Dessiatnikoff, Akrout, Alata, Kaaniche, & Nicomette, 2011). Thus, it was recommended that multiple scanners be used for assessment" (Ruma Das, 2012).

"We selected the method of automated dynamic vulnerability detection for our experiment. As per Austin et al., in case of limited time, "automated penetration testing can be conducted to discover penetration bugs". Our assumption was that we would use the same dynamic analysis tools on both candidate products, causing similar results in both. However, we recognized that the usage of the tool and code-based characteristics of candidate software products had an impact on the results of the static analysis tool ("Software [In]security: Comparing Apples, Oranges, and Aardvarks (or, All Static Analysis Tools Are Not Created Equal) | As the Market Grows | InformIT,").The goal of the research was to evaluate relative security risk level of the products and not the absolute security risk levels. Since we did not find any absolute solution for vulnerability detection, we proceeded with the automated vulnerability detection method. However, we acknowledge that detection of vulnerability is an issue as identified by (Austin &

Williams, 2011) (Doupe, Cova, & Vigna, 2010) and others" (Ruma Das, 2012).

"We examined the topic groups and found that they presented the vulnerability domain very well, thereby, confirming the results presented by Neuhaus et al. (Neuhaus & Zimmermann, 2010) and our hypothesis one. We encountered multiple challenges when implementing the assessment model for software selection procedure. First, we wanted to design the proposed procedure such that the practitioners using the procedure could find it practical and not overly time consuming. In an empirical study of how OSS is selected in small companies, authors (Hauge et al., 2009) find that OSS selection procedure is sequential , with the 'first fit' solution being selected rather than 'best-fit'(Hauge et al., 2009). Thus, we assumed that practitioners review multiple tools in a short duration of time, leaving very little time for extensive security testing. Therefore, we selected automatic black-box testing methods for vulnerability detection since it was time-efficient for detecting implementation flaws (Austin & Williams, 2011). Second, finding a method to effectively create list(s) of vulnerabilities for a candidate software system was difficult due to issues reported in prior research (Austin & Williams, 2011) ("Software [In]security: Comparing Apples, Oranges, and Aardvarks (or, All Static Analysis Tools Are Not Created Equal) | As the Market Grows | InformIT,") (Suto, 2010). Manual testing depended on the skills and knowledge of the tester, including the fact that it was time consuming. Automated vulnerability detection was limited by the vulnerability analysis tools ("Software [In]security: Comparing Apples, Oranges, and Aardvarks (or, All Static Analysis Tools Are Not Created Equal) | As the Market Grows | InformIT,"), a high percentage of false positives, lack of coverage due to inability to scan all pages, etc. We argued that some evaluation was better than none. At least the software

systems would not be selected without any kind of security risk evaluation. Finally, we were faced with the issue of matching the results of different scanners with the quantitative security risk assessment model. The results of static as well as dynamic analyses were in different textual formats. The documentation of different vulnerability scanners contained a long list of individual security problems detected by the scanners. In addition, since one scanner and one methodology would not find all the vulnerabilities, we proposed the use of multiple tools for scanning. This added the complexity of converting the text output of the security scanners to a format that could be used to infer the topic and then, obtain the impact score and frequency for that topic. However, the topic model once trained performed well when categories from different scanners were fed into it. Many of them were categorized to the appropriate category as verified by cross-referencing with CWE and precision-recall test".

The goal of our research was to investigate following questions:

RQ1: What are the essential dimensions for evaluating a candidate software product before selection? (Ruma Das, 2012)

Following 'System Thinking,' we proposed that the security of the product is based on the software product design, its required system environment, and the protection level of the environment. "The security risk level of a potential software product can be evaluated by considering two dimensions: System Requirement Risk Dimension and Product Design Risk dimensions" (Ruma Das, 2012). Each of these has sub dimensions and steps as described in 5.5.2 Dimensions of Risk section. Our final model is shown in Figure 5-3.

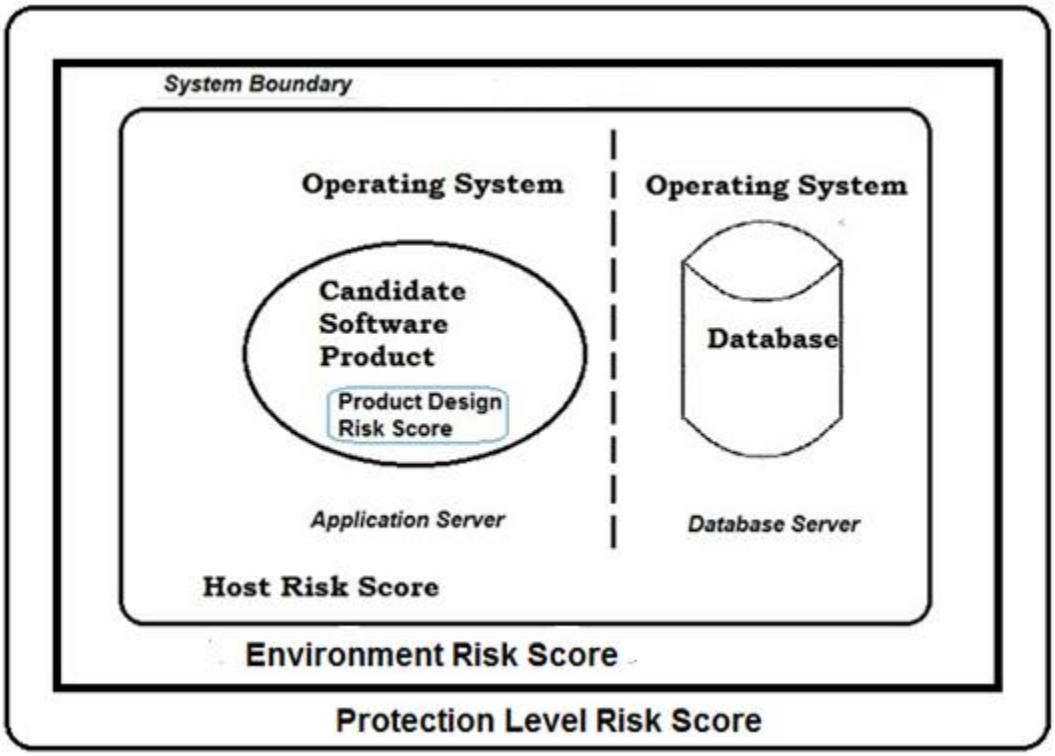


Figure 6-1. RQ1- Dimensions of Risk(Ruma Das 2012)

RQ2: Can we use a quantitative security risk assessment model to compare and select between custom software products? (Ruma Das, 2012)

Prior research has demonstrated metrics and measurement to extract impact score and probability from the NVD dataset. We have adopted these to calculate the system components and host risk score of our software product. "We can build a quantitative security risk assessment model using empirical data about vulnerabilities from NVD, as demonstrated in prior research and in Section 5.5.3 of this paper. The information is available as summary text, and thus, we have used topic modeling algorithm to determine groups of threats and labeled these groups or topics using the categories from the vulnerability scanner" (Ruma Das, 2012). Finally, the likelihood and average impact score for each security risk group has been calculated. For evaluating the latent security

risk due to the design of the product, we have used vulnerability scanner and scanned the product. Then, we have mapped the output of the scanner to our quantitative risk assessment model to get the product design risk score as explained in Section 5.6.2.

RQ3: How do we analyze the security risk level of a potential software product? (Ruma Das, 2012)

The security risk of a potential software product is evaluated based on system requirements and dynamic analysis using vulnerability scanner. We analyze the security risk level of a potential software product using our defined dimensions and steps mentioned in section 5.6.1 which are similar to 800:53 IT risk assessment (Gary Stoneburner, 2002). This has been described in Section 5.6 through an experimental case study.

6.3 Hypothesis Reference

The hypotheses were tested during the course of the steps followed using design science research methodology and therefore, the sections evaluating these have been highlighted below:

Hypothesis 1: Historic vulnerability data can be used discover software security risk groups through mining and clustering textual data

We validated the clusters created by topic modeling using following methods:

Word Cloud Visualization (5.5.4.1)

Cross-comparison with CWE types (5.5.4.6)

Content Validation (5.5.4.8)

The CWE cross-comparison results indicate that the topic security risk groups

discovered by the algorithm identify the common security risk groups like buffer overflow, cross-site scripting, SQL injection, which are part of top 10 security risks for software. The weights of the terms that help to label the topic have higher probability distribution in the summary text of that CVE in the related topic as shown by ‘word cloud visualization’ and ‘content validation’ analysis.

