

QUANTITATIVE RISK ASSESSMENT MODEL FOR SOFTWARE SECURITY IN
THE DESIGN PHASE OF SOFTWARE DEVELOPMENT

Except where reference is made to the work of others, the work described in this dissertation is my own or was done in collaboration with my advisory committee. This dissertation does not include proprietary or classified information.

Idongesit Okon Mkpong-Ruffin

Certificate of Approval:

David A. Umphress, Co-Chair
Associate Professor,
Computer Science Software Engineering

John A. Hamilton, Co-Chair
Associate Professor,
Computer Science Software Engineering

Juan E. Gilbert
Associate Professor,
Computer Science Software Engineering

George T. Flowers
Dean
Graduate School

QUANTITATIVE RISK ASSESSMENT MODEL FOR SOFTWARE SECURITY IN
THE DESIGN PHASE OF SOFTWARE DEVELOPMENT

Idongesit Mkpong-Ruffin

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Doctor of Philosophy

Auburn, Alabama

May 9, 2009

QUANTITATIVE RISK ASSESSMENT MODEL FOR SOFTWARE SECURITY IN
THE DESIGN PHASE OF SOFTWARE DEVELOPMENT

Idongesit Mkpong-Ruffin

Permission is granted to Auburn University to make copies of this dissertation at its discretion, upon request of individuals or institutions and at their expense. The author reserves all publication rights.

Signature of Author

Date of Graduation

DISSERTATION ABSTRACT

QUANTITATIVE RISK ASSESSMENT MODEL FOR SOFTWARE SECURITY IN
THE DESIGN PHASE OF SOFTWARE DEVELOPMENT

Idongesit Mkpong-Ruffin

Doctor of Philosophy, May 9, 2008
(M.S. Troy University, 2007)
(M.B.A, Tennessee State University, 1992)
(B.S. Freed-Hardeman University, 1985)

186 Typed Pages

Directed by David A. Umphress and John A. Hamilton

Risk analysis is a process for considering possible risks and determining which are the most significant for any particular effort. Determining which risks to address and the optimum strategy for mitigating said risks is often an intuitive and qualitative process. An objective view of the risks inherent in a development effort requires a quantitative risk model. Quantitative risk models used in determining which risk factors to focus on tend to use a traditional approach of annualized loss expectancy (ALE) based on frequency of occurrence and the exposure factor (EF) which is the percentage of asset loss due to the potential threat in question. This research uses empirical data that reflects the security posture of each vulnerability to calculate Loss Expectancy, a risk impact

estimator. Data from open source vulnerability databases and results of predicted threat models are used as input to the risk model. Security factors that take into account the innate characteristics of each vulnerability are incorporated into the calculation of the risk model. The result of this model is an assessment of the potential threats to a development effort and a ranking of these threats based on the risk metric calculation.

ACKNOWLEDGEMENTS

I have been very blessed to have been advised, supported, and encouraged by the different members of my committee during the different stages of my educational journey. I thank Dr David Umphress for his patience, deliberate guidance, and direction. I would also like to thank Dr. John Hamilton for his insight and support. I would like to give my thanks to Dr. Juan Gilbert for his support and encouragement throughout my stay at Auburn.

I would like to thank the members of the Human Centered Computing Lab (HCCL), the Information Assurance Lab and other members of the Computer Science department for their support. Their helpful comments, suggestions and acts of kindness are very much appreciated.

I would also like to thank my family, especially my parents, Obong Okon Mkpog and, ObongAwan Affiong Mkpog, for instilling in me the zeal for knowledge, as well as my sisters and brothers for their continued support and belief in me and my ability to achieve this milestone. A very special thanks to my husband, Dale, and children, Akaninyene and Aniekan, who have been my motivation. Without their support, patience and encouragement I would not have started nor finished this program. I dedicate this work to them.

Most of all, special thanks to God-- for His mercies, for putting each person in my path to strengthen me and make this journey what it has become. His protection and guidance have been very evident, and I thank Him.

Style manual or journal used: ACM Computing Surveys

Computer software used: Microsoft Word, Microsoft Excel, Microsoft Visual Studio (C#, SQL Server, SQL Analysis Server)

TABLE OF CONTENTS

TABLE OF CONTENTS	viii
LIST OF FIGURES.....	xii
LIST OF TABLES	xiv
LIST OF EQUATIONS	xv
1 INTRODUCTION.....	1
1.1 Problem Statement	1
1.2 Research Objective.....	2
1.3 Background	2
1.3.1 Risk Assessment.....	4
1.3.2 Threat Identification.....	6
1.3.3 Vulnerability Identification	7
1.3.4 Likelihood Determination	7
1.3.5 Impact Determination.....	8
1.3.6 Risk Determination	9
2 STATE OF RISK ASSESSMENT.....	11
2.1 Threat/Vulnerability Identification	11
2.1.1 Threat/Vulnerability Categorization	12
2.2 Software Testing and Assessment.....	13
2.2.1 Black-Box Testing.....	14

2.2.2 White-Box Testing	15
2.3 Stakeholder's Goals.....	17
2.3.1 Boehm Win-Win approach.....	17
2.3.2 RISKIT	18
2.4 Risk Management Framework (RMF)	19
2.5 Software Reliability.....	20
2.6 Traditional approach	20
2.7 Cross-Disciplinary ties to Insurance	21
2.8 Problems with Existing Approaches	21
3 RESEARCH DETAILS	25
3.1 Conceptual Overview.....	26
3.2 Data Preparation.....	27
3.3 Parsing the Data	27
3.3.1 Validation of Data Entry	29
3.4 Cluster Determination	30
3.4.1 Validation of Clustering.....	35
3.5 Data Classification	37
3.5.1 Data Classification Validation	41
3.6 Loss Expectancy Determination.....	42
3.6.1 Determining Fields to Predict CVSS Score	43

3.6.1.1	Regression Analysis: Cluster Node versus Loss Type, Vulnerability Type and Exploit Range.....	45
3.6.1.2	Regression Analysis: CVSS Score versus Loss Type, Vulnerability Type and Exploit Range.....	47
3.6.2	Determining Loss Expectancy	48
3.6.2.1	Validation of Loss Expectancy Calculations	50
3.6.3	Validation of Loss Expectancy	52
3.7	Confidence Interval of Predictions.....	57
3.8	Summary	58
4	APPLICATION OF MODEL TO CASE STUDY	61
4.1	Vulnerability Identification and Data Preparation	62
4.2	Data Classification and Loss Expectation Determination.....	64
5	SUMMARY, FUTURE WORK AND CONCLUSIONS.....	71
5.1	Summary	71
5.2	Future Work	74
5.3	Conclusions	75
	BIBLIOGRAPHY	78
	APPENDICES.....	84
	Appendix A-1 – The NVD Schema	85
	Appendix A-2 – An Example of an NVD entry.....	92

Appendix A-3 – Script for Parsing XML document.....	94
Appendix A-4 – Validation of Data Upload	112
Appendix B-1 – Creating Term Vector.....	113
Appendix B-2 – Determining Clustering Algorithm.....	117
Appendix B-3 – Query for Identifying Clusters in Training Data	127
Appendix B-4 – Classified Data Matrix.....	129
Appendix B-5 – Validation of Clustering Algorithms.....	132
Appendix B-6 – Processing Time for Classifiers.....	135
Appendix B-7 – Calculating CVSS ‘base’ Score.....	136
Appendix C-1 – Stored Procedure for Calculating Impact Score Attributes	138
Appendix C-2 – Stored Procedure for Determining Prioritized Listing	140
Appendix C-3 – Stored Procedure for Persisting Predictions	141
Appendix D-1 – Singleton Request Algorithm.....	142
Appendix D-2 – Data Table Request (Code for Load Data).....	146
Appendix D-3 – Data for SSRAM Validation	155
Appendix E-1 – Microsoft’s TAMT Listing of Vulnerabilities.....	160
Appendix E-2 – Prime III Data	165
Appendix E-3 – Prime III Predictions.....	166
Appendix F- Vulnerability Classification	171

LIST OF FIGURES

Figure 1 -- Difference between Speculative and Hazard Risk [Alberts 2006].....	3
Figure 2 - Risk Function.....	4
Figure 3 - Risk Management Cycle [GAO AIMD-00-33].....	5
Figure 4 - Top-N Risk Item List[Boehm 2001]	18
Figure 5 - Definition of Risk in Riskit Method [Kontio et al. 1998]	19
Figure 6 -Example of the Riskit analysis graph (risk scenarios) [Kontio et al. 1998]	19
Figure 7 -Reported Incidences [NVD].....	25
Figure 8 - SSRAM's Cluster Creation.....	26
Figure 9 - Software Security Risk Assessment.....	26
Figure 10 - NVD's First Level Node.....	28
Figure 11- NVD Description Node.....	28
Figure 12 - Node with nested lower level tags.....	29
Figure 13 - SSRAM Schema.....	29
Figure 14 - Independent Variable Selection Statement.....	31
Figure 15 -Clustering without Description.....	33
Figure 16 - Clustering with Description.....	34
Figure 17 - Cluster Assignment Query Example	35
Figure 18 -Mining Accuracy Chart.....	36
Figure 19 -Classifiers with Description Vector.....	40

Figure 20 - Classifiers without Description Vector	40
Figure 21 - Prediction Accuracy of Classifiers	42
Figure 22 - CVSS base Score and Vector	43
Figure 23 - Suggested Input Fields	44
Figure 24 - Multiple Regression Result (Cluster Node)	46
Figure 25 - Multiple Regression Result (CVSS score)	47
Figure 26-Query for Sample data to validate Calculation.....	50
Figure 27 - Calculation of Impact Factors Result (Stored Procedure).....	51
Figure 28 -Single Entry Impact Estimation.....	52
Figure 29 Vulnerability Entries (Test Data – 2002).....	53
Figure 30 - Predicted Values	53
Figure 31 - Impact and Loss Expectation Estimation	53
Figure 32: Prime III System Architecture High-Level Overview	62
Figure 33 - Load Prime III	65
Figure 34 - Prime III De-normalized Predictions.....	65
Figure 35 - Prime III Loss Expectation (Prioritized List) – Predictions	69
Figure 36 - Prime III Loss Expectation (Prioritized List) – 2006 Predictions	69
Figure 37 - Processing Time for Classification Algorithm.....	135

LIST OF TABLES

Table 1 Likelihood Definitions [SP 800-30].....	8
Table 2 Magnitude of Impact Definitions [SP 800-12].....	9
Table 3 Risk-Level Matrix	10
Table 4 Software Risk Management Techniques [Boehm 2001]	17
Table 5 - Cluster Algorithm Cohesion Factors	37
Table 6 - Classifier Processing Time & Score – Cluster Classifier without Description .	39
Table 7 - Classification Confusion Matrix – Classifier without Description.....	41
Table 8 – Calculation of Impact Factors (Excel Result)	51
Table 9 - t-Test: Paired Two Sample for Means	54
Table 10 - Confidence Interval Derivation	57
Table 11 - Prime III Vulnerability List	63
Table 12- Prime III Normalized Tables	64
Table 13 - Example of PrimeIII's Input Data.....	64
Table 14 – Comparison of SSRAM with other Risk Analysis Methodologies.....	76
Table 15 - Data Entry Validation Entries.....	112

LIST OF EQUATIONS

Equation 1 - Traditional Annual Loss Expectancy 21

Equation 2 - TFDIF 31

Equation 3 – Accuracy Determination 41

Equation 4 – Predicted Loss Expectancy (PLE) 48

Equation 5 - Predicted Impact Score (PIS) 49

Equation 6 - Weighted Average CVSS Score..... 49

Equation 7 - Growth Rate..... 49

Equation 8 - Margin of Error..... 57

1 INTRODUCTION

1.1 Problem Statement

Assessing security risks in software is predominately a qualitative process. Traditionally, efforts to deal with security vulnerabilities focus on hardening networks and peripherals that have access to computer systems. Hardening of networks and peripherals entails the application of security measures, such as firewalls and virtual private networks, or removing non secure systems and services from the network. In 2002, the National Institute of Standards (NIST) reported that \$59.5 billion was spent annually in breakdowns and repairs of faulty software [NIST 2002-10] [Mead and Stehney 2005]. NIST also found that 92% of all security vulnerabilities were due to application vulnerabilities as opposed to network vulnerabilities [Curphey 2004]. Efforts have been underway to deal with application vulnerabilities early in the software development life cycle. These efforts have underscored the fact that risk management should drive the software development process, which assures that security is made an emergent feature of the development process [Mkpong-Ruffin and Umphress 2007].

Methodologies for performing risk management activities – risk identification, risk assessment and risk mitigation – are largely qualitative in nature. Risk identification approaches such as attack trees, attack nets, and attack pattern matching aid in determining and identifying the risks that exist in a development effort [Mkpong-Ruffin

and Umphress 2007, Jurjens 2001, Sindre and Opdahl, Microsoft 2006]; whereas risk assessment methodologies and tools aid in determining what and where resources can be allocated to ameliorate the risks identified [Boehm 2001, SP 800-30, FIPS 2004, Kontio 1999, Voas et al. 1997]. The qualitative nature of these methodologies makes it difficult to generalize assessment and duplicate results from other projects. Since effective risk mitigation strategies are dependent upon the results of the risk assessment process, there needs to be an empirical means for implementing risk assessment.

1.2 Research Objective

The objective of this research is to use historical data as a basis to categorize and quantitatively assess risk elements by incorporating risk impact factors in the assessment model. This objective is articulated through the creation of a software security risk assessment model (SSRAM) that categorizes, estimates, and ranks risk elements that have been identified during a threat modeling activity in the design phase of a development effort. The result of the research is a quantitative risk assessment of software security for a development effort.

1.3 Background

Risk is defined as the estimation of the probability and severity of an adverse effect [Haimes 2004]. There are two kinds of risks: speculative and hazard risk [Voas et al. 1998; Young 2001]. A speculative risk, as in a stock investment, may produce a loss or a profit. A hazard risk, on the other hand, always produces a loss; see Figure 1 [Alberts 2006]. In dealing with risk, the prevalent approach is to find a plan that will reduce the loss incurred or the actual occurrence of the risk. This plan can take the form of insurance purchased to reduce the effect of the loss if it were to happen, installation of

deterrent devices around existing structures, or the incorporation of restraints within the construct of the system. For example, a homeowner could buy insurance, install security systems on an existing home, or incorporate burglary bars and other theft resistance structures into the home.

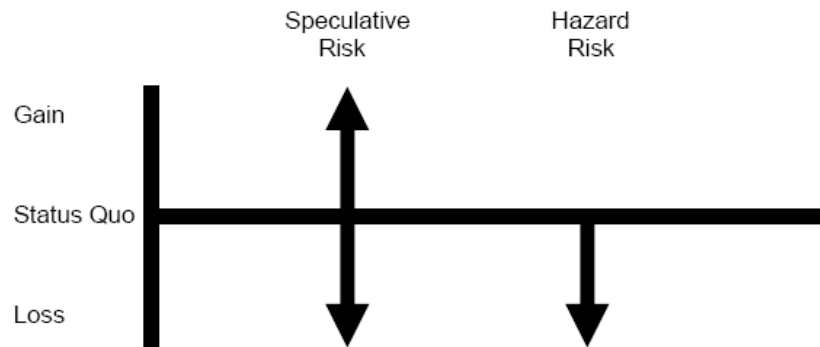


Figure 1 -- Difference between Speculative and Hazard Risk [Alberts 2006]

Risk exists when the exercise of a vulnerability produces a net negative impact. Risk management is the process of identifying, assessing, and taking the steps to reduce risk to a satisfactory level. It takes into account the probability and impact of the occurrence of risk factors. This process allows operational and economic costs to be balanced while still protecting the development effort. It minimizes the negative impact on the organization and gives the sound basis needed for decision-making. Risk management allows for a better security posture in that it enables stakeholders “to make well-informed risk management decisions to justify the expenditures” that are a part of the IT budget for the development effort. It also assists stakeholders in “authorizing (or accrediting)” the development effort based on the documentation derived from the performance of risk management [SP 800-30; SP 800-53].