Hypothesis 2: Vulnerability categories from the scanner can be mapped to the discovered software security risk groups

Cross-comparison with CWE types (5.5.4.6)

Mapping categories of 2 different scanners (5.5.4.7)

After a topic model is trained using the LDA algorithm, it can be used to infer topic groups for new dataset. We verified this by using categories from two different scanners and using string similarity algorithm to verify that similar categories are associated with same topic.

Hypothesis 3: Risk groups found through clustering historic data provides good representation of the security risk score

Standard deviation test for impact score (5.5.4.4)

T-Test of the impact score (5.5.4.5)

In order to test this hypothesis, we calculated the average CVSS score for each topic group and evaluated the standard deviation of the CVSS score of members in the group. The results indicate that the topic CVSS score indicate tight cohesion in the topic group. The statistical evidence based on the t-test also indicates that the predicted impact scores are effective presentation of the security risk groups created through topic modeling.

6.4 Limitations

"The first limitation might be due to the use of single data source, NVD. Prior researches have pointed out the fact that "not all vulnerabilities are present in NVD" (Zheng Chen, Yuqing, Yingfei, & Qixu, 2011). However, the model was proposed for web-based applications for which significant amount of data exists in NVD. Second, one of the limitations of the model is that we are dependent on the descriptions text of the reported vulnerability. Thus, the textual data quality affects the model. Third, the assessment model will only report on known vulnerabilities since it is based on historical data. Most users might be more worried about unknown vulnerabilities than known security risks. However, known vulnerabilities still represent a significant percentage of threats to a software system. Scholte et al. conducted an empirical study of vulnerabilities for web-based software systems and found that complexities of these vulnerabilities did not increase over time. Most of these vulnerabilities occurred due to simple reasons, such as lack of input validation (Scholte et al., 2012) and were variations of already known vulnerabilities (Rehman & Mustafa, 2009)" (Ruma Das, 2012).

" The output of the vulnerability scanner can have an impact on the model. In a study comparing the effectiveness of different vulnerability scanners, authors Bau et al. found that different scanners were strong in different categories (Jason, 2010). Also, they found that vulnerabilities detected by scanners in popular categories, such as SQL injection and cross-site scripting were proportional to those found in the real situation. The variations between scanners and real situations existed in finding second order vulnerabilities, such as "Cross-Channel Scripting, Cross-Site Request Forgery, and Malware Presence"(Jason, 2010). We performed a statistical analysis and found that our scores had less than '2.38'

standard deviation. However, it would have been better to obtain scores less than 2 standard deviations" (Ruma Das, 2012).

One of the challenges we encountered is determination of a range or indicator value (Low, Medium, and High) to indicate which product has significantly lower security risk level. In other words, how much of difference should there be between the scores of the two product to indicate of the product one has a lower risk level than product two and how do those differences map to a range such as 'low, medium, high'. Thus one of the limitations of our work is that the numeric values do not have a unit or indicate any range levels (high, low, medium) to signify if the score is good or bad. Even though, they use the same measure (namely the CVSS score from NVD) and the same formula, the numeric value is only useful for quick relative comparison between the security levels of the two software products. As future work, the recommendation to overcome this limitation would be to calibrate the model by obtaining the scores using multiple software products with known security levels and developing a range value(low, medium, high) to serve as unit of comparison.

6.5 Conclusion

"The aim of this research is to propose an artifact to support the evaluation of off-the-shelf or OSS web-based software products before adoption using a quantitative security risk assessment model. We propose the application of the topic model to discover latent groups of vulnerabilities and develop a risk model based on those groups" (Ruma Das, 2012). The results support our hypothesis that topic modeling can be used to identify security groups and map those groups to the vulnerability scanner categories. "This quantitative security risk model combined with dynamic analysis output, serves to

evaluate multiple software products. We acknowledge the limitation of vulnerability scanning tools and its impact on the procedure. However, given the time constraints that practitioners face to evaluate a software system and the lack of better methods for objective analysis, we propose that our model is a support for a quick and preliminary analysis, which can be further supplemented by more intensive analysis using security experts, depending on the context and available time. For future work, we propose investigating other topic modeling algorithms and the possibility of applying automatic labeling to the topic groups. We believe that the addition of such features to the assessment model would assist in the development of a user-friendly tool to assist the practitioners in comparing multiple products. Topics have been manually labeled. Having topic clusters automatically labeled would provide the practitioners an enhanced accuracy and more practical way to judge the score(s) and determine their mitigation strategies" (Ruma Das, 2012).

7 Future work

The purpose of this chapter is to provide suggestions for future work to extend this research. Research is not only a fact finding and problem solving endeavor, it is also a journey of discovery of unknowns and questions for the future. Given below are few suggestions for future work for extending the research:

An area of future work is to evaluate a better method for automatic labeling and merging of the topics. We have not investigated the methods of correlating NVD dataset and static analysis results, to evaluate the security risk score. This is another area of future work. The topic clusters generated by machine might have overlap between them. The automatic merging of similar topic has not been handled in this research. Finally, manual methods were used to correct the mapping between the topic and scanner labels. Better algorithms may be applied to fit the scanner label to the topic groups.

The model itself may be improved by evaluating topic modeling algorithms other than LDA. For example, development of the quantitative risk model using hierarchical latent dirichlet allocation (hLDA) algorithm. hLDA can produce a security group hierarchy, with generic terms at the top level and with specialized topics shown at the bottom of the hierarchy (Hongyan & Lei, 2011). Analysis of the relations between the topics so as to manage overlap between them is a possible extension of this work.

Another improvement to the model can be made by incorporating some algorithm to establish the unit or range (low, medium, high) for the security

scores obtained.

Even though our goal was risk estimation, future work may extend the model to explore of its usage for risk prediction. In our test, the variables were found to be significant, but the variance in the data was not explained well in the linear or multi-step regression test and thus could not be extended for risk prediction.

Comparative studies with other text-mining and clustering algorithm may be another extension of this work, where different vulnerability clustering and categorizing frameworks are evaluated and compared. For example, our research is similar to study by Venter et al. which involved clustering NVD data and mapping to vulnerability scanner categories using self-organizing maps(SOM) by (Venter et al., 2008). One area of extending this research would be to compare both the methods and evaluate their performance.

Empirical data can be utilized to develop dashboards for engineering management. With the potential of automatically detecting latent security groups and labeling them, 'Security Risk Dashboards' may be developed to assist in decision-making and providing insight into emerging and re-emerging risk groups. These groups may be further sliced by product type or year to see the trend and evolution of different risks. Thus, as a decision support tool for engineering management, this research can be extended to build a security risk dashboard by utilizing the quantitative model developed with probabilities from the historic data. Thus, a potential research area might be to develop security risk dimension for such dashboard and explore visualization techniques such network visualization, graphs and concept maps, to create the components of the

dashboard.

Another area of research would be to quantitatively determine the security score using static analysis. In our research we spent the effort on dynamic analysis since it is applicable to both COTS and OSS. However with OSS, there is the benefit of being able to perform static analysis which provides in depth analysis of the code. The method of developing quantitative security score using static analysis output and historic data would greatly benefit this model.

This research supports the risk identification and risk assessment stages of security risk management. Extending this by adding mitigation approaches for each security risk cluster or by connecting the clusters to the security policy dimensions may lead to a useful security risk management tool.

8 Publication history

8.1 Communication

The last step of design science research methodology is ‘communication’. The material in this research has been presented in the following avenues:

Conference Presentations

INCOSE: Ruma Das, Dr. Shahram Sarkani, & Dr. Thomas Mazzuchi, Software System Selection Based on Quantitative Security Risk Assessment Model, presented at 2012 INCOSE Great Lakes Regional Conference, October, 2012

IEEE Computer Society : Das, Ruma; Sarkani, Shahram; Mazzuchi, Thomas A.; , "Fast Abstract: Software Selection Based on Quantitative Security Risk Assessment," High-Assurance Systems Engineering (HASE), 2012 IEEE 14th International Symposium on , vol., no., pp.171-172, 25-27 Oct. 2012 . doi: 10.1109/HASE.2012.10

Publications

Ruma Das, Shahram Sarkani and Thomas A Mazzuchi. Article: Software Selection based on Quantitative Security Risk Assessment. IJCA Special Issue on Computational Intelligence & Information Security CIIS(1):45-56, November 2012. Published by Foundation of Computer Science, New York, USA.

9 References

Aggarwal, A., & Jalote, P. (2006, 17-21 Sept. 2006). *Integrating Static and Dynamic Analysis for Detecting Vulnerabilities.* Paper presented at the Computer Software and Applications Conference, 2006. COMPSAC '06. 30th Annual International.

Ahmed, M. S., Al-shaer, E., Taibah, M., & Khan, L. (2011). Objective Risk Evaluation for Automated Security Management. *Journal of Network and Systems Management, 19*(3), 343-366.

Al-Qutaish, R. E., Muhairat, M. I., & Al-Kasasbeh, B. M. (2009). *The analytical hierarchy process as a tool to select open source software*. Paper presented at the Proceedings of the 8th WSEAS International Conference on Software engineering, parallel and distributed systems, Cambridge, UK.

Alhazmi, O. H., Malaiya, Y. K., & Ray, I. (2007). Measuring, analyzing and predicting security vulnerabilities in software systems. *Computers & Security, 26*(3), 219-228. doi: 10.1016/j.cose.2006.10.002

Alrouh, B., Al-Debei, M. M., & Ghinea, G. (2010). *Developing a decision-making framework for web service security profiles: a design-science paradigm*. Paper presented at the Proceedings of the 1st International Conference on Intelligent Semantic Web-Services and Applications, Amman, Jordan.

Alves, C., & Finkelstein, A. (2002). *Challenges in COTS decision-making: a goal-driven requirements engineering perspective*. Paper presented at the Proceedings of the 14th international conference on Software engineering and knowledge engineering, Ischia, Italy.

Anthes, G. (2010). Topic Models Vs. Unstructured Data. [Article]. *Communications of the ACM, 53*(12), 16-18. doi: 10.1145/1859204.1859210

Arnold, A. D., Hyla, B. M., & Rowe, N. C. (2006, 21-23 June 2006). *Automatically Building an Information-Security Vulnerability Database.* Paper presented at the Information Assurance Workshop, 2006 IEEE.

Asuncion, H. U., Asuncion, A. U., & Taylor, R. N. (2010, 2-8 May 2010). *Software traceability with topic modeling.* Paper presented at the Software Engineering, 2010 ACM/IEEE 32nd International Conference on.

Austin, A., & Williams, L. (2011). *One Technique is Not Enough: A Comparison of Vulnerability Discovery Techniques*. Paper presented at the Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement.

Bau, J., Bursztein, E., Gupta, D., & Mitchell, J. (2010, 16-19 May 2010). *State of the Art: Automated Black-Box Web Application Vulnerability Testing.* Paper presented at the Security and Privacy (SP), 2010 IEEE Symposium on.

Bayuk, J., & Mostashari, A. (2012). Measuring systems security. *Systems Engineering*, n/a-n/a. doi: 10.1002/sys.21211

Bayuk, J. L. (2011). Systems Security Engineering. *Security & Privacy, IEEE, 9*(2), 72-74. doi: 10.1109/msp.2011.41

Bayuk, J. L., & Horowitz, B. M. (2011). An architectural systems engineering methodology for addressing cyber security. *Systems Engineering, 14*(3), 294-304. doi: 10.1002/sys.20182

Blei, D. M. (2012). Probabilistic topic models. *Commun. ACM, 55*(4), 77-84. doi:

10.1145/2133806.2133826

Breier, J., & Hudec, L. (2011). *Risk analysis supported by information security metrics*. Paper presented at the Proceedings of the 12th International Conference on Computer Systems and Technologies, Vienna, Austria.

Brian, C. (2005). *Metrics that matter: quantifying software security risk*. Paper presented at the Workshop on Software Security Assurance Tools, Techniques and Metrics.

. Bugzilla. from http://www.bugzilla.org

Chen, Z., Yuqing, Z., Yingfei, S., & Qixu, L. (2011, 1-2 June 2011). *IVDA: International Vulnerability Database Alliance.* Paper presented at the Cybersecurity Summit (WCS), 2011 Second Worldwide.

Chen, Z., Zhang, Y., & Chen, Z. (2010). A Categorization Framework for Common Computer Vulnerabilities and Exposures. *The Computer Journal, 53*(5), 551-580. doi: 10.1093/comjnl/bxp040

Chenmeng Sui, Y. L., Yun Liu. (2012). A Software Security Assessment System Based On Analysis of Vulnerabilities. *Journal of Convergence Information Technology, 7*(6), 211 ~ 219.

Childs, D. (2003, March 8-15, 2003). *Information technology security system engineering methodology*. Paper presented at the Aerospace Conference, 2003. Proceedings. 2003 IEEE.

Chowdhury, I., & Zulkernine, M. (2011). Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities. *Journal of Systems Architecture, 57*(3), 294-

313. doi: 10.1016/j.sysarc.2010.06.003

Chung, Y. J., Kim, I., Lee, N., Lee, T., & In, H. P. (2005). *Security risk vector for quantitative asset assessment*. Paper presented at the Proceedings of the 2005 international conference on Computational Science and Its Applications - Volume Part II, Singapore.

Das, R., Sarkani, S., & Mazzuchi, T. A. (2012, 25-27 Oct. 2012). *Fast Abstract: Software Selection Based on Quantitative Security Risk Assessment.* Paper presented at the High-Assurance Systems Engineering (HASE), 2012 IEEE 14th International Symposium on.

Dessiatnikoff, A., Akrout, R., Alata, E., Kaaniche, M., & Nicomette, V. (2011, 12-14 Dec. 2011). *A Clustering Approach for Web Vulnerabilities Detection.* Paper presented at the Dependable Computing (PRDC), 2011 IEEE 17th Pacific Rim International Symposium on.

Doupe, A., Cova, M., & Vigna, G. (2010). *Why Johnny can't pentest: an analysis of black-box web vulnerability scanners*. Paper presented at the Proceedings of the 7th international conference on Detection of intrusions and malware, and vulnerability assessment, Bonn, Germany.

Dwaikat, Z., & Parisi-Presicce, F. (2005). *Risky trust: risk-based analysis of software systems*. Paper presented at the Proceedings of the 2005 workshop on Software engineering for secure systems\—building trustworthy applications, St. Louis, Missouri.

Fonseca, J., Vieira, M., & Madeira, H. (2007, 17-19 Dec. 2007). *Testing and*

Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks. Paper presented at the Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on.

Frei, S., May, M., Fiedler, U., & Plattner, B. (2006). *Large-scale vulnerability analysis*. Paper presented at the Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense, Pisa, Italy.

Fruhwirth, C., & Mannisto, T. (2009). *Improving CVSS-based vulnerability prioritization and response with context information*. Paper presented at the Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement.

Gary Stoneburner, A. G., and Alexis Feringa. (2002). Risk Management Guide for Information Technology Systems. *NIST-SP 800:30*. Retrieved from

Groven, A.-K., Haaland, K., Glott, R., & Tannenberg, A. (2010). *Security measurements within the framework of quality assessment models for free/libre open source software*. Paper presented at the Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, Copenhagen, Denmark.

Grunske, L., & Joyce, D. (2008). Quantitative risk-based security prediction for component-based systems with explicitly modeled attack profiles. *Journal of Systems and Software, 81*(8), 1327-1345. doi: 10.1016/j.jss.2007.11.716

Gu, Y.-h., & Li, P. (2010, 4-6 June 2010). *Design and Research on Vulnerability Database.* Paper presented at the Information and Computing (ICIC), 2010 Third International Conference on.

Hall, D. C. (2011). Making risk assessments more comparable and repeatable. *Systems Engineering, 14*(2), 173-179. doi: 10.1002/sys.20169

Hauge, O., Osterlie, T., Sorensen, C.-F., & Gerea, M. (2009). *An empirical study on selection of Open Source Software - Preliminary results*. Paper presented at the Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Q., 28*(1), 75-105.

Hindle, A., Ernst, N. A., Godfrey, M. W., & Mylopoulos, J. (2011). *Automated topic naming to support cross-project analysis of software maintenance activities*. Paper presented at the Proceedings of the 8th Working Conference on Mining Software Repositories, Waikiki, Honolulu, HI, USA.