Effective risk management gives software engineers the necessary focus to understand the stakeholders' objectives. It also provides a context for exploring solution approaches; thereby reducing the risk of building the wrong system. Effective risk management also makes it possible for risks to be resolved early, avoiding extensive rework late into the project [Boehm 2001]. These reasons show the need for risk management to be fully integrated into the software development life cycle [SP 800-30; SP 800-55; Boehm 2001].

1.3.1 Risk Assessment

Figure 2 below, shows a high-level illustration of the risk assessment process [SP 800-100].



Figure 2- Risk Function

Risk assessment is the process of estimating the impact of a successful exploitation and determining the likelihood of an attacker successfully exploiting a given vulnerability. The estimation of impact is done by looking at the effect an exploitation can have on the confidentiality, integrity and availability of the system [SP 800-100].

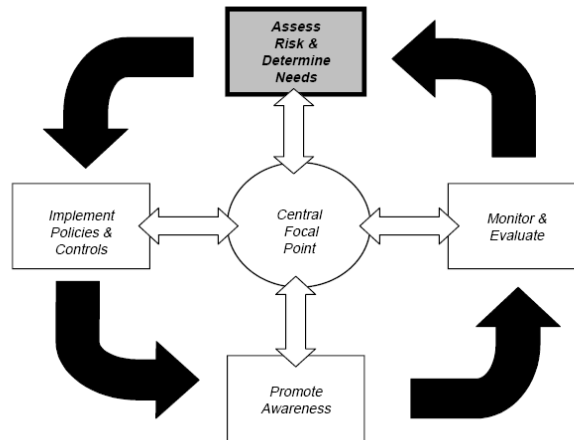


Figure 3 - Risk Management Cycle [GAO AIMD-00-33]

Risk assessment provides the foundation for the risk management process. As shown in Figure 3, risk assessment affords the basis for the establishment of appropriate policies upon which the other elements of risk management are directed [GAO AIMD-00-33]. It determines the extent of the potential threat and the risks associated with the software developed, throughout its life cycle [Voas et al 1997]. It also provides decision makers with needed information that allows them to understand factors that can negatively impact operations and outcomes. Additionally, risk assessment allows decision makers to make informed judgments about actions to take and the extent to which such actions should be taken to reduce risk. The output of this process aids in finding out what controls will be needed to reduce or eliminate risks during the risk mitigation process. For example, bank officials make judgments concerning loan portfolios based on the results of risk assessment. Insurance providers use risk assessment to determine the amount to charge for insurance provided. Nuclear power plant engineers conduct risk assessments to ascertain risks to public health and safety. With the increased dependence on computer systems, the growth of electronic data and the ubiquity of software, risk assessment in the software development effort is critical.

In order to provide adequate assessment of the risks for a development effort, threats and vulnerabilities have to be identified. The likelihood and impact of each risk occurring have to be determined. In general, risk analysis entails [SP 800-30; McGraw 2006; GAO AIMD-00 33 11/99; Boehm1989]:

- Determination of the system's character under question - understanding the system boundaries, the data sensitivity and criticality, and the stakeholders' perspectives on the system
- Identification of the threats – attackers that would likely want to attack the system
- Identification of vulnerabilities – flaws that exist or could exist in each level of the development and the operating environment
- Determination of the likelihood of a vulnerability being exploited.
- Analysis of current and planned controls to mitigate the threats/vulnerabilities exposed
- Determination of the impact on the system and the organization should the risk be realized
- Determination of the risks based on the threat and vulnerability identification, the likelihood of exploitation and impact on the system.
- Determination of controls to mitigate stated risks

1.3.2 Threat Identification

Without a vulnerability to exploit, an attacker cannot exercise a threat. Such a threat-source can come either from a deliberate attack or from an accidental activation of a vulnerability. It is generally accepted that a compilation of potential threat sources be made that are applicable to the IT system under evaluation. These should be adapted to

the organization and its environment. Information on threats and threat sources are now available from different sources, such as:

- Federal Bureau of Investigation's National Infrastructure Protection Center (www.fbi.gov)
- Federal Computer Incident Response Center (FedCIRC) – www.us-cert.gov
- Mass media, particularly Web-based resources such as National Vulnerability Database (NVD), SecurityFocus.com, SecurityWatch.com, SecurityPortal.com, and SANS.org

1.3.3 Vulnerability Identification

Vulnerability is a flaw or weakness in a system's security procedure, design, implementation, or internal controls. Its exploitation could be either intentional or accidentally triggered and would result in a violation of the system's security policy or a security breach [SP 800-30; SP 800-12; Steel et al. 2005].

In making an analysis of the threats to an IT system, an analysis of the vulnerabilities associated with the system and its environment must also be made. A list of flaws or weakness that could be exploited by the probable threat-sources has to be developed. The different vulnerabilities can be categorized to allow for ease of identification. [Tsipenyuk et al. 2005; Howard et al. 2005; OWASP]

1.3.4 Likelihood Determination

Once vulnerabilities and attackers have been determined, the likelihood of a potential vulnerability being exercised within the associated threat environment has to be derived.

The attacker's motivation and capability, nature of the vulnerability, existence and effectiveness of current control are factors that govern likelihood determination. [SP 800-30; SP 800-100; SP 800-12; Steel et al. 2005]

To describe the likelihood determination most practitioners use qualitative measures of high, medium, low to describe likelihood levels as shown in Table 1.

Likelihood Level	Likelihood Definition
High	The threat-source is highly motivated and sufficiently capable; and the controls to prevent the vulnerability from being exercised are ineffective
Medium	The threat-source is motivated and capable, but controls are in place that may impede successful exercise of the vulnerability
Low	The threat-source lacks motivation or capability, or controls are in place to significantly impede, the vulnerability from being exercised.

Table 1 Likelihood Definitions [SP 800-30]

1.3.5 Impact Determination

When a vulnerability is exercised, the impact of the security event is usually described as either a degradation of one or a combination of the following security goals: integrity, availability and confidentiality. Stoneburner [SP 800-30] briefly describes each security goal and the result of the goal not being met as follows:

Loss of Integrity. System and data integrity refers to the requirement that information be protected from improper modification. Integrity is lost if unauthorized changes are made to the data or IT system by either intentional or accidental acts. If the loss of system or data integrity is not corrected, continued use of the contaminated system or corrupted data could result in inaccuracy, fraud, or erroneous decisions. Also, violation of integrity may be the first step in a successful attack against system availability or confidentiality. For all these reasons, loss of integrity reduces the assurance of an IT system.

Loss of Availability. If a mission-critical IT system is unavailable to its end users, the organization's mission may be affected. Loss of system functionality and operational effectiveness, for example, may result in loss of productive time, thus impeding the end users' performance of their functions in supporting the organization's mission.

Loss of Confidentiality. System and data confidentiality refers to the protection of information from unauthorized disclosure. The impact of unauthorized disclosure of confidential information can range from the jeopardizing of national security to the disclosure of Privacy Act data. Unauthorized, unanticipated, or unintentional

disclosure could result in loss of public confidence, embarrassment, or legal action against the organization. Some tangible impacts can be measured quantitatively in lost revenue, the cost of repairing the system, or the level of effort required to correct problems caused by a successful threat action. Other impacts (e.g., loss of public confidence, loss of credibility, damage to an organization's interest) cannot be measured in specific units but can be qualified or described in terms of high, medium, and low impacts.

Qualitative categories of high, medium and low are also used to describe the impact of these security goals (Table 2).

Magnitude of Impact	Impact Definition
High	Exercise of the vulnerability (1) may result in the highly costly loss of major tangible assets or resources or (2) may significantly violate, harm or impede an organization's mission reputation, or interest or (3) may result in human death or serious injury
Medium	Exercise of the vulnerability (1) may result in the costly loss of some tangible assets or resources or (2) may violate, harm or impede an organization's mission reputation, or interest or (3) may result in human injury
Low	Exercise of the vulnerability (1) may result in the loss of some tangible assets or resources or (2) may noticeably affect an organization's mission reputation, or interest.

Table 2 Magnitude of Impact Definitions [SP 800-12]

1.3.6 Risk Determination

Risk is a function of the likelihood of a given attacker's ability to exercise a potential vulnerability and the resulting impact of that adverse event on the organization. The impact realized is the degree of harm that could be caused when vulnerability is exercised. The level of impact is based on the relative value of the resources affected such as the sensitivity and criticality of the system's components and data [SP 800-30 SP 800-12; Steel et al. 2005]. The linking of system-level concerns with probability and impact measures that matter to a software development organization produces superior risk analysis [McGraw 2006; Voas et. al 1997]. As such, it is important that the threats to a system, the probable vulnerabilities, and the controls in existence be analyzed in tandem for the system in question.

Once the threat-sources, vulnerabilities, likelihood of occurrence and impacts have been determined, a risk measure can then be developed. Deciding the risk for a particular threat/vulnerability pair may be expressed as a function of:

- probability of a particular attacker trying to exercise a particular vulnerability
- the magnitude of impact if the vulnerability is successfully exercised
- the adequacy of planned or existing security controls for removing or reducing risk [SP 800-30]

Stoneburner advocates that a risk scale and risk-level matrix, shown in Table 3, be developed for measuring risk [SP 800-30].

Threat Likelihood	Impact		
	Low (10)	Medium (50)	High (100)
High (1.0)	Low $10 \times 1.0 = 10$	Medium $50 \times 1.0 = 50$	High $100 \times 1.0 = 100$
High (0.5)	Low $10 \times 0.5 = 5$	Medium $50 \times 0.5 = 25$	High $100 \times 0.5 = 50$
High (0.1)	Low $10 \times 0.1 = 1$	Medium $50 \times 0.1 = 5$	High $100 \times 0.1 = 10$

Risk Scale: High (>50 to 100); Medium (>10 to 50); Low (1 to 10) [SP 800-30]

Table 3 Risk-Level Matrix

When the risk has been determined, the results of the assessments can be documented and maintained; this allows for accountability. Since risks and threats change over time, it is important that risks and threats be reassessed periodically. Subsequent risk assessment can use the report as the basis for subsequent risk assessment and evaluations [GAO AIMD-00-33]. This information also assists designers and developers in challenging their own built up assumptions about their system [McGraw 2006]. Risk assessment early in the process molds and provides contextual framework for the assumptions developers have about the system.

2 STATE OF RISK ASSESSMENT

Risk management is an integral part of the software development process. Since risk assessment is the foundation for other risk management activities, it should drive the development process to ameliorate security issues. Developers are expected to identify, assess, rank, mitigate and manage risk through out the software product life cycle [Humphrey et al. 2004; Addison and Vallabh 2002]. As noted previously, methodologies used to allow risk to drive the development process have in large part been qualitative in nature.

2.1 Threat/Vulnerability Identification

To determine what threats and vulnerabilities exist in a development effort, methodologies for identification early in the development process exist. Some of them are:

- SecureUML – a modeling language that incorporates information pertinent to access control into applications modeled or defined using the Uniform Modeling Language (UML) [Lodderstedt et al. 2002]. It models security requirement for “well-behaved applications in predictable environment’ [McGraw 2006].
- UMLsec [Jurjens 2001] – an extension of UML to encapsulate the modeling of security-related features, such as confidentiality and access control.

- Abuse Cases [Sindre and Opdahl] – an approach that extends use-cases to include misuse-cases, showing side-by-side what behavior should be supported and/or prevented.
- Microsoft’s Threat Analysis and Modeling Tool (TAMT) – a tool that generates risks based on the components, roles, data, external dependencies and the application’s use-cases of a given development effort [Microsoft 2006].

2.1.1 Threat/Vulnerability Categorization

Different approaches have been used to categorize vulnerabilities. A representative list is given below.

- Neural Networks - have been used to categorize software security risks within software with different levels of success [Neumann 2002; Jain et al. 1996]. In most cases they are used with software metrics, such as McCabe’s complexity metrics, failure history, lines of code, etc., on modules that have already been written.

Neumann [Neumann 2002] utilized neural networks combined with factor and component analysis to generate a set of orthogonal independent variables that could be used in representing the dependent variable, the goal being to allow for the focus of testing efforts on the portions of code with the largest number of faults.

- STRIDE –Introduced by Michael Howard and David Leblanc [Howard and Leblanc], this approach relies heavily on cycling through a list of attacks and applying to each attack one or more of the different risk categories of - Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service or Elevation

of Privilege (STRIDE) - during analysis. McGraw finds it a good starting point that gives a useful list of things to consider when identifying risks [McGraw 2006].

- Tsipenyuk [Tsipenyuk et al. 2005] uses a taxonomy of seven “kingdoms” (Input validation and representation; API abuse; Security features, Time and State, Errors, Code quality, Encapsulation and Environment) to categorize risk elements. Under each kingdom are the specific elements relevant to the kingdom. For example SQL injection and buffer-overflow belong to the input-validation and representation kingdom.
- Others, such as the Open Web Application Security Project [OWASP] and [Howard et al. 2005] provide lists of types of vulnerabilities that can be used to facilitate adoption of vulnerability groupings. These lists tend to have a mixture of specific types of errors and groupings of vulnerabilities. For example, *cross-site scripting* and *SQL-injection* are discussed at the same level of abstraction as *Improper Error Handling* and *Trusting Network Address Information*. A mapping of the different types of vulnerabilities as given by Howard, OWASP and Tsipenyuk is shown in Appendix F.

2.2 Software Testing and Assessment

Traditionally, testing has been done to see if an application works correctly. For security testing, emphasis is placed on identifying and removing vulnerabilities that could result in security violations. It validates the effectiveness of the security measures that are in place [Pan 1999]. Just as with ordinary testing, security testing methods fall into one of two major categories: black-box or white-box testing.

What method the tester uses to carry out security assessment and testing depends on the tester's perspective with respect to the software component. Black-box tests are conducted when specific knowledge about the software internals is not known, and the test cases are constructed based on functional requirements. White-box tests are conducted when the application's internal structure is known.

2.2.1 Black-Box Testing

In black-box testing, the data used is taken from the specified functional requirements [Howard et. al. 2005; Hetzel 1988]. Testing approaches such as penetration, functional, risk-based, and unit testing are commonly used to perform black-box testing [Howard et. al. 2005; Hetzel 1988; Voas et al. 1998; Michael and Radosevich].

- Penetration testing looks at how easy it would be for a component to be broken into.
- Functional testing looks to see whether software behaves as it was required to behave and whether it adheres to the given functional requirements.
- Risk-based testing is a subset of functional testing that focuses on those negative requirements that do not have a direct code base to be tested against. For example, the requirement to make sure that the application is not vulnerable to buffer overflows would be hard to map to a particular segment of code but still requires that testing be implemented to address this requirement. Risk-based testing is usually done by testing for misuse and abuse cases [Michael and Radosevich 2005].
- Unit security testing looks at how an adversary could gain access to the software and then take control of the software after gaining access. Given this two-stage

approach to unit testing, many use attack trees as a method for identifying and modeling threats since attack trees are amenable to portraying the different stages of implementation of a threat [Schneier 2000].

2.2.2 White-Box Testing

White-box approaches use test and assessment activities to show the structure and flow of the software under review. The program structure provides the basis for the test cases generated [Mkpong-Ruffin and Umphress 2007]. Commonly used white-box testing and assessment methods are fault-injection, source-code analysis and profiling.

Fault-Injection

Fault-injection uses information based on test cases – flaws – to show the effects of successfully exploiting a vulnerability by adding code that would forcefully change program state. This approach gives insight into predictive measures of risk such as minimum-time-to-hazard and mean-time-to-failure. Wallace defines hazard as “an unsafe condition that may lead to an unintended event that causes an undesirable outcome” [Wallace 1991].

Fault-injection allows for absolute worst-case prediction [Voas et al. 1998]. Jeffery Voas and colleagues distill the problems that software risks could stem from to the following: “erroneous input data (from sensors, humans, or stored files), faulty code, or a combination of the two” [Voas et al. 1997]. They postulate that accurate assessment of liability requires worst-case predictions of software outputs if any of the aforementioned problems would occur. To identify faulty or defective code, analytical techniques such as formal code verification and testing can be employed. To detect wrong input data and faulty code, fault injection should be performed.