Hongyan, L., & Lei, L. (2011, 27-29 Nov. 2011). *Multi-document summarization based on hierarchical topic model.* Paper presented at the Natural Language Processing andKnowledge Engineering (NLP-KE), 2011 7th International Conference on.

Houmb, S. H., Franqueira, V. N. L., & Engum, E. A. (2010). Quantifying security risk level from CVSS estimates of frequency and impact. *Journal of Systems and Software, 83*(9), 1622-1634. doi: http://dx.doi.org/10.1016/j.jss.2009.08.023

Ibidapo, A. O., Zavarsky, P., Lindskog, D., & Ruhl, R. (2011, 9-11 Oct. 2011). *An Analysis of CVSS v2 Environmental Scoring.* Paper presented at the Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom).

INCOSE. (2003). INCOSE Systems Engineering Handbook. *3.2*.

Jadhav, A., & Sonar, R. (2009, 16-18 Dec. 2009). Analytic Hierarchy Process (AHP), Weighted Scoring Method (WSM), and Hybrid Knowledge Based System (HKBS) for Software Selection: A Comparative Study. Paper presented at the Emerging Trends in Engineering and Technology (ICETET), 2009 2nd International Conference on.

Jadhav, A. S., & Sonar, R. M. (2011). Framework for evaluation and selection of the software packages: A hybrid knowledge based system approach. *Journal of Systems and Software, 84*(8), 1394-1407. doi: 10.1016/j.jss.2011.03.034

Jason, B. (2010). State of the Art: Automated Black-Box Web Application Vulnerability Testing.

Joh, H. (2011). *Quantitative analyses of software vulnerabilities.* Ph.D. 3489881, Colorado State University, United States -- Colorado. Retrieved from http://proxygw.wrlc.org ProQuest Dissertations & Theses (PQDT) database.

Ju An, W., Hao, W., Minzhe, G., Linfeng, Z., & Camargo, J. (2010, 5-8 Jan. 2010). *Ranking Attacks Based on Vulnerability Analysis.* Paper presented at the System Sciences (HICSS), 2010 43rd Hawaii International Conference on.

Ju An, W., Linfeng, Z., Minzhe, G., Hao, W., & Camargo, J. (2010, 5-8 Jan. 2010). *Measuring Similarity for Security Vulnerabilities.* Paper presented at the System Sciences (HICSS), 2010 43rd Hawaii International Conference on.

Juhani Iivari , J. V. (2009). Action Research and Design Science Research – Seemingly similar but decisively dissimilar. *17th European Conference on Information Systems*.

Karabacak, B., & Sogukpinar, I. (2005). ISRAM: information security risk analysis method. *Computers & Security, 24*(2), 147-159. doi: 10.1016/j.cose.2004.07.004

Kennedy, M. (2010). Evaluating Open Source Software. *Defense AT&L*. Retrieved from http://www.dau.mil/pubscats/ATL%20Docs/Jul-Aug10/Kennedy_jul-aug10.pdf

Khan, R. (2011). Secure software development: a prescriptive framework. *Computer Fraud & Security, 2011*(8), 12-20. doi: 10.1016/s1361-3723(11)70083-5

Krivograd, N., & Fettke, P. (2012, 4-7 Jan. 2012). *Development of a Generic Tool for the Application of Maturity Models--Results from a Design Science Approach.* Paper presented at the System Science (HICSS), 2012 45th Hawaii International Conference on.

Kuan-Yu, C., Hsuan-Sheng, C., & Chen, B. (2010, 14-19 March 2010). *Latent topic modeling of word vicinity information for speech recognition.* Paper presented at the Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on.

Kujawski, E., & Angelis, D. (2010). Monitoring risk response actions for effective project risk management. *Systems Engineering, 13*(4), 353-368. doi: 10.1002/sys.20154

Lafferty, D. B. a. J. (2009). Topic Models. *A. Srivastava and M. Sahami, editors, Text Mining: Classification, Clustering, and Applications* . . Retrieved from http://www.cs.princeton.edu/~blei/papers/BleiLafferty2009.pdf

Lau, J. H., Grieser, K., Newman, D., & Baldwin, T. (2011). *Automatic labelling of topic models*. Paper presented at the Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, Portland, Oregon.

Lavazza, L., Morasca, S., Taibi, D., & Tosi, D. (2012). *An empirical investigation of perceived reliability of open source Java programs*. Paper presented at the Proceedings of the 27th Annual ACM Symposium on Applied Computing, Trento, Italy.

Lee, Y. M., Kim, J. B., Choi, I. W., & Rhew, S. Y. (2007, 22-24 Aug. 2007). *A Study on Selection Process of Open Source Software.* Paper presented at the Advanced Language Processing and Web Information Technology, 2007. ALPIT 2007. Sixth International Conference on.

Leidner, J. L., & Schilder, F. (2010). *Hunting for the black swan: risk mining from text*. Paper presented at the Proceedings of the ACL 2010 System Demonstrations, Uppsala, Sweden.

Linstead, E., Rigor, P., Bajracharya, S., Lopes, C., & Baldi, P. (2007). *Mining concepts from code with probabilistic topic models*. Paper presented at the Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, Atlanta, Georgia, USA.

Liu, C., Li, J., & Chen, X. (2012). *Network vulnerability analysis using text mining*. Paper presented at the Proceedings of the 4th Asian conference on Intelligent Information and Database Systems - Volume Part II, Kaohsiung, Taiwan.

Liu, Q., & Zhang, Y. (2011). VRSS: A new system for rating and scoring vulnerabilities. *Comput. Commun., 34*(3), 264-273. doi: 10.1016/j.comcom.2010.04.006

Lukins, S. K., Kraft, N. A., & Etzkorn, L. H. (2010). Bug localization using latent Dirichlet allocation. *Information and Software Technology*, *52*(9), 972-990. doi: 10.1016/j.infsof.2010.04.002

Magatti, D., Calegari, S., Ciucci, D., & Stella, F. (2009, Nov. 30 2009-Dec. 2 2009). *Automatic Labeling of Topics.* Paper presented at the Intelligent Systems Design and Applications, 2009. ISDA '09. Ninth International Conference on.

Massacci, F., & Nguyen, V. H. (2010). *Which is the right source for vulnerability studies?: an empirical analysis on Mozilla Firefox*. Paper presented at the Proceedings of the 6th International Workshop on Security Measurements and Metrics, Bolzano, Italy.

McCallum, A. K. (2002). "MALLET: A Machine Learning for Language Toolkit.", from http://mallet.cs.umass.edu/

Mendes, N., Duraes, J., & Madeira, H. (2011, 25-29 April 2011). *Benchmarking the Security of Web Serving Systems Based on Known Vulnerabilities.* Paper presented at the Dependable Computing (LADC), 2011 5th Latin-American Symposium on.

Michal Zalewski, N. H., Sebastian Roschke. (2012). Skipfish-web application security scanner Retrieved 05/05/2012, 2012, from http://code.google.com/p/skipfish/wiki/SkipfishDoc

Microsoft. (2012). Microsoft Threat Analysis & Modeling Retrieved 5/3/2012, 2012, from http://www.microsoft.com/en-us/download/details.aspx?id=14719

MITRE. Common Weakness Enumeration

MITRE. (2011a). CVE -Common Vulnerabilities and Exposures (CVE) Retrieved 06-01-2012, 2012, from http://cve.mitre.org/

MITRE. (2011b). CWE -Common Weakness Enumeration Retrieved 6/24/2012, 2012, from http://cwe.mitre.org/compatible/category.html

MITRE. (2012). MITRE | Systems Engineering Guide | Acquisition Systems

Engineering | Risk Management | Risk Impact Assessment and Prioritization. *MITRE--Applying Systems Engineering and Advanced Technology to Critical National Problems* Retrieved 2/7/2012, 2012, from http://www.mitre.org/work/systems_engineering/guide/acquisition_systems_engineering/risk_management/risk_impact_assessment_prioritization.html

Mkpong-Ruffin, I. (2009). *Quantitative risk assessment model for software security in the design phase of software development.* Ph.D. 3348283, Auburn University, United States -- Alabama. ProQuest Dissertations & Theses (PQDT) database.

Mkpong-Ruffin, I., Umphress, D., Hamilton, J., & Gilbert, J. (2007). *Quantitative software security risk assessment model*. Paper presented at the Proceedings of the 2007 ACM workshop on Quality of protection, Alexandria, Virginia, USA.