Injecting anomalies allows the simulation of the effects that the real anomalies would probably have and makes it feasible to build up a body of knowledge about probable future behavior. The simulation of the effects makes it possible to quantify the risks that a system's component would create and allows observation of how bad things could get. Being able to quantify the risks gives a way to determine the 'boundaries of liability' [Voas et al. 1997]. Like other testing techniques, fault-injection is a technique utilized after a development effort is well under way.

Static Analysis

Static analysis tools look at the text of program, while not in execution, to discover potential vulnerabilities within the program. For example, many vulnerabilities are known to come from reusable library functions such as C's `strcpy()` and `stat()`. A static analyzer could scan the programs to see if they contain any calls to those functions [Mkpong-Ruffin and Umphress 2007] and then analyze the potential vulnerabilities uncovered to ascertain that they do not lead to security violations [Janardhanudu].

Profiling

Profiling tools allow the tester to observe the application's performance while in execution to see where bottlenecks occur. It also allows for observation and understanding of the sequence of function calls and the time spent in different areas of the software. This observation of the performance bottlenecks of the software with profiling tools allows for a better understanding of vulnerability areas, such as areas with memory leaks, which would not be apparent with the use of static code analyzers [Steel et al. 2005].

In all the instances given for software testing and assessment, these methodologies are at a lower level of abstraction in the development process.

2.3 Stakeholder's Goals

2.3.1 Boehm Win-Win approach

Boehm advocates using a risk management cycle consisting of risk identification, risk assessment and risk tracking. To this end, risk identification aided by software risk management techniques [Boehm 2001] proactively identifies, as early as possible, the possible sources of significant risks (see Table 4).

<i>Source of Risk</i>	<i>Risk Management Techniques</i>
Personnel shortfalls	Staffing with top talent; key personnel agreements; team-building; training; tailoring process to skill mix; walkthroughs.
Schedules, budgets, process	Detailed, multi-source cost and schedule estimation; design to cost; incremental development; software reuse; requirements descoping; adding more budget and schedule; outside reviews.
COTS, external components	Benchmarking; inspections; reference checking; compatibility prototyping and analysis
Requirements mismatch	Requirements scrubbing; prototyping; cost-benefit analysis; design to cost; user surveys
User interface –mismatch	Prototyping; scenarios; user characterization (functionality; style, workload); identifying the real users
Architecture, performance, quality	Simulation; benchmarking; modeling; prototyping; instrumentation; tuning
Requirements changes	High change threshold; information hiding; incremental development (defer changes to later increments)
Legacy software	Reengineering; code analysis; interviewing; wrappers; incremental deconstruction
Externally-performed tasks	Pre-award audits, award-fee contracts, competitive design or prototyping
Straining computer science	Technical analysis; cost-benefit analysis; prototyping; reference checking

Table 4 Software Risk Management Techniques [Boehm 2001]

The cycle would then continue with resolving each risk and, if not possible, addressing the resolution in the risk management plan by re-scoping the risk. Barry Boehm [Boehm 2001] uses the ‘Top-N Risk Item List’ to monitor risk management. This process gives a

summary of the risk items that are either growing or decreasing in criticality. The elaboration of the cycle is shown in Figure 4.

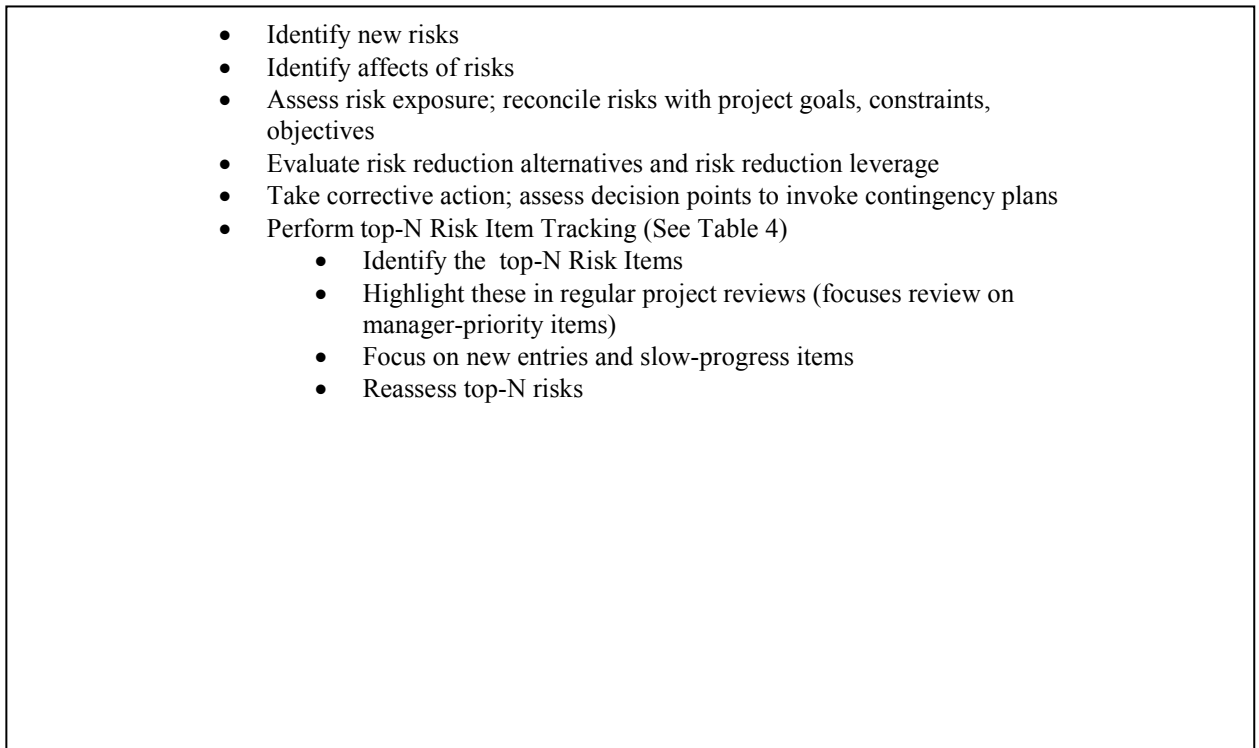


Figure 4 - Top-N Risk Item List[Boehm 2001]

2.3.2 RISKIT

RISKIT [Kontio 1997] is a method for risk management developed by J. Kontio. It incorporates utility theory with a support for the perspective of different stakeholders' goals, by the maintenance of links between risks and stakeholders (See Figure 5). It utilizes analysis graphs, as shown in Figure 6, for visual documentation of risks. The goal of the analysis graph is to give a deeper understanding of the process thereby enhancing communication [Kontio et al. 1998].

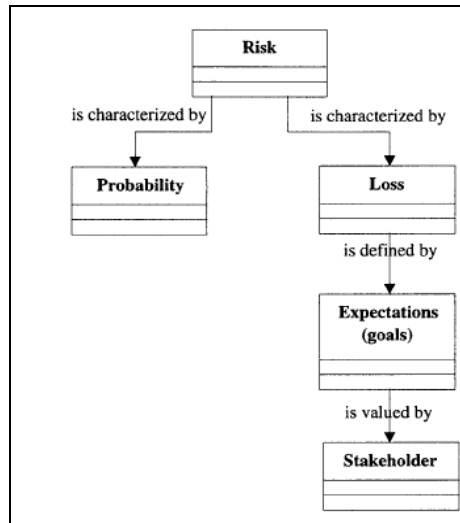


Figure 5 - Definition of Risk in Riskit Method [Kontio et al. 1997]

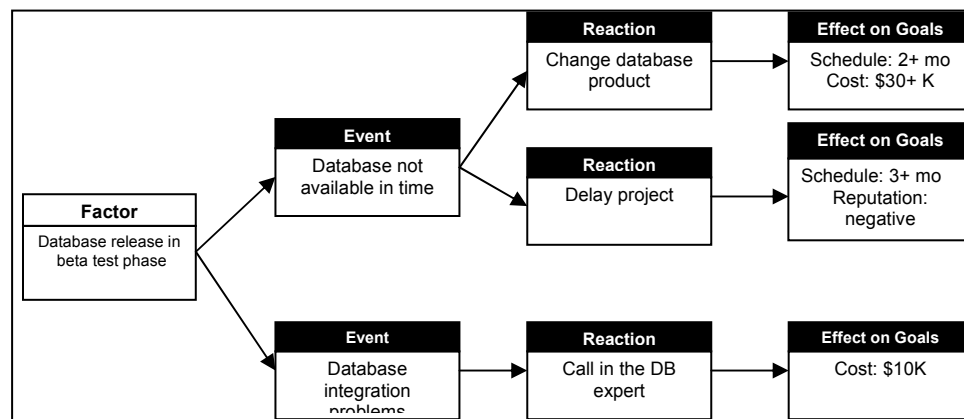


Figure 6 -Example of the Riskit analysis graph (risk scenarios) [Kontio et al. 1998]

2.4 Risk Management Framework (RMF)

Introduced by Gary McGraw, this risk analysis approach is embedded within a risk management framework [McGraw 2006]. He suggests three steps – attack resistance analysis, ambiguity analysis, and weakness analysis as the sub processes performed to accomplish risk analysis. Attack patterns and exploit graphs are explored for attack resistance analysis. Ambiguity analysis looks at the disparate viewpoints of analysts of a particular system, on how the system should work to determine whether disagreements are due to misunderstanding or the need for further analysis. [Viega and McGraw 2001].

Weakness analysis is performed by looking at the security of commonly used frameworks such as .NET and JEE and other third-party components.

2.5 Software Reliability

The area of software reliability engineering looks at the probability of failure-free software. [Rosenberg et al. 1998]. It is a mature area of software engineering with models and methods for assessing the reliability of software and, in some cases, predicting future-failures based on error-history [Karunanithi et al. 1992]. Software reliability engineering utilizes fault prevention, fault removal, fault tolerance and fault/failure forecasting as methods to provide system dependability. These terms are explained below [Lyu et al.1996]:

- Fault is a defect within software that causes a failure.
- Failure is the erroneous output as determined by the requirements for the system.
- Fault prevention is the avoidance of the occurrence of a fault.
- Fault removal is the usage of verification and validation to find and remove faults and
- Fault/failure forecasting is the estimation of the existence of faults and where they occur and the consequences of failure.

2.6 Traditional approach

A traditional approach to quantitative risk analysis is to measure risk with respect to financial loss or loss expectancy [GAO AIMD-00-33; Mkpong-Ruffin and Umphress 2007; Verdon and McGraw 2004; Steel et al. 2005]. This involves identifying all key risk elements and assigning estimates of values associated with each risk element, such as frequency, asset cost, potential loss value, business impact. With these values, estimates

of potential loss can be computed and an analysis of potential threats can be performed.

The Annual Loss Expectancy (ALE) can be computed as follows:

$$\text{ALE} = \text{Single Loss Expectancy (SLE)} \times \text{Annualized Rate of Occurrence (ARO)}$$

Where,

ARO is the frequency of threats per year,

SLE is the asset value x exposure factor (EF)

EF is the percentage of asset loss caused by the potential threat

Equation 1 - Traditional Annual Loss Expectancy

2.7 Cross-Disciplinary ties to Insurance

In actuarial science, actuarial tables (known also as mortality tables or life tables) exist to show the probability, at each age, of a person dying before their next birthday. With this as a starting point, statistics are derived on the probability of surviving any particular year of age. This process uses characteristics such as smoking, occupation and socio-economic class to distinguish between different risks in determining life expectancy. To develop predictions of future insured events, such as sickness, death, disability, etc., actuaries study incidence and severity of these events based on the recent past. They use this study to develop expectations about how the factors that motivated the events in the past will change over time. Armed with factors that affect risk, they are able to create models that allow them to evaluate risk [Klugman et al. 2004].

2.8 Problems with Existing Approaches

Most of the work done on risk management in software development has been from a subjective point of view [Boehm 2001; Farahmand 2003]. The main problem with qualitative approaches is the lack of objectivity, which makes it hard to duplicate results

or generalize assessments from previous projects [Voas et al. 1997]. The approaches enumerated above have the following limitations:

- Threat/Vulnerability Identification models and tools have as a goal the listing of risks that need addressing. These lists provide a starting point in dealing with security but it is not the case that all risks identified can be addressed. Even if all risks enumerated could be addressed, protecting against every single conceivable threat would reduce security to “a labor-some project that gets in the way of functionality ...” [Cheswick et al. 2003]. The removal of all enumerated risk would make it very difficult, if not impossible, to produce a system that would be useful as well as secure. Since all risks may not be addressable, determining which vulnerabilities to address becomes a problem.
- The neural network approaches have had disparate results as evidenced by the statements given by Sherer and Koshgoftar [Sherer 1995, Koshgoftar 1995]. Sherer’s conclusion was that neural networks did not do a good job of predicting the components that would have many faults. Koshgoftar, on the other hand, saw neural networks as effective modeling tool that should be seriously considered for software engineers because of its predictive capabilities.
- The testing approaches, though effective in measuring some classes of risks described, are used later in the development process, after most of the work for development has been accomplished. Any changes due to the analysis would be more expensive in terms of time and cost.
- The traditional quantitative approach of annualized loss expectancy (ALE) and the exposure factor (EF) carries with it the burden of knowing what asset costs to use.

The impact effect is measured by the cost of asset loss, which is not easily determined during the design phase of development. If the cost of asset loss is available, it may not adequately reflect the seriousness of the problem. This approach also suffers from the lack of sufficient data to determine probability of loss. Unlike the insurance or finance companies, software developers do not have probability loss tables available to adequately determine the frequency of occurrence and the probability of a risk occurring.

- Another reason given for not using quantitative models is the lack of access to reliable and current vulnerability data to support estimating probability of loss [GAO AIMD-00-33]. The data on the likelihood and costs associated with information security risk factors have been limited and the risk factors in this domain are constantly changing. This lack of reliable and current data has made it hard to determine which information security risks are the most significant. It is also difficult to compare controls to determine which ones are most effective. To this end, many organizations historically have leaned towards methods that do not rely on empirical data for obtaining reliable results.
- Risk assessment methodologies that are based on being able to accurately quantify reliability have also not been found to be good approaches because 100 percent reliability does not necessarily correlate to zero percent risk. Another problem with using reliability for liability prediction is that the reliability prediction models in existence are conflicting and make it almost impossible to know definitively the true reliability of a piece of software. Most of the error-history-based reliability models

predict reliability differently using the same data, which makes determining the model that is most accurate for a definite system hard to do.

- Relevant work on risk management in other disciplines has not been fully embraced by the software risk management community [Kontio 1999]. Since risk management is mature in other disciplines, knowledge can be gleaned from those disciplines to strengthen software risk management process.

Risk analysis is a process for considering possible risks and determining which are the most significant for any particular effort. Work has been done on risk identification, mitigation, evaluation and assessment, but not much has been done in assessing and ranking risks during the early part of the software development life cycle [Voas et al. 1997]. Determining which risk elements should be addressed and the optimum strategy for mitigating risks is often an intuitive process. Risk element assessment in software security is predominantly a qualitative process. An objective view of the risks inherent in a development effort requires a quantitative risk model.

3 RESEARCH DETAILS

A great need exists for tools to evaluate and quantify risks that are attributable to software development, if for no other reason than to reduce litigation costs due to software failure [Voas et al. 1997]. This research demonstrates that those attempting to perform risk assessment can do so from an objective point of view.

As previously noted, one of the hindrances to objective measures for risk assessment is the lack of available data. As shown in Figure 7, the rate of reported incidence of vulnerabilities as recorded in NVD has increased dramatically. This increase in available data provides a basis for building a database for determining and evaluating risks.