Morasca, S., Taibi, D., & Tosi, D. (2009). *Towards certifying the testing process of Open-Source Software: New challenges or old methodologies?* Paper presented at the Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development.

Negoescu, R. A., & Gatica-Perez, D. (2010). Modeling Flickr Communities Through Probabilistic Topic-Based Analysis. *Multimedia, IEEE Transactions on, 12*(5), 399-416. doi: 10.1109/tmm.2010.2050649

Neuhaus, S., & Zimmermann, T. (2010, 1-4 Nov. 2010). *Security Trend Analysis with CVE Topic Models.* Paper presented at the Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on.

Nguyen, V. H., & Massacci, F. (2012). *An idea of an independent validation of*

vulnerability discovery models. Paper presented at the Proceedings of the 4th international conference on Engineering Secure Software and Systems, Eindhoven, The Netherlands.

NVD. (2012). National vulnerability database Retrieved 11/1/2011, 2011, from http://nvd.nist.gov

Ouedraogo, M., Khadraoui, D., Mouratidis, H., & Dubois, E. (2012). Appraisal and reporting of security assurance at operational systems level. *Journal of Systems and Software, 85*(1), 193-208. doi: 10.1016/j.jss.2011.08.013

Ozment, A. (2007). *Improving vulnerability discovery models*. Paper presented at the Proceedings of the 2007 ACM workshop on Quality of protection, Alexandria, Virginia, USA.

Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2008). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems, 24*(3), 45-77. doi: citeulike-article-id:4342024

Peffers, K. E. N., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. [Article]. *Journal of Management Information Systems, 24*(3), 45-77.

Peter Mell, K. S., Sasha Romanosky. (2012). A Complete Guide to the Common Vulnerability Scoring System Version 2.0, from http://www.first.org/cvss/cvss-guide.html

Polancic, G., Horvat, R. V., & Rozman, T. (2004, 7-10 June 2004). *Comparative assessment of open source software using easy accessible data.* Paper presented at the

Information Technology Interfaces, 2004. 26th International Conference on.

Probabilistic Topic Models. (2012). [Article]. *Communications of the ACM, 55*(4), 77-84. doi: 10.1145/2133806.2133826

Proofpoint - Enterprise Email Security, E. A., and Privacy Protection Solutions. (2012). Denial of service attacks increasing, causing market growth Retrieved 12-20-2012, 2012, from http://www.proofpoint.com/about-us/security-compliance-and-cloud-news/articles/denial-of-service-attacks-increasing-causing-market-growth--800927878

Ramirez, E. H., Brena, R., Magatti, D., & Stella, F. (2012). Topic model validation. *Neurocomput., 76*(1), 125-133. doi: 10.1016/j.neucom.2011.04.032

Rehman, S., & Mustafa, K. (2009). Research on software design level security vulnerabilities. *SIGSOFT Softw. Eng. Notes, 34*(6), 1-5. doi: 10.1145/1640162.1640171

Ruma Das, S. S. a. T. A. M. (2012). Software Selection based on Quantitative Security Risk Assessment. *IJCA Special Issue on Computational Intelligence & Information Security, CIIS*(1), 45-56.

Ruma Das , S. S. a. T. A. M. (2012). *System Selection Based on Quantitative Security Risk Assessment Model.* Paper presented at the INCOSE Great Lakes Regional Conference, Chicago.

Ruoyu, Z., Shiqiu, H., Zhengwei, Q., & Haibin, G. (2011, June 30 2011-July 2 2011). *Combining Static and Dynamic Analysis to Discover Software Vulnerabilities.* Paper presented at the Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on.

Ryan, J. J. C. H., Mazzuchi, T. A., Ryan, D. J., Lopez de la Cruz, J., & Cooke, R.

(2012). Quantifying information security risks using expert judgment elicitation. *Computers & Operations Research, 39*(4), 774-784. doi: 10.1016/j.cor.2010.11.013

Sage, A. P., & Rouse, W. B. (2009). *Handbook of System Engineering*: Wiley.

Salmela, L., Tarhio, J., & Kytjoki, J. (2007). Multipattern string matching with q-grams. *J. Exp. Algorithmics, 11*, 1.1. doi: 10.1145/1187436.1187438

Salter, C. P., & Buede, D. M. (2001). A lifecycle-based method for the acquisition of commercial-off-the-shelf (COTS) technology to support organizational processes. *Systems Engineering, 4*(4), 287-304. doi: 10.1002/sys.10000

Savola, R. (2010). On the Feasibility of Utilizing Security Metrics in Software-Intensive Systems. *International Journal of Computer Science and Network, 10*(1).

Schneidewind, N. F. (1998, 16-20 Nov 1998). *Methods for assessing COTS reliability, maintainability, and availability.* Paper presented at the Software Maintenance, 1998. Proceedings. International Conference on.

Scholte, T., Balzarotti, D., & Kirda, E. (2012). Have things changed now? An empirical study on input validation vulnerabilities in web applications. *Computers & Security, 31*(3), 344-356. doi: 10.1016/j.cose.2011.12.013

Schryen, G., & Kadura, R. (2009). *Open source vs. closed source software: towards measuring security*. Paper presented at the Proceedings of the 2009 ACM symposium on Applied Computing, Honolulu, Hawaii.

Scott A. Hissam, D. C., Daniel Plakosh. (1998). DoD Security Needs and COTS-Based Systems. Retrieved from

Shi, D. (2004). A review of enterprise supply chain risk management. *Journal of*

Systems Science and Systems Engineering, 13(2), 219-244. doi: 10.1007/s11518-006-0162-2

Shuguang, H., Heping, T., Min, Z., & Jie, T. (2010, 19-21 March 2010). *Text Clustering on National Vulnerability Database.* Paper presented at the Computer Engineering and Applications (ICCEA), 2010 Second International Conference on.

. Software [In]security: Comparing Apples, Oranges, and Aardvarks (or, All Static Analysis Tools Are Not Created Equal) | As the Market Grows | InformIT. *InformIT: The Trusted Technology Source for IT Pros and Developers* Retrieved 05/05/2012, 2012, from http://www.informit.com/articles/article.aspx?p=1680863

Somasundaram, K., & Murphy, G. C. (2012). *Automatic categorization of bug reports using latent Dirichlet allocation*. Paper presented at the Proceedings of the 5th India Software Engineering Conference, Kanpur, India.

Spinellis, D., Gousios, G., Karakoidas, V., Louridas, P., Adams, P. J., Samoladas, I., & Stamelos, I. (2009). Evaluating the Quality of Open Source Software. *Electronic Notes in Theoretical Computer Science, 233*(0), 5-28. doi: 10.1016/j.entcs.2009.02.058

Stanford. (2012). Stanford Topic Modeling Toolbox. *Stanford Topic Modeling Toolbox* Retrieved 5/5/2012, 2012, from http://nlp.stanford.edu/software/tmt/tmt-0.4/

Stanley Kaplan, B. J. G. (1981). On the Quantitative Definition of Risk.

Sung, W. J., Kim, J. H., & Rhew, S. Y. (2007, 22-24 Aug. 2007). *A Quality Model for Open Source Software Selection.* Paper presented at the Advanced Language Processing and Web Information Technology, 2007. ALPIT 2007. Sixth International Conference on.

Suto, L. (2010). *Analyzing the Accuracy and Time Costs of Web Application Security Scanners*. Retrieved from http://www.ntobjectives.com/files/Accuracy_and_Time_Costs_of_Web_App_Scanners.pdf

Tarawneh, F., Baharom, F., Yahaya, J. H., & Zainol, A. (2011, 17-19 July 2011). *COTS software evaluation and selection: A pilot study based in Jordan firms.* Paper presented at the Electrical Engineering and Informatics (ICEEI), 2011 International Conference on.

Teng, L., Lin, L., Yijun, Y., & Zhi, J. (2009, Aug. 31 2009-Sept. 4 2009). AVT Vector: A Quantitative Security Requirements Evaluation Approach Based on Assets, Vulnerabilities and Trustworthiness of Environment. Paper presented at the Requirements Engineering Conference, 2009. RE '09. 17th IEEE International.