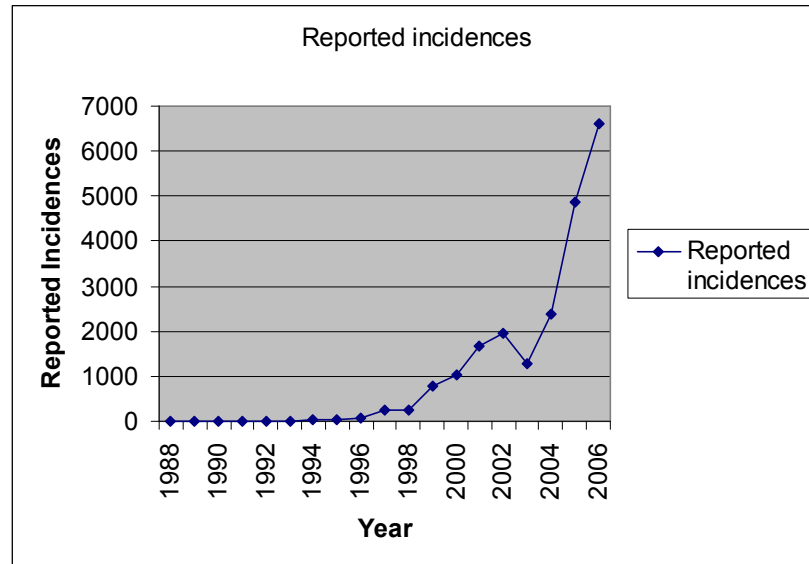


Figure 7 -Reported Incidences [NVD]

3.1 Conceptual Overview

To empirically assess the risks inherent in a development effort, using the approach applied in this research requires that the historical data be placed in a format that allows for classification. A clustering algorithm can then be applied to the data to determine the best segmentation of the underlying data as illustrated in Figure 8. This allows for new risk elements to be classified based on their similarities to the clusters discovered. Once the risk elements have been classified, the cluster that best represents the risk element in question is then used to determine the impact factor for that risk element. The impact factor value can then be used to calculate loss expectancy. This process of determining an empirical posture for a given development effort is illustrated in the sequence diagram of Figure 9.

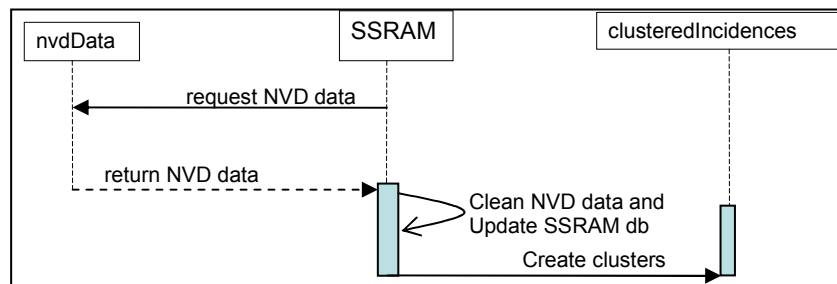


Figure 8 - SSRAM's Cluster Creation

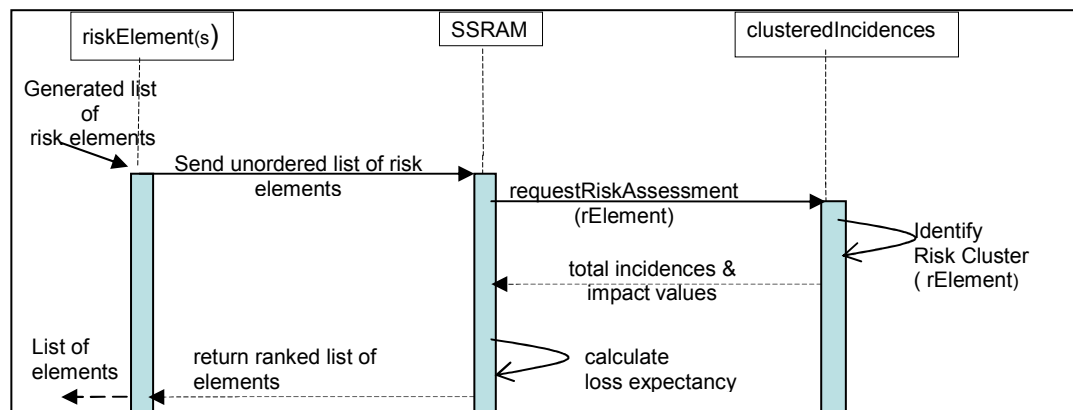


Figure 9 - Software Security Risk Assessment

This model provides an empirical assessment of the potential threats to a development effort and a ranking of these threats based on the risk metric calculation.

3.2 Data Preparation

To be able to assess risk elements, the historical data must be in a categorized form that allows new elements to be classified (Figure 8). To this end, historical data for this research was taken from an open-source vulnerability database provided by the U.S. government, the National Vulnerability Database (NVD). The NVD database provides RSS (Really Simple Syndication) feeds, in an XML format, of incidences reported from a variety of sources. The schema for this data is presented in Appendix A-1. It has seven sublevels with a nested hierarchy. An example of an entry is provided in Appendix A-2.

3.3 Parsing the Data

To parse the XML data, we had to traverse each node of the NVD XML document and used different types of SQL INSERT SELECT statements to accommodate the different ways that the XML feed reports node level data. An overview of these steps is given below, accompanied by the database diagram that depicts the tables generated based on the schema given.

- Insert the XML documents into a table

```
use ssram
INSERT nvdtmp
SELECT CONVERT(xml, BulkColumn, 2) FROM
    OPENROWSET(Bulk 'F:\Dissertation\Databases\National Vulnerability
Database\nvdcve-recent.xml', SINGLE_BLOB) [rowsetresults]
```

- Parse the data into the different tables based on the schema of the XML document. To parse the data into each table required that we implement different scripts for each table due to the differences in each of the tags of the XML document.

- Parsing first level tags – The first level tag `<entry>` has all of its values in attributes as shown below in Figure 10. To parse this data, an attribute centric approach was applied to the data in the first level node,

```
<entry
  type="CVE"
  name="CVE-2006-0948"
  seq="2006-0948"
  published="2006-08-21"
  modified="2006-08-22"
  severity="High"
  CVSS_score="7.0"
  CVSS_vector="(AV:L/AC:L/Au:NR/C:C/I:C/A:C/B:N)">
```

Figure 10 - NVD's First Level Node

- Parsing lower level tags with additional tag and attribute values – The `<desc>`, `<ref>` and `<sols>` tags use a combination of tag and attribute values that need be parsed independently. For example, Figure 11 shows a `<desc>` tag that has the nested `<descript>` tag and the `source` attribute. The tag and attribute have to be parsed separately. Also, since these are nested tags, parsing involved getting information from the parent tag along with information from the lower level tag.

```
<desc>
  <descript source="cve"> AOL 9.0 Security Edition revision 4184.2340, and probably
  other versions, uses insecure permissions (Everyone/Full Control) for the "America
  Online 9.0" directory, which allows local users to gain privileges by replacing
  critical files.
  </descript>
</desc>
```

Figure 11- NVD Description Node

- Parsing lower level tags with nested lower level tags –
The `<loss_types>`, `<vuln_types>` and `<range>` tag entries have nested lower-level tags that contain the data needed. If the value is not present, then the tag is not available for that particular instance. For example Figure 13 shows two instances of the `<loss_types>` tag. Note that Figure 12(a), has the `<int />` tag which is not in Figure 12(b)'s tag and Figure 12(b) has the `<sec_prot>` tag and its associated attribute which is not in Figure 12(a). These values are embedded as part of the

meta property of the tag. The meta property data for these tags needed to be accessed to accurately represent their information.

<loss_types> <avail /> <int /> </loss_types> (a)	<loss_types> <avail /> <sec_prot other="1" /> </loss_types> (b)
---	--

Figure 12 - Node with nested lower level tags

Appendix A-3 has the full script for parsing the NVD XML document into required tables. Successful parsing of the data resulted in the population of the SSRAM database based on the schema depicted in Figure 13. Each entry level tag is depicted as a record in the entry table with a unique entry name that is used to associate with the lower level tags in the other tables shown in Figure 13.

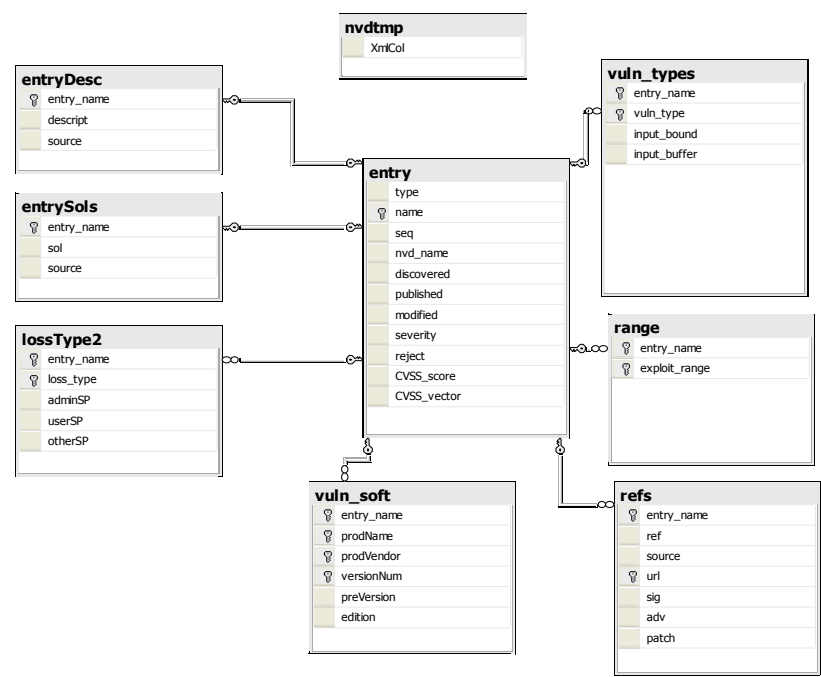


Figure 13 - SSRAM Schema

3.3.1 Validation of Data Entry

To validate that the data was gotten correctly from the XML files, a random sample of entries was chosen. [Bartlett II et al] recommends a sample size that is less than or equal

to 5% of the captive population. Given that the NVD data is captive data, in that all the data requested is in the file gotten, a file with 136 entries was downloaded on 8/26/06 and used to test the process for data entry. A systematic random sampling approach was implemented to test the data. With systematic random sampling, every k -th entry is chosen, with the first entry randomly selected. Since this test file had 136 entries, we chose $\lceil 5\% \times 136 \rceil = 7$ entries to be compared with the actual XML data to see if there were any discrepancies and chose $\lceil 136 \div 7 \rceil = 20$ as the k factor. To this end, a random number, between 1 and 19 based on a random generator, was chosen to determine the first entry to be validated. Thereafter, every 20th entry was chosen for validation. The following records were checked - the 5th, 25, 45, 65, 85, 105, 125 and, for added measure, the first and last entries. We compared the contents of the actual XML document with the content of the uploaded table to ascertain that they had the same information (Table 20 in Appendix A-4). Any discrepancies between the table and the actual record information were corrected and the data uploaded again. We again compared the contents of the first sample of data and then using the same approach, selected seven other entries (2, 22, 42, 62, 82, 102, 122) to confirm that the data had been correctly uploaded. See Appendix A-4 for complete data upload validation information.

3.4 Cluster Determination

With the data parsed into the component tables, we were able to look at the fields necessary for clustering the data. Clustering the data allows for determining the different kinds of natural groupings or classes that may exist within the underlying data. Given the nature of the data collected, we chose fields within each table that were independent as

fields necessary for clustering. For example, in the *entry* table, we did not choose the *severity* field, as it depends on the *CVSS_score* field. Along with the scalar fields, we also chose the *description* text field as it contained information about the vulnerability entry that is cogent in determining the kind of vulnerabilities and losses affected. Figure 14 shows the select statement of the fields chosen to determine the clusters for the elements reported.

```
select entry.name, entry.discovered,entry.published, entry.cvss_score,
      lossType2.loss_type, lossType2.adminSP,lossType2.userSP,
      lossType2.otherSP,vuln_types.vuln_type, vuln_types.input_bound,
      vuln_types.input_buffer,range.exploit_range, refs.source,
      refs.sig, refs.adv, refs.patch,vuln_soft.prodName,
      vuln_soft.versionNum, vuln_soft.preVersion, vuln_soft.edition
into denormNVD
from entry, lossType2, vuln_types, range, refs,vuln_soft
where
      entry.name = lossType2.entry_name AND
      entry.name = vuln_types.entry_name AND
      entry.name = range.entry_name AND
      entry.name = refs.entry_name AND
      entry.name = vuln_soft.entry_name
```

Figure 14 - Independent Variable Selection Statement

Since the description field contained ‘comment-like’ data that cannot be clustered in the form given, it was necessary to create a table to which the nouns and noun-phrases of the description field were extracted. With each term an associated score Term Frequency and Inverse Document Frequency (TFIDF) was assigned. This score is defined as:

$$TFIDF \text{ of a Term } T = (\text{frequency of } T) * \log(\#rows \text{ in Input} / (\#rows \text{ having } T)).$$

Equation 2 - TFDIF

Upon completion, a vector table that associates each term with the description entry was also created. (Appendix B-1)

Determining the clusters to use to classify risk elements required the exploration of different segmentation schemes. Some clustering algorithms produce well-separated clusters, such that an object can belong to only one cluster. Others use an overlapping

approach so that an object could belong to more than one cluster. To identify the natural groupings that may exist in the historical data, we examined two clustering algorithms that implemented each of the segmentation schemes described above, K-Means and Expected Maximization (EM).

K-Means is a clustering algorithm that produces k partitions or clusters of the underlying data by trying to find the centers of natural clusters in the data. This is achieved by minimizing the total variance within the cluster [Tang and MacLennan 2005]. Using this approach, the K-Means algorithm assigns each data record to a specific cluster. This approach does not allow for uncertainty in the membership record within a cluster.

The EM clustering algorithm, like K-Means, produces k partitions by finding the center of the natural clusters in the data, but unlike the K-means, relaxes the assumption that a data record has to belong to only one cluster. It allows for the overlapping of clusters, in that data can belong to other clusters but with different probabilities of membership [Bradley, Fayyad, Reina 1998].

Two mining structures were developed, one for the data without the description terms and the other with the vector table included as a nested table. Within each mining structure, mining models using EM and KM algorithms with different target numbers for clusters were generated. These models were then compared for predictive accuracy. We found that the EM mining models with the description vector table showed better predictive accuracy than those without the description vector tables. Figure 15 and Figure 16 show the mining accuracy charts for the different structures. The X-axis of each chart represents the percentage of the test dataset that is used to compare the predictions, while

the Y-axis represents the percentage of predictions that are correct. In both cases, the EM models outperformed the KM models. As shown in Figure 15 and Figure 16, the structure with models that took into account the description vector table had better prediction values than the mining structure without the vector table. Given 100% of the test cases used for prediction, the best predictive model for the mining structure with the description vector table (NVDCL-EM10) had a 71.86 % predictive accuracy score as compared to 61.25% predictive accuracy of the best model for the mining structure without the vector table (NVDCLwo_EM10).