Thomas, S. W., Adams, B., Hassan, A. E., & Blostein, D. (2010, 12-13 Sept. 2010). *Validating the Use of Topic Models for Software Evolution.* Paper presented at the Source Code Analysis and Manipulation (SCAM), 2010 10th IEEE Working Conference on.

Thomas, S. W., Adams, B., Hassan, A. E., & Blostein, D. (2012). Studying software evolution using topic models. *Science of Computer Programming*(0). doi: 10.1016/j.scico.2012.08.003

Tripathi, A., & Singh, U. K. (2011, Nov. 29 2011-Dec. 1 2011). *Taxonomic analysis of classification schemes in vulnerability databases.* Paper presented at the Computer Sciences and Convergence Information Technology (ICCIT), 2011 6th International Conference on.

Vache, G. (2009). *Vulnerability analysis for a quantitative security evaluation*. Paper presented at the Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement.

Vanek, F., Jackson, P., & Grzybowski, R. (2008). Systems engineering metrics and applications in product development: A critical literature review and agenda for further research. *Systems Engineering, 11*(2), 107-124. doi: 10.1002/sys.20089

Venter, H. S., Eloff, J. H. P., & Li, Y. L. (2008). Standardising vulnerability categories. *Computers & Security, 27*(3–4), 71-83. doi: 10.1016/j.cose.2008.04.002

Verendel, V. (2009). *Quantified security is a weak hypothesis: a critical survey of results and assumptions*. Paper presented at the Proceedings of the 2009 workshop on New security paradigms workshop, Oxford, United Kingdom.

Wagner, S. (2011). *The use of application scanners in software product quality assessment*. Paper presented at the Proceedings of the 8th international workshop on Software quality, Szeged, Hungary.

Wang, J. A., Guo, M., Wang, H., Xia, M., & Zhou, L. (2009). *Ontology-based security assessment for software products*. Paper presented at the Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies, Oak Ridge, Tennessee.

Wang, J. A., Wang, H., Guo, M., & Xia, M. (2009). *Security metrics for software systems*. Paper presented at the Proceedings of the 47th Annual Southeast Regional Conference, Clemson, South Carolina.

Weber, S., Beck, R., Wolf, M., & Vykoukal, J. (2010, 5-8 Jan. 2010). Portfolio

Performance Measurement Based on Service-Oriented Grid Computing: Developing a Prototype from a Design Science Perspective. Paper presented at the System Sciences (HICSS), 2010 43rd Hawaii International Conference on.

Wheeler, D. Retrieved 10/22/2012, 2012, from http://ftp.osuosl.org/pub/osl/goscon/2009/presentations-compiled/Mngmt%203%20-%20Procurement%20&%20Acquisition%20of%20OSS.pdf

Woo, S.-W., Joh, H., Alhazmi, O. H., & Malaiya, Y. K. (2011). Modeling vulnerability discovery process in Apache and IIS HTTP servers. *Computers & Security*, *30*(1), 50-62. doi: 10.1016/j.cose.2010.10.007

Xin, C., Xiaohua, H., Xiajiong, S., & Rosen, G. (2010, 18-21 Dec. 2010). *Probabilistic topic modeling for genomic data interpretation.* Paper presented at the Bioinformatics and Biomedicine (BIBM), 2010 IEEE International Conference on.

Youn, H., Park, C., & Lee, E. (2011). *Security based survivability risk analysis with extended HQPN*. Paper presented at the Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication, Seoul, Korea.

Yue, W. T., Çakanyıldırım, M., Ryu, Y. U., & Liu, D. (2007). Network externalities, layered protection and IT security risk management. *Decision Support Systems*, *44*(1), 1-16. doi: http://dx.doi.org/10.1016/j.dss.2006.08.009

Yung-Yu, C., Zavarsky, P., Ruhl, R., & Lindskog, D. (2011, 9-11 Oct. 2011). *Trend Analysis of the CVE for Software Vulnerability Management.* Paper presented at the Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom).

Zhihu, W., & Xin, W. (2010, 20-22 Aug. 2010). *Research on technologies in quantitative risk assessment and forcast of network security.* Paper presented at the Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on.

11 Appendix

11.1 Appendix A: Scanner Categories

Table 11-1. Skipfish Scanner Categories Used

ID	Scanner category
1	Server-side SQL injection (including blind vectors, numerical parameters).
2	Explicit SQL-like syntax in GET or POST parameters.
3	Server-side shell command injection (including blind vectors).
4	Server-side XML / XPath injection (including blind vectors).
5	Format string vulnerabilities.
6	Integer overflow vulnerabilities.
7	Stored and reflected XSS vectors in document body (minimal JS XSS support present).
8	Stored and reflected XSS vectors via HTTP redirects.
9	Stored and reflected XSS vectors via HTTP header splitting.
10	Directory traversal (including constrained vectors).
11	Assorted file POIs (server-side sources, configs, etc).
12	Attacker-supplied script and CSS inclusion vectors (stored and reflected).
13	External untrusted script and CSS inclusion vectors.
14	Mixed content problems on script and CSS resources (optional).
15	Incorrect or missing MIME types on renderables.
16	Generic MIME types on renderables.
17	Incorrect or missing charsets on renderables.
18	Conflicting MIME / charset info on renderables.
19	Bad caching directives on cookie setting responses.
20	Directory listing bypass vectors.
21	Redirection to attacker-supplied URLs (stored and reflected).
22	Attacker-supplied embedded content (stored and reflected).
23	External untrusted embedded content.
24	Mixed content on non-scriptable subresources (optional).
25	HTTP credentials in URLs.
26	Expired or not-yet-valid SSL certificates.
27	HTML forms with no XSRF protection.
28	Self-signed SSL certificates.
29	SSL certificate host name mismatches.
30	Bad caching directives on less sensitive content

Table 11-2. AppScan Scanner Categories used

ID	Scanner category
1	Brute Force
2	Insufficient Authentication
3	Credential/Session Prediction
4	Insufficient Authorization
5	Insufficient Session Expiration
6	Session Fixation
7	Content Spoofing
8	Cross-site Scripting
9	Buffer Overflow
10	Format String Attack
11	LDAP Injection
12	OS Commanding
13	SQL Injection
14	SSI Injection
15	XPath Injection
16	Directory Indexing
17	Information Leakage
18	Path Traversal
19	Predictable Resource Location
20	Abuse of Functionality
21	Denial of Service

Run each of the scanner vulnerability category lists through topic model trained using NVD data. Load the results into MS SQL server. Find the topic which represents the reach of category with maximum probability and store in the database.

11.2 Appendix B: CWE used in NVD

Table 11-3. CWE used in NVD

CWE ID	Name	Description
CWE-287	Authentication Issues	Failure to properly authenticate users.
CWE-255	Credentials Management	Failure to properly create, store, transmit, or protect passwords and other credentials.
CWE-264	Permissions, Privileges, and Access Control	Failure to enforce permissions or other access restrictions for resources, or a privilege

CWE ID	Name	Description
		management problem.
CWE-119	Buffer Errors	Buffer overflows and other buffer boundary errors in which a program attempts to put more data in a buffer than the buffer can hold, or when a program attempts to put data in a memory area outside of the boundaries of the buffer.
CWE-352	Cross-Site Request Forgery (CSRF)	Failure to verify that the sender of a web request actually intended to do so. CSRF attacks can be launched by sending a formatted request to a victim, then tricking the victim into loading the request (often automatically), which makes it appear that the request came from the victim. CSRF is often associated with XSS, but it is a distinct issue.
CWE-79	Cross-Site Scripting (XSS)	Failures of a site to validate, filter, or encode user input before returning it to another user's web client.
CWE-310	Cryptographic Issues	An insecure algorithm or the inappropriate use of one; an incorrect implementation of an algorithm that reduces security; the lack of encryption (plaintext); also, weak key or certificate management, key disclosure, random number generator problems.
CWE-22	Path Traversal	When user-supplied input can contain .. or similar characters that are passed through to file access APIs, causing access to files outside of an intended subdirectory.
CWE-94	Code Injection	Causing a system to read an attacker-controlled file and execute arbitrary code within that file. Includes PHP remote file inclusion, uploading of files with executable extensions, insertion of code into executable files, and others.
CWE-134	Format String Vulnerability	The use of attacker-controlled input as the format string parameter in certain functions.
CWE-16	Configuration	A general configuration problem that is not associated with passwords or permissions.
CWE-200	Information Leak / Disclosure	Exposure of system information, sensitive or private information, fingerprinting, etc.
CWE-20	Input Validation	Failure to ensure that input contains well-formed, valid data that conforms to the applications specifications. Note: this overlaps other categories like XSS, Numeric Errors, and SQL Injection.
CWE-189	Numeric Errors	Integer overflow, signedness, truncation, underflow, and other errors that can occur when handling numbers.

CWE ID	Name	Description
CWE-78	OS Command Injections	Allowing user-controlled input to be injected into command lines that are created to invoke other programs, using system() or similar functions.
CWE-362	Race Conditions	The state of a resource can change between the time the resource is checked to when it is accessed.
CWE-399	Resource Management Errors	The software allows attackers to consume excess resources, such as memory exhaustion from memory leaks, CPU consumption from infinite loops, disk space consumption, etc.
CWE-89	SQL Injection	When user input can be embedded into SQL statements without proper filtering or quoting, leading to modification of query logic or execution of SQL commands.
CWE-59	Link Following	Failure to protect against the use of symbolic or hard links that can point to files that are not intended to be accessed by the application.

11.3 Appendix C: String Match using Q-Gram

Table 11-4. Topic Top Terms & Scanner category String Match using Q-Gram

Topic	Risk	Top Terms	scanner name	Q-gram Score
Topic 00	Integer overflow vulnerabilities.	integ , function , valu , craft , overflow , imag , code , trigger , context , depend , larg , memori , heap , length , error , size , earlier , relat , data , arrai , lead , multipl , neg , buffer , librari , font , possibl , product , read , decod , pdf , png , alloc , number , signed , gif , jpeg , contain , certain , field	SkipFish	0.08
Topic 01	Server-side shell command injection (including blind vectors).	cgi , command , build , shell , metacharact , program , vmware , host , earlier , server , script , workstat , virtual , argument , bugzilla , guest , filenam , version , player , tool , process , setup , configur , esx , hotfix , net , cmd , function , ecom , possibl , bin , packag , solut , exec , user , includ , samba ,	SkipFish	0.12

Topic	Risk	Top Terms	scanner name	Q-gram Score
		awstat , read , esxi		
Topic 01	OS Command Injections	cgi , command , build , shell , metacharact , program , vmware , host , earlier , server , script , workstat , virtual , argument , bugzilla , guest , filenam , version , player , tool , process , setup , configur , esx , hotfix , net , cmd , function , ecom , possibl , bin , packag , solut , exec , user , includ , samba , awstat , read , esxi	None-CWE	0.06
Topic 02	Format String Vulnerability	string , function , format , charact , code , variabl , specifi , earlier , demonstr , queri , contain , possibl , certain , encod , sequenc , mysql , argument , input , log , express , tabl , command , filter , escap , relat , filenam , data , version , insert , regular , statement , utf , perl , termin , eval , quot , modifi , unicod , inject , doubl	None-CWE	0.06
Topic 02	Format string vulnerabilities.	string , function , format , charact , code , variabl , specifi , earlier , demonstr , queri , contain , possibl , certain , encod , sequenc , mysql , argument , input , log , express , tabl , command , filter , escap , relat , filenam , data , version , insert , regular , statement , utf , perl , termin , eval , quot , modifi , unicod , inject , doubl	SkipFish	0.06
Topic 02	Format String Attack	string , function , format , charact , code , variabl , specifi , earlier , demonstr , queri , contain , possibl , certain , encod , sequenc , mysql , argument , input , log , express , tabl , command , filter , escap , relat , filenam , data , version , insert , regular , statement , utf , perl , termin , eval , quot , modifi , unicod , inject , doubl	AppScan	0.06
Topic 04	Credentials Management	inform , obtain , sensit , password , store , access , web , control , root , databas , request , direct , insuffici , read , cleartext , download , encrypt , configur , data , credenti , plaintext , mdb , potenti ,	None-CWE	0.06

Topic	Risk	Top Terms	scanner name	Q-gram Score
		usernam , user , log , document , contain , panel , earlier , weak , hash , txt , includ , snif , kei , decrypt , phpinfo , backup , plesk		
Topic 05	Server-side SQL injection (including blind vectors, numerical parameters).	sql , inject , command , index , earlier , action , multipl , com , joomla , compon , _id , magic , disabl , _quotes_gpc , usernam , cat , new , asp , view , page , categori , aka , catid , cid , beta , portal , search , mambo , board , articl , possibl , book , link , list , classifi , login , blog , pid , pro , modul	SkipFish	0.08
Topic 05	SQL Injection	sql , inject , command , index , earlier , action , multipl , com , joomla , compon , _id , magic , disabl , _quotes_gpc , usernam , cat , new , asp , view , page , categori , aka , catid , cid , beta , portal , search , mambo , board , articl , possibl , book , link , list , classifi , login , blog , pid , pro , modul	None-CWE	0.05
Topic 06	Denial of Service	servic , caus , denial , crash , possibl , applic , memori , corrupt , long , code , craft , hang , daemon , free , trigger , unspecifi , version , earlier , bound , engin , multipl , argument , doubl , oper , string , shutdown , empti , platform , devic , nicknam , ircd , audio , persist , unknown , openttd , alpha , seri , instabl , thread , counter	AppScan	0.05
Topic 09	Assorted file POIs (server-side sources, configs, etc).	internet , explor , web , microsoft , page , aka , object , server , url , code , contain , jsp , access , net , html , refer , sourc , browser , read , applic , com , help , zone , extens , demonstr , framework , modifi , sap , view , certain , outlook , document , cach , uri , dll , content , tiki , servlet , delet , window	SkipFish	0.09
Topic 12	Content Spoofing	user , content , domain , type , doe , spoof , open , url , assist , origin , displai , site , bypass , frame , download , address , target , window , link , dialog , polici , secur , opera , trust , conduct , bar	AppScan	0.05

Topic	Risk	Top Terms	scanner name	Q-gram Score
		, check , prevent , trick , event , set , featur , verifi , malici , properli , possibl , box , phish , read , save		
Topic 13	Stored and reflected XSS vectors in document body (minimal JS XSS support present).	script , site , web , html , cross , xss , inject , multipl , earlier , index , search , page , field , action , possibl , login , messag , guestbook , version , email , queri , uri , titl , beta , view , comment , text , portal , url , msg , theme , insert , jsp , usernam , link , keyword , new , result , cfm , templat	SkipFish	0.06
Topic 13	Cross-site Scripting	script , site , web , html , cross , xss , inject , multipl , earlier , index , search , page , field , action , possibl , login , messag , guestbook , version , email , queri , uri , titl , beta , view , comment , text , portal , url , msg , theme , insert , jsp , usernam , link , keyword , new , result , cfm , templat	AppScan	0.06
Topic 16	Mixed content problems on script and CSS resources (optional).	http , request , server , header , web , apach , url , respons , contain , mod , crlf , charact , uri , earlier , proxi , encod , post , content , conduct , refer , sequenc , irc , sourc , tomcat , host , channel , log , asterisk , messeng , modul , split , chat , slash , client , demonstr , version , agent , forward , send , aim	SkipFish	0.09
Topic 16	Stored and reflected XSS vectors via HTTP header splitting.	http , request , server , header , web , apach , url , respons , contain , mod , crlf , charact , uri , earlier , proxi , encod , post , content , conduct , refer , sequenc , irc , sourc , tomcat , host , channel , log , asterisk , messeng , modul , split , chat , slash , client , demonstr , version , agent , forward , send , aim	SkipFish	0.08
Topic 17	Buffer Overflow	overflow , buffer , code , long , base , stack , heap , multipl , function , earlier , craft , user , possibl , assist , string , argument , command , version , filenam , dll , field , playlist , mai , line , trigger , contain , cpp , player , request , pars , malform , pro , hostnam , librari , entri ,	AppScan	0.07