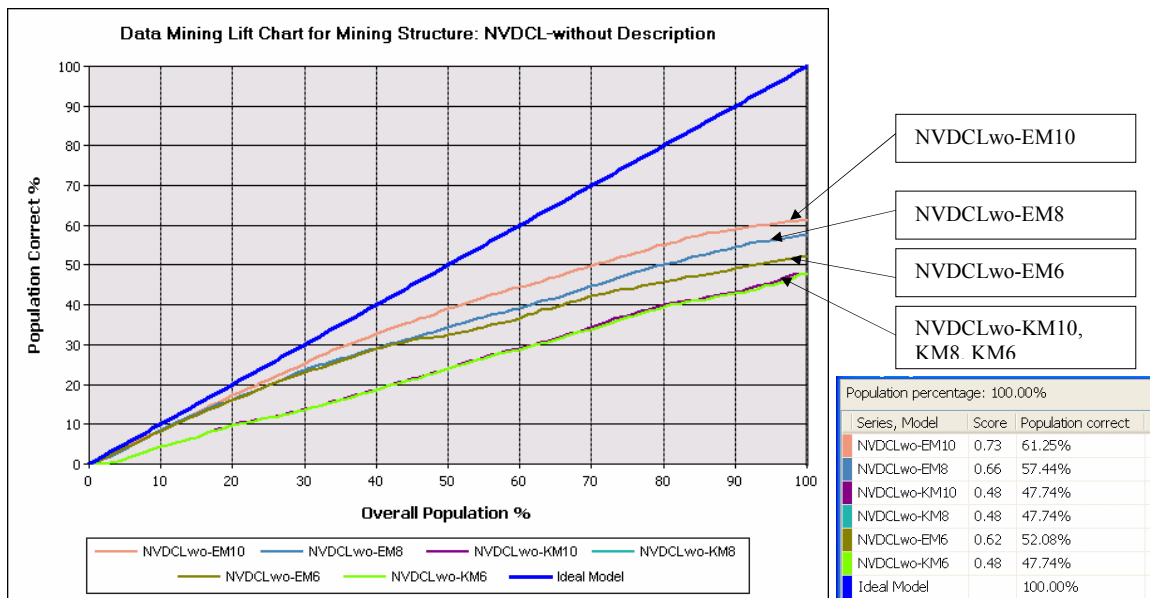


Figure 15 -Clustering without Description

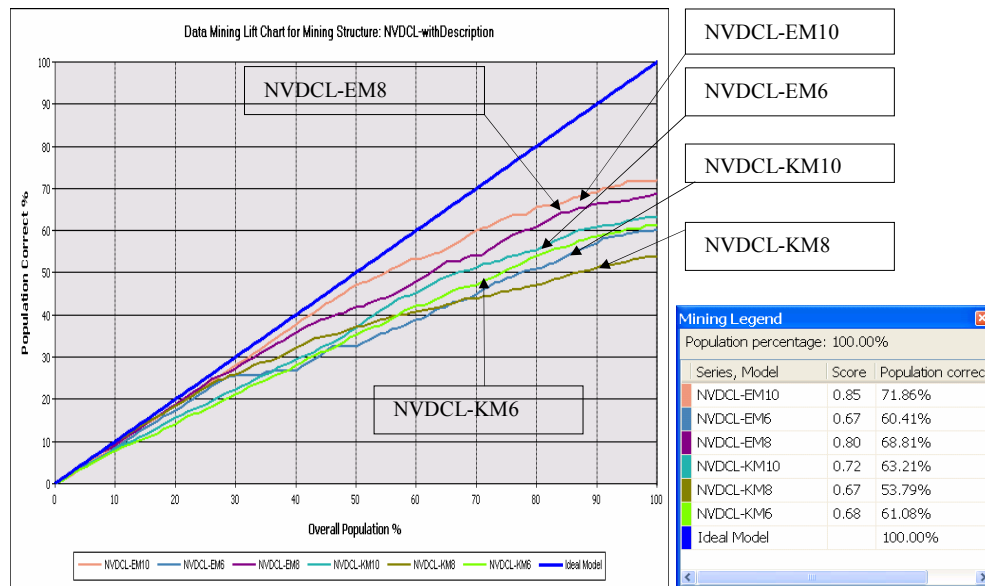


Figure 16 - Clustering with Description

Choosing to use the mining structures with the description vector, we decided to use the EM clustering algorithm to segment our training data since it predominantly outperformed the K-Means algorithms, regardless of the mining structure used, as shown in the graphs in Figure 15 and Figure 16. To determine the cluster node size to use, we looked at predictive results obtained from mining structures with 4, 6, 8 and 10 cluster node sizes. We decided on using node size of four based on the heuristic determination of the clustering algorithm. We then incremented the node size by two, up to the default node size of ten for the different mining models and compared the predictive results obtained with each model. Upon completion we chose the EM clustering algorithm based on 10 nodes because the EM 10-Node size outperformed the other node sizes across most of the population. See Figure 18.

With the clusters formed, each data item of the training model was assigned to a cluster. This was done by extracting all the data necessary for the prediction query along with the

cluster tag associated with each entry and saving this data so that prediction algorithms would be run against it. Effectively, a query such as that is shown in Figure 17.

```
SELECT
  (t.[name]), CLUSTER() FROM [NVDCL-EM-10]
PREDICTION JOIN
OPENQUERY([SSRAM],...) AS t
ON
  [NVDCL-EM-10].[Cvss Score] = t.[cvss_score] AND ...
```

Figure 17 - Cluster Assignment Query Example

3.4.1 Validation of Clustering

We separated the data from NVD into two parts so that we could use one part for training and the other for testing our algorithm. After training the algorithm, we used two approaches to validate the clustering algorithms. In the first approach, we used the test data to find an indication of how likely it is for each test case to exist within a determined cluster. A score for the model as shown in the mining legend table to the right of the graph in Figure 18 reflects the average case likelihood of each of the training cases existing within each model. A score closer to 1 is an indication that the training points are close to the clusters in the model and, as such, implies that the clusters are compact and well-defined. A score that is close to 0 implies that the training data points are scattered and the clusters are not as well defined [Tang and MacLennan 2005].

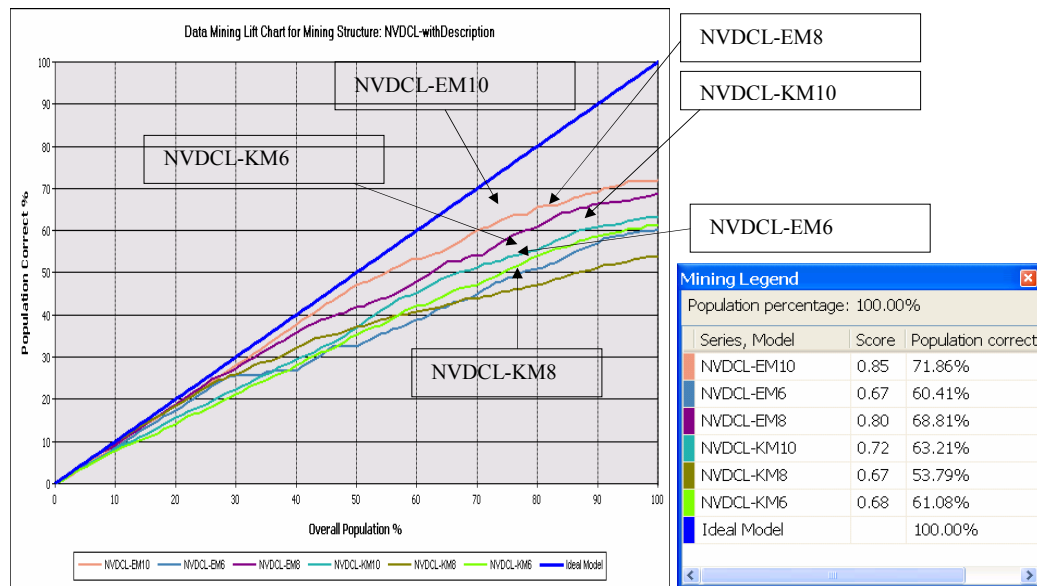


Figure 18 -Mining Accuracy Chart

Also, we examined the cohesion of the cluster with respect to the centroid of the cluster by looking at the mean, standard deviation, maximum and minimum values for each of the clusters within each mining model to. In essence these statistics were used to reflect the proximity of the entries within each cluster to the mean. An example of the results gotten is shown in the Table 5. For the algorithm represented here, seven clusters were derived and the count shows the number of entries within the cluster. With the smallest number of entries in a node being 300, we assume a normal distribution for our data. The average CVSS score and the standard deviation values reflect the fact that though the range of scores in each of the clusters ranged from 1.9 – 10, 96.3% of the data within the clusters were within 1.95 standard deviation from the mean, implying a tight cohesion. Cluster 4 had the least cohesion with a standard deviation of 2.27, while Cluster 7 had the tightest cohesion with entries that were exactly the same, as reflected in its standard deviation and range of 0.

Cluster	Count	Std Deviation	Avg Score	Max Score	Min Score
1	1700	1.61	3.8	10	1.9
2	8351	0.81	6.74	10	1.9
3	8761	1.34	9.29	10	1.6
4	860	2.27	4.41	10	1.9
5	429	1.91	4.45	10	2.3
6	2990	1.54	3.28	10	2.3
7	300	0	7	7	7

Table 5 - Cluster Algorithm Cohesion Factors (NVDCL-EM10)

Cluster	Count	Std Deviation	Avg Score	Max Score	Min Score
1	22717	2.08	7.44	10	1.6
2	7816	2.2	5.43	10	1.9
3	7296	2.19	5.16	10	1.9
4	3856	2.39	5.01	10	2.3
5	9306	2.33	4.75	10	2.3
6	514	0	7	7	7

Table 6-Cluster Algorithm Cohesion Factors (NVDCL-EM8)

Looking at the cohesion values for the mining structures in Table 5 and Table 6, it is evident that the clusters formed by the mining model NVDCL-EM10 exhibited tighter cohesion values than that of NVDCL-EM8. All but one of the clusters in Table 5 have has data within 1.95 standard deviations from the mean. On the other hand, NVDCL-EM8 has only one cluster with data within 1.95 standard deviations from the mean (Table 6). Given that NVDCL-EM10 (Table 5) had the tightest cohesion factors, we chose this algorithm for clustering the historical data so that new data can be classified against it. See Appendix B-5 for complete validation results.

3.5 Data Classification

Determining the general categories of the vulnerability database allows for classifying new data. Classification is the process of assigning objects to one of several predefined categories. To find the cluster most similar to new data given, we examined Decision

Tree, Neural Networks and Naïve Bayes classification algorithms and chose the classification algorithm that best predicted the cluster that should be used.

Decision Tree algorithm: The decision tree algorithm is a classification algorithm for predictive modeling of continuous and discrete attributes. It uses the relationships between the input columns within the data set to make predictions. It does this by identifying the input columns that correlate with the predictable columns. Each time an input column is found to be significantly correlated with the predictable column, the node is added [Microsoft 2008-3].

Neural Networks: Neural network algorithms stem from the 1940s research work by Warren McCulloch and Walter Pits on simulating how the brain works [Tang and MacLennan 2005]. Neural network algorithms address primarily classification and regression tasks of data mining. Like decision trees, they find non-linear relationships between input attributes, but unlike decision trees, they find smooth as opposed to discontinuous nonlinearities. Neural network algorithms use networks that are made up of three types of nodes or layers (input, hidden and output) and directed edges that show the data flow during the prediction process. The input nodes, which form the first layer of the network, are mapped to the input attributes. The hidden nodes, which are the nodes in intermediate layer, combine the values gotten from previous layers with weights of associated edges to perform some calculations and generate the result value to the next layer. The output layer represents the predictable attribute(s) [Microsoft 2008-4].

Naïve Bayes: The Naïve Bayes algorithm, like decision tree and neural network algorithms, is a classification algorithm for predictive modeling. It works by calculating the conditional probability of input and predictive attributes. Naïve Bayes assumes that

the input attributes are independent and as such does not take into account any dependencies that may exist between attributes. In doing so, the probabilities can be multiplied so that the likelihood of each state can be determined [Microsoft 2008-5, Tang and MacLennan 2005].

To help in determining the model to be used for classification, two mining structures were created, one using the vector table with descriptive information and one without. A comparison of the classification algorithms explained above was done. Figure 19 and Figure 20 pictorially depict the result of classifying the test data with and without the description vector. The Naïve Bayes and Neural Network algorithm outperformed the Decision Tree algorithm in each instance. Though the scores for Neural and Naïve Bayes were the same (Table 7), the Naïve Bayes algorithm took less time (0:00:03 versus 0:00:32) to process the same data.

Model Name	Processing Duration Time	Score
NBClusterClassifier (Naïve Bayes)	3 seconds	0.81
DTClusterClassifier (Decision Tree Classifier)	5 seconds	0.78
NNClusterClassifier (Neural Network Classifier)	32 seconds	0.81

Table 7 - Classifier Processing Time and Score – Cluster Classifier without Description

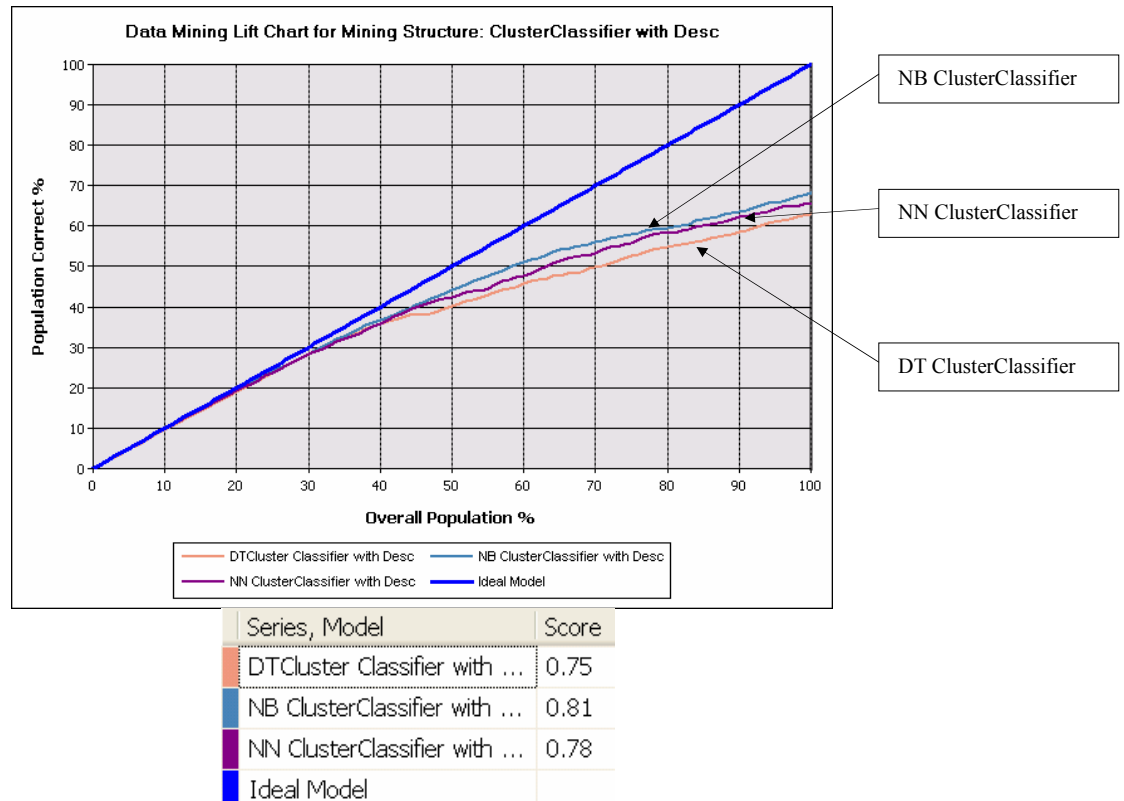


Figure 19 -Classifiers with Description Vector

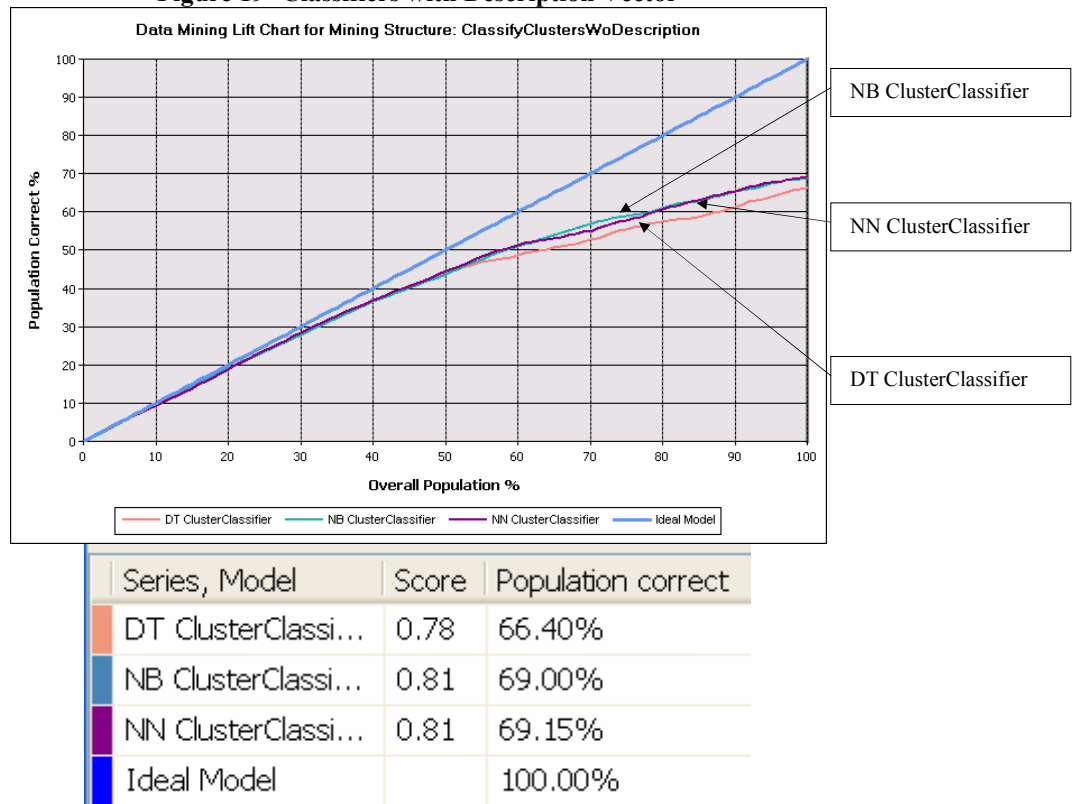


Figure 20 - Classifiers without Description Vector

3.5.1 Data Classification Validation

Classification models are evaluated based on the number of test records that they correctly and incorrectly predict. An example of this evaluation is shown in Table 8. Columns correspond to the actual values and rows depict the predicted values. The complete matrix can be seen in Appendix B-4. Table 8 shows the confusion Matrix for the Naïve Bayes model, the first row should be read as follows: Row 1 depicts the entries that were predicted for Cluster 1. Of all the entries predicted for Cluster 1, 3158 actually belonged to Cluster 1, 496 belonged to Cluster 6 and 157 entries predicted as Cluster 1 were in actuality Cluster 2's entries. This matrix provides enough information to measure the performance of each model using the Accuracy metric defined as follows:

$$Accuracy = \frac{\text{Number of correct Predictions}}{\text{Total Number of Predictions}}$$

Equation 3 – Accuracy Determination

Counts for NB ClusterClassifier on [Cluster Node]									
Predicted	Cluster 1 (Actual)	Cluster 6 (Actual)	Cluster 4 (Actual)	Cluster 3 (Actual)	Cluster 8 (Actual)	Cluster 5 (Actual)	Cluster 7 (Actual)	Cluster 2 (Actual)	Total Predicted
Cluster 1	3158	496	60	145	74	164	41	157	4295
Cluster 6	307	680	1	89	0	63	2	29	1171
Cluster 4	35	14	257	5	75	32	197	6	621
Cluster 3	19	129	10	513	0	7	4	103	785
Cluster 8	2	0	9	0	17	0	15	6	49
Cluster 5	47	89	6	15	2	1017	7	59	1242
Cluster 7	28	2	226	8	202	2	578	22	1068
Cluster 2	11	21	6	221	10	27	15	628	939
Accuracy=0.67									

Table 8 - Classification Confusion Matrix – Classifier without Description

Figure 21 shows a comparison of the accuracy performance metric for the classifiers in both mining structures. The mining structure without the description vector tables had an average accuracy score of 0.66 while the structure without the description vector had an average accuracy score of 0.79. Though the description vector was necessary in

determining cluster, it was not necessary to have that information in classifying a new entry.

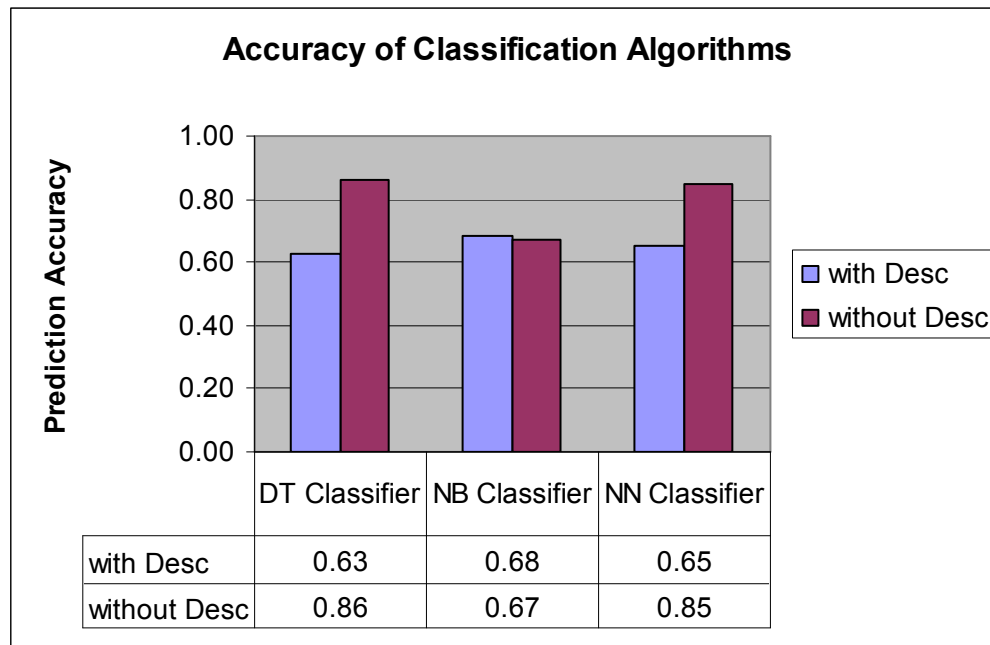


Figure 21 - Prediction Accuracy of Classifiers

3.6 Loss Expectancy Determination

With the data segmented into clusters and a classification algorithm adopted, the impact score and frequency of occurrence of a given risk entry was determined and used to calculate loss expectancy. Traditionally, loss expectancy is computed by determining the value of the percentage of asset loss caused by the potential threat and finding the frequency of occurrence of threats within a given year, as shown in Equation 1.

Since asset value loss cannot be easily determined, we opted to use the Common Vulnerability Scoring System (CVSS) ‘base’ score because it shows the inherent characteristics of the vulnerability incidence reported and reflects the impact value of the reported incident.

```
<entry
  type="CVE"
  name="CVE-2006-0948"
  seq="2006-0948"
  published="2006-08-21"
  modified="2006-08-22"
  severity="High"
  CVSS_score="7.0"
  CVSS_vector="(AV:L/AC:L/Au:NR/C:C/I:C/A:C/B:N)">
```

Figure 22 - CVSS base Score and Vector

Each NVD entry has along with the CVSS base score, a vector that reflects how the base score was calculated. The vector, as shown in Figure 22, is used to determine the base score for a particular entry. In Figure 22, AV: Access Vector = L (Local accessibility) intimates that this vulnerability can only be exploited locally; AC: Access Complexity = L (Low) indicates that an attack based on this vulnerability would not require special access conditions, as it could be performed manually. It would not necessarily require much skill or a lot of additional information for its execution. Au: Authentication = NR (Not Required) shows that authentication is not required to trigger this vulnerability. The impact metrics of C: Confidentiality, I: Integrity and A: Availability all equaling C (Complete) imply a total system compromise if vulnerability is exercised. Based on the NVD algorithm for computing base score [First 2005], the resulting base score of 7.0 was calculated as shown in (Appendix B-7).

3.6.1 Determining Fields to Predict CVSS Score

The data mining wizard, supplied with Microsoft's Visual Studio 2005, suggests fields that would likely give information that would lead to the selected output using entropy-based analysis [Tang and MacLennan 2005]. This analysis looks at the contribution of attribute values to predicting a particular attribute. In our case, the analysis looked at the contribution of all of the attributes in predicting CVSS score. Those attributes with an

entropy value of 0 are evaluated as being able to contribute nothing to the prediction decision. Those attributes with entropy values of 1 are so distinct that in effect, each record would be a partition of its own, thereby making it too small to make any reliable prediction.

Columns related to cvss_score:		
Column Name	Score	Input
ClusterNode	0.363	x
loss_type	0.263	x
exploit_range	0.174	x
adminSP	0.172	x
vuln_type	0.076	x
input_buffer	0.045	
otherSP	0.042	
userSP	0.024	
patch	0.009	
input_bound	0.007	
adv	0.007	
preVersion	0.003	
sig	0.002	
edition	0.000	
ref	0.000	
discovered	0.000	
ClusterProbability		
versionNum		
prodName		
source		
published		

Figure 23 - Suggested Input Fields

This suggestion provided a basis for the fields that could be used in determining the parameters for predicting impact. Since the scores provide values between 0 and 1, we chose to remove those attributes whose entropy scores were less than 0.01, since the level of contribution was so close to zero implying very little contribution, and those whose entropy values were very close to 1 were also removed as they would not help in making reliable prediction. Figure 23 depicts all the attributes for the model, their associated entropy score and the Input column that identifies those fields suggested for predicting CVSS score. Although Administrative level security protection (adminSP) was suggested as an input attribute, we did not include it in our model since its value is dependent on a specific loss type. Those attributes grayed out in Figure 23 are those with entropy values very close to 1. We chose Cluster Node, Loss Type, Vulnerability Type

and Exploit Range as the fields for predicting CVSS score. To validate the choice of fields for predicting impact factors, we performed two multiple regression analysis test, one between cluster node and loss type, vulnerability type and exploit range (Figure 24) and the other between CVSS Score and loss type, vulnerability type and exploit range (Figure 25). We performed the multiple regression analysis against cluster node (Figure 24), to ascertain whether loss type, vulnerability type and exploit range could be used to determine a specific cluster for a risk element since our algorithm requires us to classify a risk element to a specific cluster (Figure 9). The determination of the CVSS score is also based on the cluster node chosen; as such, the second multiple regression analysis was conducted to examine whether loss type, vulnerability type, and exploit range were variables that could be used to predict CVSS score (Figure 25).

3.6.1.1 Regression Analysis: Cluster Node versus Loss Type, Vulnerability Type and Exploit Range

The result of the multiple regression analysis, as shown in Figure 24, implies that loss type, vulnerability type and exploit range are significantly related to cluster node. The coefficient of determination (R^2) and the adjusted R^2 of 27.9% show that 27.9% of variations in the dependent variable (Cluster Node) can be explained by the independent variables (Loss Type, Vulnerability Type and Exploit Range). This indicates that Loss Type, Vulnerability Type and Exploit Range can be used to determine the cluster node to which a risk element could belong. Given the lack of difference between the R^2 and the adjusted R^2 , we can intimate that all the variables chosen to explain variation of cluster nodes are necessary (see Table 9 for interpretation of other Regression statistics).

**Regression Analysis:
Cluster Node versus Loss Type, Vulnerability Type and Exploit Range**

<i>Regression Statistics</i>	
Multiple R	0.528
R Square	0.279
Adjusted R Square	0.279
Standard Error	1.992
Observations	13140

<i>ANOVA</i>					
	<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>
Regression	3	20169.732	6723.244	1693.694	0
Residual	13136	52144.324	3.970		
Total	13139	72314.056			

	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>
Intercept	8.070	0.084	96.497	0
LossType	-0.927	0.014	-65.714	0
VulnType	-0.279	0.009	-31.171	7.5E-206
ExploitRange	-0.337	0.034	-10.020	1.52E-23

Figure 24 - Multiple Regression Result (Cluster Node)

The F test statistic (F) and its corresponding p-value (Significance F) indicate an overall goodness of fit for the model. The p-value (0) is considered highly significant as it is less than 1 %. The F-test statistic of 1693.694 shows that the ratio of explained variation (6723.244) to unexplained variation (3.970) is very large but more so, the significance value ($p < 0.001$) of the F-test statistic allows us to reject the null hypothesis that there is no significant relationship between Cluster Node and the independent variables, Loss Type, Vulnerability Type and Exploit Range. We also looked at the influence of Loss Type, Vulnerability Type and Exploit Range on Cluster Node and as can be seen by the results of the test statistics for LossType, VulnType and ExploitRange, along with their

P-values ($p < .0001$), the three independent variables have a significant relation with Cluster Node and as such can be used to classify a risk element entry to a cluster.

Regression Statistics		Interpretation
Multiple R	0.528	R= Coefficient of Multiple Correlation = the positive square root of R-squared
R Square	0.279 = 27.9%	R-squared = Coefficient of Multiple Determination = percent of the variation in the cluster node (dependent variable) that is explained by the x-variables (LossType, VulnType, ExploitRange) and the model
Adjusted R Square	0.279	R-squared adjusted = version of R-squared that has been adjusted for the number of predictors in the model. R-squared tends to over estimate the strength of the association, especially when more than one independent variable is included in the model.
Standard Error	1.992	S = Standard Error = Standard Error of the Estimate = average squared difference of the error in the actual to the predicted values.
Observations	13140	Number of observations in the sample.

Table 9 - Regression Statistic Interpretation

3.6.1.2 Regression Analysis: CVSS Score versus Loss Type, Vulnerability Type and Exploit Range

Regression Analysis: CVSS Score versus Loss Type, Vulnerability Type and Exploit Range					
<i>Regression Statistics</i>					
Multiple R	0.593				
R Square	0.352				
Adjusted R Square	0.352				
Standard Error	1.943				
Observations	13140				
ANOVA					
	<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>
Regression	3	26962.24325	8987.414	2380.023	0
Residual	13136	49604.01067	3.776188		
Total	13139	76566.25392			
	<i>Standard</i>				
	<i>Coefficients</i>	<i>Error</i>	<i>t Stat</i>	<i>P-value</i>	
Intercept	0.916	0.082	11.224	4.2E-29	
LossType	1.147	0.014	83.300	0	
VulnType	0.108	0.009	12.335	9.15E-35	
ExploitRange	0.774	0.033	23.583	1.8E-120	

Figure 25 - Multiple Regression Result (CVSS score)

The result of the analysis to see if the parameters, loss type, vulnerability type and exploit range, would aid in predicting CVSS score is summarized in Figure 25. The coefficient of determination, R^2 and the adjusted $R^2 = .352$, showed that 35.2% of the variation of CVSS score could be explained by the independent variables, loss type, vulnerability type and exploit range. Also, the F-test statistic of 2380.023 with P-value ($p < .001$) showed that there is a significant relationship between the dependent variable CVSS score and the independent variables in question. The P-values for t-test statistic for LossType, VulnType and ExploitRange are all ($p < .0001$), also show that the independent variables (LossType, VulnType and ExploitRange) have a significant linear relationship with the dependent variable CVSS score.

3.6.2 Determining Loss Expectancy

The result of the multiple regression analysis motivated us to use the related data from a classified cluster to determine the predicted impact score and the predicted frequency of occurrence. We used the predicted impact score in the same manner *asset loss x exposure factor* is used in the traditional approach, to determine single loss expectancy. Given the predicted impact score, frequency of occurrence of incidence, and growth rate for each of these attributes, the loss expectancy for a specified time was calculated as follows:

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

Equation 4 – Predicted Loss Expectancy (PLE)

Using the growth rate of the CVSS score and the average CVSS score, the predicted impact value was calculated as follows for each test entry:

$$\text{Predicted Impact Score} = \bar{x}_{CVSS \text{ Score} (period)} \times (1 + \mu_{Growth \text{ Rate}(period)})$$

where

period covers the evaluation time given,

\bar{x} = the weighted mean, with the weight being the number of incidences reported

μ = the arithmetic mean

Equation 5 - Predicted Impact Score (PIS)

To find the weighted mean CVSS scores for the period ($\bar{x}_{CVSS\ Score(period)}$), we used the number of incidences reported for each month as the weighting factor to obtain the weighted mean. We used the relative weights of the number of incidences for each month as coefficients to express the weighted mean in a linear form. The coefficients α_i were determined by dividing the number of incidences reported per month by the total number of incidences reported for the time period under evaluation.

$$\bar{x}_{CVSS\ Score(period)} = \alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_n x_n$$

where

the real numbers α_i satisfy $\alpha_i \geq 0$ and $\alpha_1 + \alpha_2 + \dots + \alpha_n = 1$ and

α_i = number of incidence coefficient for the month and

x_i = average CVSS scores for the month

Equation 6 - Weighted Average CVSS Score

The CVSS score and frequency of occurrence growth rates were calculated as follows:

$$\left(\frac{\text{current Month's Avg Value} - \text{previous Month's Avg Value}}{\text{current Month's Avg Value}} \right)$$

Equation 7 - Growth Rate

3.6.2.1 Validation of Loss Expectancy Calculations

To make sure that the calculations in our algorithm were correct, we generated a query to select certain sample data from the input set (Figure 26).

```
select datediff(month, published, '01/01/2002') as  
timeLag, avg(cvss_score) as avgScore, count(distinct name) as  
reportedEntries  
from trainingdataawclusters96_01 where  
    (datediff(month, published, '01/01/2002') <= 12) and  
    (datediff(month, published, '01/01/2002') > 0 ) and  
    clusternode = 'Cluster 1'  
group by datediff(month, published, '01/01/2002')  
order by timeLag
```

Figure 26-Query for Sample data to validate Calculation

We used Excel to calculate the values in question (Table 10), and then compared the results to those gotten from the actual running of the stored procedure used within SSRAM (Figure 27). See Appendix C-1 for complete listing of the stored procedure used to calculate different values necessary for computation of loss expectancy. The coefficients for the reported entries were calculated for each month by dividing the number of entries for that month by the total number of entries. For example, in Table 10, the coefficient for Dec-01 (0.15) was calculated by dividing 94 by 630. The weighted average score was then calculated using Equation 6. The growth rate for the score and frequency of occurrence were calculated using Equation 7 and the total number of incidences reported was determined by adding the reported entries for each month. These values as noted in Table 10 and Figure 27 are the same and validates that the algorithm works as expected.