Topic	Risk	Top Terms	scanner name	Q-gram Score
		winamp , build , tftp , beta , header		
Topic 22	Permissions, Privileges, and Access Control	user , directori , search , path , local , librari , share , trojan , hors , untrust , current , folder , board , work , dll , privileg , gain , malici , code , load , point , program , possibl , power , locat , modifi , place , earlier , invis , pathnam , contain , python , variabl , technolog , demonstr , _library_path , bulletin , zero , unc , version	None-CWE-(manual)	0.06
Topic 23	Bad caching directives on cookie setting responses.	doe , bypass , restrict , access , properli , make , intend , easier , check , kei , secur , protect , conduct , forc , function , certain , configur , authent , valid , gener , valu , disabl , implement , mechan , brute , time , set , perform , session , requir , leverag , modifi , provid , enabl , guess , return , acl , random , polici , signatur	SkipFish	0.06
Topic 23	Insufficient Session Expiration	doe , bypass , restrict , access , properli , make , intend , easier , check , kei , secur , protect , conduct , forc , function , certain , configur , authent , valid , gener , valu , disabl , implement , mechan , brute , time , set , perform , session , requir , leverag , modifi , provid , enabl , guess , return , acl , random , polici , signatur	AppScan	0.06
Topic 23	Permissions, Privileges, and Access Control	doe , bypass , restrict , access , properli , make , intend , easier , check , kei , secur , protect , conduct , forc , function , certain , configur , authent , valid , gener , valu , disabl , implement , mechan , brute , time , set , perform , session , requir , leverag , modifi , provid , enabl , guess , return , acl , random , polici , signatur	None-CWE	0.06
Topic 23	Configuration	doe , bypass , restrict , access , properli , make , intend , easier , check , kei , secur , protect , conduct , forc , function , certain , configur , authent , valid , gener , valu , disabl , implement , mechan ,	None-CWE	0.06

Topic	Risk	Top Terms	scanner name	Q-gram Score
		brute , time , set , perform , session , requir , leverag , modifi , provid , enabl , guess , return , acl , random , polici , signatur		
Topic 23	Insufficient Authentication	doe , bypass , restrict , access , properli , make , intend , easier , check , kei , secur , protect , conduct , forc , function , certain , configur , authent , valid , gener , valu , disabl , implement , mechan , brute , time , set , perform , session , requir , leverag , modifi , provid , enabl , guess , return , acl , random , polici , signatur	AppScan	0.06
Topic 25	SQL Injection	asp , multipl , search , _id , forum , cart , shop , calendar , inject , pro , cat , action , product , default , script , event , field , categori , poll , earlier , login , email , manag , softwar , date , year , net , keyword , order , _name , possibl , view , new , sort , msg , _list , faq , banner , month , lite	None-CWE-(manual)	0.06
Topic 26	Permissions, Privileges, and Access Control	user , privileg , local , gain , root , program , command , variabl , line , environ , argument , access , option , earlier , aix , solari , setuid , run , process , group , unix , ibm , irix , modifi , configur , mai , specifi , sgi , drop , daemon , environment , possibl , oper , addit , linux , certain , util , sco , home , sun	None-CWE-(manual)	0.09
Topic 28	Permissions, Privileges, and Access Control	secur , antiviru , archiv , viru , bypass , product , test , anti , symantec , multipl , suit , detect , demonstr , extens , internet , version , firewal , protect , zip , scan , craft , header , contain , filenam , micro , engin , comput , certain , trend , malwar , person , gatewai , parser , later , aka , includ , norton , error , tar , associ	None-CWE-(manual)	0.06
Topic 29	Cross-Site Request Forgery (CSRF)	user , authent , request , site , administr , action , cross , form , edit , csrf , forgeri , delet , perform , modifi , hijack , interfac , redirect , multipl , add , web , chang ,	None-CWE	0.08

Topic	Risk	Top Terms	scanner name	Q-gram Score
		open , account , field , manag , ticket , earlier , creat , conduct , certain , unauthor , admin , page , demonstr , new , hord , registr , webmail , aspx , aka		
Topic 32	Race Conditions	kernel , user , linux , local , function , memori , doe , condit , process , certain , solari , red , race , hat , panic , driver , sun , ioctl , read , properli , kerbero , bit , snv , filesystem , mount , structur , relat , platform , guest , caus , oper , freebsd , netbsd , write , socket , devic , run , opensolari , virtual , subsystem	None-CWE	0.05
Topic 38	Directory traversal (including constrained vectors).	directori , dot , travers , read , includ , sequenc , local , index , earlier , filenam , list , multipl , pathnam , compon , disabl , possibl , languag , com , absolut , magic , joomla , backslash , _quotes_gpc , lang , write , download , control , page , slash , overwrit , creat , templat , beta , action , dir , view , encod , uri , path , _file+C1	SkipFish	0.12
Topic 38	Path Traversal	directori , dot , travers , read , includ , sequenc , local , index , earlier , filenam , list , multipl , pathnam , compon , disabl , possibl , languag , com , absolut , magic , joomla , backslash , _quotes_gpc , lang , write , download , control , page , slash , overwrit , creat , templat , beta , action , dir , view , encod , uri , path , _file	None-CWE	0.05
Topic 38	Directory Indexing	directori , dot , travers , read , includ , sequenc , local , index , earlier , filenam , list , multipl , pathnam , compon , disabl , possibl , languag , com , absolut , magic , joomla , backslash , _quotes_gpc , lang , write , download , control , page , slash , overwrit , creat , templat , beta , action , dir , view , encod , uri , path , _file	AppScan	0.05
Topic 39	External untrusted script and CSS inclusion vectors.	firefox , mozilla , javascript , tag , element , attribut , seamonkei , thunderbird , uri , code , relat , document , browser , style , img , html , url , css , demonstr , engin , function , chrome , data , src , object , cpp , properti ,	SkipFish	0.05

Topic	Risk	Top Terms	scanner name	Q-gram Score
		implement , page , googl , event , sheet , ifram , opera , cascad , contain , free , craft , involv , bbcode		

11.4 Appendix D: Terms in each topic

Table 11-5. Top Ten Terms in Each Topic

Topic	Top 10 Terms
Topic 00	integ function valu craft overflow code imag trigger context larg
Topic 01	cgi command build shell metacharact vmware program earlier host script
Topic 02	string function format charact code variabl specifi earlier demonstr contain
Topic 03	control activex dll method code argument media realplay player ocx
Topic 04	inform obtain sensit password store access web control root request
Topic 05	sql inject command index earlier action multipl com compon joomla
Topic 06	servic caus denial crash possibl applic memori corrupt long code
Topic 07	oracl unspecifi compon vector unknown affect integr relat confidenti user
Topic 08	password authent upload administr access cooki login usernam session admin
Topic 09	internet explor web microsoft page aka object server url code
Topic 10	java updat earlier sun jre class applic runtim environ applet
Topic 11	caus servic denial crash null pointer trigger derefer malform function
Topic 12	user content domain type doe spoof url assist open site
Topic 13	script site web html cross xss inject multipl earlier index
Topic 14	properli appl doe handl craft mac code applic memori webkit
Topic 15	server mail command ftp messag client address certif authent user
Topic 16	http request server header web apach url respons contain mod
Topic 17	overflow buffer code long base stack heap multipl function earlier
Topic 18	cve differ vector adob code unspecifi issu relat player reader
Topic 19	vector unspecifi unknown impact relat involv googl multipl chrome possibl
Topic 20	servic caus denial port packet consumpt connect larg tcp number
Topic 21	cisco devic aka bug firmwar interfac router seri commun servic
Topic 22	user directori search path local trojan librari share hors untrust
Topic 23	doe bypass restrict access properli make intend easier check kei
Topic 24	inform obtain note parti thi unknown proven sole multipl aka
Topic 25	asp multipl search _id forum shop cart inject calendar pro
Topic 26	user privileg local gain root command program variabl environ line
Topic 27	code url inclus earlier multipl regist enabl _global includ index
Topic 28	secur antiviru viru archiv bypass product anti multipl symantec test

Topic	Top 10 Terms
Topic 29	user authent request site administr action cross csrf form forgeri
Topic 30	manag server ibm servic weblog agent unspecifi consol applic network
Topic 31	note thi issu cve disput report vendor becaus research origin
Topic 32	kernel user local linux function memori doe condit certain process
Topic 33	window server microsoft aka gold craft vista novel servic win
Topic 34	error messag path obtain inform sensit request instal reveal direct
Topic 35	offic microsoft code craft aka excel memori corrupt document mac
Topic 36	modul field drupal comment galleri post aka user plugin multipl
Topic 37	user local creat overwrit permiss symlink temporari instal log modifi
Topic 38	dot directori travers read includ sequenc local index earlier filenam
Topic 39	firefox mozilla javascript tag element attribut seamonkei thunderbird cod