TimeLag	12 months prior to 1/1/2002	Reported Entries	Avg Score	Coefficient of Reported Entries	Score Growth	Frequency Growth
1	Dec-01	94	7.58	0.15	-0.11	-0.72
2	Nov-01	26	6.77	0.04	0.06	0.96
3	Oct-01	51	7.16	0.08	-0.03	-0.10
4	Sep-01	46	6.97	0.07	0.23	0.83
5	Aug-01	84	8.57	0.13	-0.07	-0.04
6	Jul-01	81	8.01	0.13	0.09	-0.12
7	Jun-01	71	8.72	0.11	-0.05	-0.39
8	May-01	43	8.32	0.07	-0.33	-0.91
9	Apr-01	4	5.57	0.01	0.59	7.25
10	Mar-01	33	8.86	0.05	0.00	0.48
11	Feb-01	49	8.87	0.08	-0.24	-0.02
12	Jan-01	48	6.77	0.08		
Total		630	WeightedAvg	7.93	0.0143	0.6564

Table 10 – Calculation of Impact Factors (Excel Result)

TimeLag	AvgScores	reportedEntries
1	7.5792	94
2	6.7664	26
3	7.1594	51
4	6.9651	46
5	8.5697	84
6	8.0072	81
7	8.7243	71
8	8.3226	43
9	5.5654	4
10	8.8574	33
11	8.8693	49
12	6.7701	48

TotalEntries	1	630
weightedAvg	1	7.93
avgScoreGrowth	1	0.0143
avgFreqGrowth	1	0.6564

Figure 27 - Calculation of Impact Factors Result (Stored Procedure)

To ascertain that the result of an impact and loss expectation prediction could be made, we created an application to determine loss expectancy for a single entry (Appendix D-1). We chose loss type, vulnerability type, and exploit range values and used the information to classify the entry to a cluster node from which the impact factors were calculated. For our case, we chose input as the vulnerability type, security protection as the loss type and exploitation range of local. This selection was classified as a Cluster 1 entry and the

calculations for impact were computed (Figure 28). Note that these values are also consistent with the values shown in Figure 27 and Table 10 for Cluster 1 data.

The 'Singleton Request' window displays the following configuration and results:

Vulnerable Type	PredictDate
input	1/ 1/2002
Loss Type	Periods (Months)
sec_prot	12
Exploit Range	
local	

Vuln Type	Loss Type	Exploit Range	Cluster Node	PredictProbability
input	sec_prot	local	Cluster 1	0.9838

Total Entries: 630 Weighted Avg: 7.93 Score Growth: 0.01 Occurrence Growth: 0.66

Predicted Impact: 8.0093
Predicted Occurrence: 1046

Buttons: Calc, Exit

Figure 28 -Single Entry Impact Estimation

3.6.3 Validation of Loss Expectancy

The goal of this research is to provide an empirical assessment of risk elements based on historical data and provide a prioritized list of the risk elements based on the empirical estimations generated. To this end we have chosen the historical data validation approach [Sargent 2003] for validating our research.

NVD's data was divided into two parts, the training and testing data. The data from 1996 - 2001 was used to build the impact prediction model and tested against the impact values for January 2002 as shown in Figure 31 (Appendix D-2 for code). In essence the test data was loaded as vulnerability entries as shown in Figure 29; each entry was then classified to a cluster and the data from that cluster used to predict the frequency of occurrence and the impact value for each entry (Figure 30). The mean values from the actual data were compared to the predicted scores from the model (see Appendix D-3 for the data) using t-Test comparison of means (Table 11).

Vulnerability Entries				
	EntryID	VulnType	LossType	ExploitRange
▶	CVE-1999-1081	design	conf	remote
	CVE-1999-1091	access	int	local
	CVE-1999-1091	access	int	remote
	CVE-2001-0887	design	int	local
	CVE-2001-0887	race	int	local

Figure 29 Vulnerability Entries (Test Data – 2002)

Predicted Values								
	EntryID	VulnType	LossType	ExploitRange	EntryCluster	PredictedImpact	PredictedFrequency	LossExpect
▶	CVE-1999-1081	design	conf	remote	Cluster 3	4.9852	747	3723.94
	CVE-1999-1091	access	int	local	Cluster 3	4.9852	747	3723.94
	CVE-1999-1091	access	int	remote	Cluster 3	4.9852	747	3723.94
	CVE-2001-0887	design	int	local	Cluster 3	4.9852	747	3723.94
	CVE-2001-0887	race	int	local	Cluster 5	5.876	366	2150.62
	CVE-2001-0891	input	sec_prot	local	Cluster 2	8.3232	1218	10137.66
	CVE-2001-1457	input	sec_prot	remote	Cluster 2	8.3232	1218	10137.66
	CVE-2002-0002	input	sec_prot	remote	Cluster 2	8.3232	1218	10137.66

PredictDate: 1/ 1/2002
 12

Figure 30 - Predicted Values

Vulnerability Listing								
	EntryID	VulnType	LossType	ExploitRange		entryID	PredImpact	PredFreq
▶	CVE-1999-1081	design	conf	remote		CVE-2001-0891	8.32	1218
	CVE-1999-1091	access	int	local		CVE-2001-1457	8.32	1218
	CVE-1999-1091	access	int	remote		CVE-2002-0002	8.32	1218
	CVE-2001-0887	design	int	local		CVE-2002-0005	8.32	1218
	CVE-2001-0887	race	int	local		CVE-2002-1594	8.32	1218

Predicted Values								
	EntryID	VulnType	LossType	ExploitRange	EntryCluster	PredictedImpact	PredictedFrequency	LossExpect
▶	CVE-1999-1081	design	conf	remote	Cluster 3	4.9852	747	3723.94
	CVE-1999-1091	access	int	local	Cluster 3	4.9852	747	3723.94
	CVE-1999-1091	access	int	remote	Cluster 3	4.9852	747	3723.94
	CVE-2001-0887	design	int	local	Cluster 3	4.9852	747	3723.94
	CVE-2001-0887	race	int	local	Cluster 5	5.876	366	2150.62
	CVE-2001-0891	input	sec_prot	local	Cluster 2	8.3232	1218	10137.66
	CVE-2001-1457	input	sec_prot	remote	Cluster 2	8.3232	1218	10137.66
	CVE-2002-0002	input	sec_prot	remote	Cluster 2	8.3232	1218	10137.66

PredictDate: 1/ 1/2002
 12

Figure 31 - Impact and Loss Expectation Estimation

The goal of this research is to predict the impact score of a given risk element. The predicted impact is analogous to the base CVSS score provided by NVD. Our claim is

that our predicted impact score will be significantly equivalent to actual NVD's CVSS scores. We express this claim as $\mu_d = 0$. As such, we have the null hypothesis (H_0)

H₀: The impact values derived from SSRAM does not reflect the risk posture of the threat element in question.

$H_0: \mu_d > 0$ or $\mu_d < 0$ where

$\mu_d = (\mu_{\text{actual}} - \mu_{\text{predicted}})$ the difference of means of the predicted scores from the actual scores

versus

H_a: The impact values derived from SSRAM correctly reflects the risk posture of a software development effort in that $\mu_{\text{actual}} - \mu_{\text{predicted}} = 0$

$H_a: \mu_d = 0$

t-Test: Paired Two Sample for Means		
	<i>Actual Score</i>	<i>Predicted</i>
Mean	5.663889	6.10243
Variance	5.556087	2.225617
Observations	36	36
Pearson Correlation	0.67673	
Hypothesized Mean Difference	0	
df	35	
t Stat	-1.51354	
P(T<=t) one-tail	0.06956	
t Critical one-tail	1.689572	
P(T<=t) two-tail	0.139121	
t Critical two-tail	2.030108	

Table 11 - t-Test: Paired Two Sample for Means

The t-test critical value was computed and compared with the critical t-value at a significance level of $\alpha = 0.05$.

The sample mean of the difference is $\bar{d} = -0.4385$ and the sample standard deviation is $s_d = 1.7385$. The test statistic is

$$t_0 - \frac{\bar{d}}{\frac{s_d}{\sqrt{n}}} = \frac{-0.4385}{\frac{1.7385}{\sqrt{36}}} = -1.5135$$

Since this is a two-tailed test, we determined the critical t-value at the $\alpha = 0.05$ level of significance with $n - 1 = 36 - 1 = 35$ degrees of freedom to be $-t_{0.05/2} = -t_{0.025} = -2.030$ and $t_{0.025} = 2.030$. Because the test statistic $t_0 = -1.5135$ is greater than the critical value $-t_{0.025} = -2.030$ and $t_0 = 1.5135$ is less than $t_{0.025} = 2.030$, we reject the null hypothesis that there is a significant difference between the actual and predicted scores. These statistics suggest that there is no statistical significant difference between the predicted scores given by SSRAM and the actual scores as reported to NVD.

To further evaluate the predictions, we made a comparison of the predictions based on the three classification schemes as shown in Table 12, Table 13, and Table 14. The critical value for all three algorithms is 2.04 and the t-stat of 1.06 and 1.633 for the neural network and decision tree, as evidenced in Table 13, and Table 14, showed that the null hypothesis could be rejected. This result was consistent with the results of the predictive accuracy shown in section 3.5.1 that reflected that there was no significant difference between neural networks and decision trees when used for predictions. There was a difference with Naïve Bayes, which is also reflected with the t-stat of 2.13 as shown in Table 12. This t-stat does not allow us to reject the null hypothesis that there is no significant difference between the actual scores and the predicted scores when using Naïve Bayes as the classification scheme for predicting. The result of this analysis shows that either neural networks or decision tree algorithm can be used for classifying new risk elements.

	<i>Actual</i>	<i>Predicted</i>
Mean	5.577419	6.23709
Variance	5.520473	2.362795
Observations	31	31
Pearson Correlation	0.68273	
Hypothesized Mean Difference	0	
df	30	
t Stat	-2.1378	
P(T<=t) one-tail	0.020399	
t Critical one-tail	1.697261	
P(T<=t) two-tail	0.040798	
t Critical two-tail	2.042272	

Table 12 - t-Test Results using Naïve Bayes algorithm

	<i>Actual</i>	<i>Predicted</i>
Mean	5.76129	6.102877
Variance	5.636452	2.316325
Observations	31	31
Pearson Correlation	0.658869	
Hypothesized Mean Difference	0	
df	30	
t Stat	-1.06461	
P(T<=t) one-tail	0.147773	
t Critical one-tail	1.697261	
P(T<=t) two-tail	0.295546	
t Critical two-tail	2.042272	

Table 13 -t-Test: Paired Two Sample for Means using Decision Tree Classifier

	<i>Actual</i>	<i>Predicted</i>
Mean	5.653125	6.180540625
Variance	5.525796371	2.337423284
Observations	32	32
Pearson Correlation	0.629684793	
Hypothesized Mean Difference	0	
Df	31	
t Stat	1.633197931	
P(T<=t) one-tail	0.056272384	
t Critical one-tail	1.695518742	
P(T<=t) two-tail	0.112544769	
t Critical two-tail	2.039513438	

Table 14 - t-Test: Paired Two Sample for Means using Neural Networks Classification Algorithm

3.7 Confidence Interval of Predictions

The result of our predictions as shown in Figure 30 and Figure 31 are based on a confidence that the 95% scores predicted are \pm the margin of error shown in Table 15. The confidence interval was computed on the training data used for classification of new risk elements. Table 15 shows the number of distinct elements in each cluster, their average CVSS score, and the standard deviation from the mean for each cluster's score. Given the large number of entries for each cluster, we can assume normal distribution, and use these values to construct the margin of error as:

$$1.96 \times \frac{\sigma}{\sqrt{n}}$$

Equation 8 - Margin of Error

where σ the standard deviation and, n is the size of each cluster and, the value 1.96, based on a two-tailed normal distribution at the 95% confidence interval, is used as the critical value.

Training Data by Clusters (1996 - 2001)					With 95% Confidence Interval	
NumInCluster	CVSS_Score	StdDev	Margin of Error	Cluster	Lower	Upper
1257	8.2541	1.8124	0.1002	Cluster 2	8.1539	8.3543
719	7.0872	1.7984	0.1315	Cluster 4	6.9557	7.2186
107	7	0	0	Cluster 7	7	7
765	5.5393	2.0795	0.1474	Cluster 5	5.3919	5.6867
1298	5.3409	1.9665	0.107	Cluster 1	5.2339	5.4478
1250	4.3934	2.0171	0.1118	Cluster 3	4.2816	4.5052
735	4.3105	1.6957	0.1226	Cluster 6	4.1879	4.4331

Table 15 - Confidence Interval Derivation

Given this confidence interval derivation, we can say that we are confident that 95% of our predictions will be within the margin of error shown in Table 15. For example, in Figure 30 'CVE-1999-1081' is classified as Cluster 3 with a predicted impact of 4.9852.

Given our confidence interval construction, we can say that we are 95% confident that the predicted impact factor for ‘CVE-1999-1081’ is 4.9852 ± 0.1118 .

3.8 Summary

One of the deterrents to creating objective measures for assessing risks in software security was purported to be the lack of data to use as a basis for historical predictions. There are now open source vulnerability databases, one such being the NVD, which provide a historical data source that can be used to assess vulnerabilities. Since the data from NVD is an RSS feed in XML format, we parsed the NVD data into SSRAM’s database. In so doing, we were able to categorize the data based on the K-Means and EM algorithm to obtain natural groupings of the entries based on the data about each entry. These algorithms portrayed different approaches to categorizing data.

We compared the performance of the models between two mining structures, one with the description entries, a comment-like attribute without discrete values, and one without. This performance was measured by dividing the NVD data into two segments, one for training and the other for testing. We used the test data to find an indication of how likely it is for each test case to exist within a determined cluster. Based on the average case likelihood scores obtained, we saw that the clustering algorithms that took into account the description vectors performed better than those without. In addition to this, we examined the cohesion of each cluster with respect to the centroid of the cluster by looking at the mean, standard deviation, maximum and minimum values for each of the clusters within each mining model. We validated our choice by comparing the cohesion values of the algorithms.

Within each structure we compared EM versus K-Means models and chose those with higher average case likelihood scores. EM consistently outperformed KM models for our data. Choosing the algorithm with the best score, we labeled each entry of the training data with the cluster it belonged to. To determine what fields should be used to classify the data we looked at those fields with entropy values > 0.05 and < 0.95 . We then ran two multiple regression analysis, which at $\alpha = 0.05$ significance showed that the variables loss type, vulnerability type and exploit range were significant to predict cluster node and CVSS score. Upon naming each entry with its cluster, we investigated different classification algorithms to see which best classified the data under investigation. We again used two mining structures to compare results of classification mining models, one with the description terms and another without the description terms. Furthermore, we looked at the accuracy of each of the classification models and with a step-wise approach chose the algorithm with good accuracy and shorter performance time. With CVSS score used as the impact factor we adapted the traditional loss expectancy model by using CVSS score, as opposed to asset cost as the impact factor to determine loss expectation. In addition, we constructed confidence intervals for our predictions based on the training data sets used for classifying new entries at a 95% level of confidence.

Given these results, this study along with the historical validation of the model with statistical significance, suggests an empirical means for implementing risk assessment. It does so by providing a means for categorizing historical data (NVD) based on risk factors such as loss type, vulnerability type, exploit range and other product dependent factors. These discovered categories allow for classifying new threats, and through this classification, predict the impact for exercising the threat and the frequency of occurrence

of such exercise. These predicted values are used to rank the risk elements identified during a threat modeling activity and provide a means for objectively justifying the approaches chosen to ameliorate stated risks.

4 APPLICATION OF MODEL TO CASE STUDY

This chapter shows the application of the model to a case study. We looked at an electronic voting system Prime III developed by the Human Computer Centered Lab at Auburn University [Prime III]:

The Prime III voting system should be engineered through a human-centered computing approach. This approach considers the users first and implements a design that accommodates users and integrates usability with the necessary safeguards to provide security, trust, and privacy (i.e., usable security). Prime III should provide a naturally interactive user interface that reduces the learning curve by using multiple modalities during the voting process. Voters will cast their votes using touch and/or voice commands as shown in [Figure 32], eliminating the need for specialized machines for one segment of the voting population. Everyone votes on the same type of machine, independently without additional assistance. The procedure for a voter should be as follows:

1. The voter steps into the voting booth and puts on the provided headset.
2. The voter begins voting as normal.
3. Each time the voter makes a selection either by voice or touch, he hears a confirmation in the headphones and simultaneously observes the confirmation on the screen. For example, when the voter selects candidate A, Prime III should say “selected candidate A” and concurrently display the selection. This audio and visual confirmation should be heard and seen for every important action that the voter takes including, selecting a candidate, unselecting a candidate, advancing races, and submitting the ballot.
4. The Prime III visual and audio output are passed to the video recording unit which records the voting session on some physical medium such as a video cassette.
5. At the end of the session the voter confirms his selection twice; then submits their ballot and leaves the voting booth.

As illustrated below in Figure 32, Prime III will be comprised of systems and software that are deployable at each electoral precinct. The basic functions of this system are to permit citizens to vote, and to allow for the tabulation and creation of an auditable trail of the ballots cast.

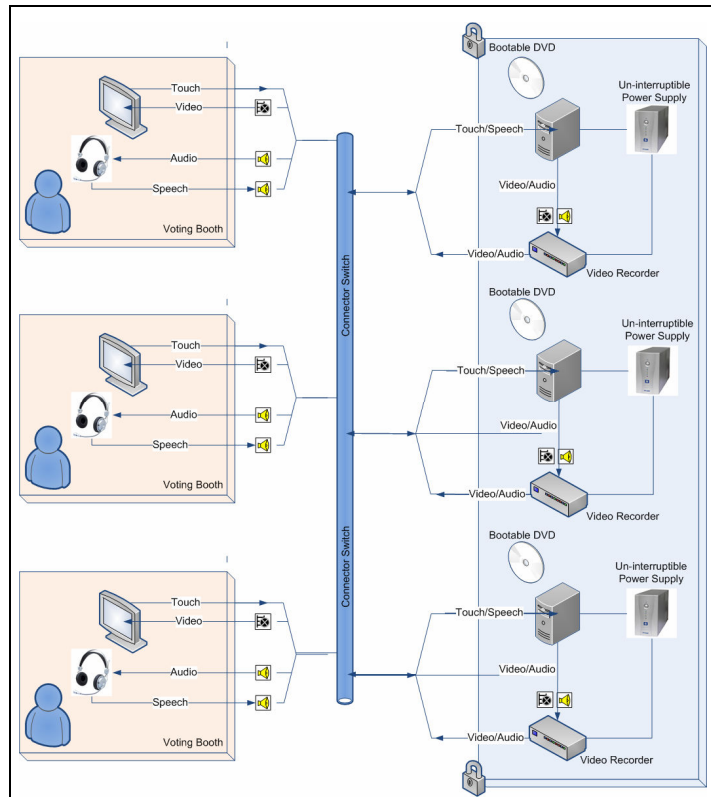


Figure 32: Prime III System Architecture High-Level Overview

4.1 Vulnerability Identification and Data Preparation

To determine the vulnerabilities for this application, Microsoft's Threat Analysis and Modeling Tool (TAMT) was used to generate the non-prioritized list of vulnerabilities (Appendix E-1), namely, Canonicalization, Buffer Overflow, Cryptanalysis Attacks, Format String and Integer Underflow or Overflow.

Before entering this non-prioritized list into SSRAM, each of the threats given was evaluated for the type of security impact that would be affected if the threat were exercised, the kind of vulnerability this threat could exploit, and the exploitation range that could exist for it. A summary of the threats as given by TAMT, the impact of loss, the vulnerability and exploit range evaluations performed are shown in Table 16. It

should be noted that the TAMT provided the threats and descriptions from which we derived the loss type, vulnerability and exploit range values.

ID	Threat	Description	Loss Type	Vulnerability	Exploit Range
Threat-1	Canonicalization	Canonicalization : Only accept primitive typed identified (e.g., integers) which are mapped to filenames	Confidentiality Availability Integrity	Input	Local User- init
Threat-2	Buffer Overflow	Buffer Overflow : Use safe functions such as strncpy, strncat instead of strcpy, strcat Validation on input should be performed.	Confidentiality Availability Integrity	Input Access	Local User- Init
Threat-3	Cryptanalysis Attacks	Cryptanalysis Attacks : Use well-known implementations of well-known cryptographic algorithms Use cryptographically generated random keys Utilize platform supplied feature to store secret key (e.g., DPAPI) Utilize SSL or IPSec w/ Encryption to establish a secure communication channel	Confidentiality Availability Integrity Security Protection – Admin	Environment Access	Local
Threat-4	Format String	Format String : Use a managed language	Confidentiality Availability Integrity	Design Access	Local
Threat-5	Integer Overflow/Underflow	Integer Overflow/Underflow : Use Language features	Confidentiality Availability Integrity	Input	Local

Table 16 - Prime III Vulnerability List

The vulnerability listing and the evaluation of threat descriptions as shown in Table 16 provided the basis for creating tables that associated each threat to its loss type, vulnerability type and exploit range as portrayed in Table 17 (a – c). These tables were then de-normalized using a select statement similar to that of Figure 14 to create a table similar to Table 18. The complete de-normalized table is shown in Appendix E-2.

ID	LossType		ID	VulnType		ID	ExploitRange	
Threat-1	Avail		Threat-1	input		Threat-1	user_init	
Threat-1	Int		Threat-2	access		Threat-1	local	
Threat-1	Conf		Threat-2	input		Threat-2	user_init	
Threat-2	Int		Threat-3	env		Threat-2	local	
Threat-2	Avail		Threat-3	access		Threat-3	local	
Threat-2	Conf		Threat-4	design		Threat-3	local	
Threat-3	Avail		Threat-5	input		Threat-4	user_init	
Threat-3	Int		(b)			Threat-4	local	
Threat-3	Conf					Threat-5	user_init	
Threat-3	Sec_Prot					(c)		
Threat-4	Avail							
Threat-4	Int							
Threat-4	Conf							
Threat-5	Avail							
Threat-5	Int							
Threat-5	Conf							
(a)								

Table 17- Prime III Normalized Tables

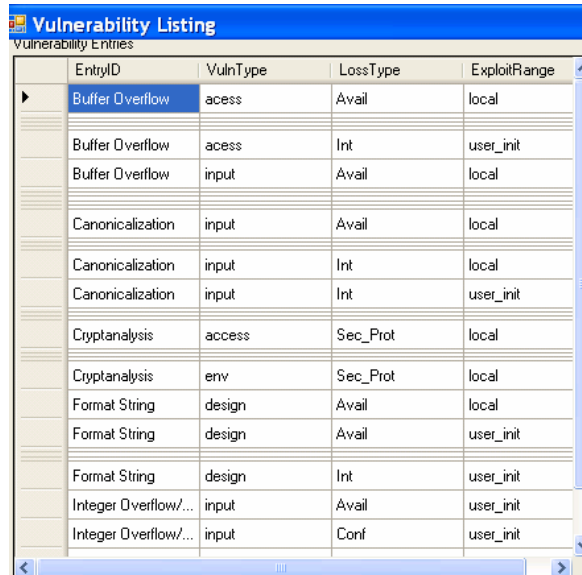
ID	Name	VulnType	Loss Type	Exploit Range
Threat-1	Canonicalization	input	Avail	user_init
Threat-1	Canonicalization	input	Avail	local
Threat-1	Canonicalization	input	Int	user_init
Threat-1	Canonicalization	input	Int	local
Threat-1	Canonicalization	input	Conf	user_init
Threat-1	Canonicalization	input	Conf	local
Threat-2	Buffer Overflow	access	Int	user_init
Threat-2	Buffer Overflow	input	Int	user_init
Threat-2	Buffer Overflow	access	Int	local
Threat-2	Buffer Overflow	input	Int	local
Threat-2	Buffer Overflow	access	Avail	user_init
Threat-2	Buffer Overflow	input	Avail	user_init
Threat-2	Buffer Overflow	access	Avail	local
Threat-2	Buffer Overflow	input	Avail	local
Threat-2	Buffer Overflow	access	Conf	user_init

Table 18 - Example of PrimeIII's Input Data

4.2 Data Classification and Loss Expectation Determination

As noted in section 3.1, using SSRAM to assess risk elements requires that the historical data used be segmented into clusters (Figure 8). Since this was already done with data from 1996 – 2001, we imported the data shown in Table 18 into SSRAM database and

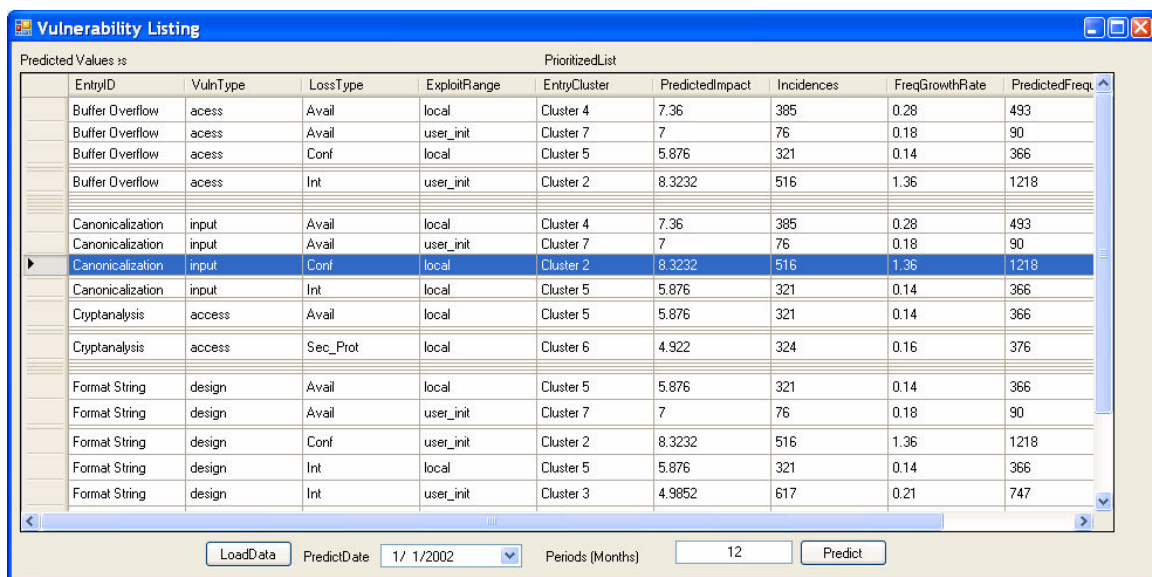
loaded same data into SSRAM's application (Figure 33) to assess the risk of the vulnerabilities identified. SSRAM classified each of the de-normalized entries based on its similarity to the clusters within SSRAM. The data within the cluster assigned to each entry was then used to calculate the impact factor and the frequency of occurrence of each threat within the time constraint given (Figure 34).



Vulnerability Listing
Vulnerability Entries

EntryID	VulnType	LossType	ExploitRange
Buffer Overflow	access	Avail	local
Buffer Overflow	access	Int	user_init
Buffer Overflow	input	Avail	local
Canonicalization	input	Avail	local
Canonicalization	input	Int	local
Canonicalization	input	Int	user_init
Cryptanalysis	access	Sec_Prot	local
Cryptanalysis	env	Sec_Prot	local
Format String	design	Avail	local
Format String	design	Avail	user_init
Format String	design	Int	user_init
Integer Overflow/...	input	Avail	user_init
Integer Overflow/...	input	Conf	user_init

Figure 33 - Load Prime III



Vulnerability Listing

Predicted Values is				PrioritizedList				
EntryID	VulnType	LossType	ExploitRange	EntryCluster	PredictedImpact	Incidences	FreqGrowthRate	PredictedFreq
Buffer Overflow	access	Avail	local	Cluster 4	7.36	385	0.28	493
Buffer Overflow	access	Avail	user_init	Cluster 7	7	76	0.18	90
Buffer Overflow	access	Conf	local	Cluster 5	5.876	321	0.14	366
Buffer Overflow	access	Int	user_init	Cluster 2	8.3232	516	1.36	1218
Canonicalization	input	Avail	local	Cluster 4	7.36	385	0.28	493
Canonicalization	input	Avail	user_init	Cluster 7	7	76	0.18	90
Canonicalization	input	Conf	local	Cluster 2	8.3232	516	1.36	1218
Canonicalization	input	Int	local	Cluster 5	5.876	321	0.14	366
Cryptanalysis	access	Avail	local	Cluster 5	5.876	321	0.14	366
Cryptanalysis	access	Sec_Prot	local	Cluster 6	4.922	324	0.16	376
Format String	design	Avail	local	Cluster 5	5.876	321	0.14	366
Format String	design	Avail	user_init	Cluster 7	7	76	0.18	90
Format String	design	Conf	user_init	Cluster 2	8.3232	516	1.36	1218
Format String	design	Int	local	Cluster 5	5.876	321	0.14	366
Format String	design	Int	user_init	Cluster 3	4.9852	617	0.21	747

LoadData PredictDate 1/ 1/2002 Periods (Months) 12 Predict

Figure 34 - Prime III De-normalized Predictions

The de-normalized vulnerabilities list, along with the impact criteria, served as the basis for generating the predicted values. The Predicted Values data grid (Figure 34) shows not only the vulnerability entries uploaded but the cluster an entry is most similar to (EntryCluster), the predicted impact based on that cluster (PredictedImpact), the number of entries reported of like incidences for the time period under evaluation (Incidences), the rate of growth of said incidences (FregGrowthRate), the predicted frequency of occurrence (Predicted Frequency) and the loss expectation (LossExpect). The predicted impact value derived is an estimation of the CVSS base score, and the predicted frequency of occurrence, which is the product of the incidences and the rate of growth of the number of incidences reported, reflects the evaluated period's periods predicted number of occurrences. These predictions are based on a statistically significant, historically validated approach as explained in section 3.6.3 and summarized in Table 11 of same section. The prediction of impact is based on the confidence that 95% of predictions will fall within the margin of error as explained in section 3.7 of Chapter 3. For example, in Figure 34, the first entry is assigned to Cluster 4 with a predicted impact of 7.36 ± 0.1315 (Table 15). The loss expectation is calculated as the product of the predicted impact and the predicted frequency of occurrence (Equation 4), following the best practice used in calculating traditional loss expectancy (Equation 1). This information is then aggregated for each vulnerability entry and ranked in descending order, as seen in the resulting prioritized list (Figure 35).

Since loss expectation is derived from the impact factor and the frequency of occurrence, we can assess situations when threats of lower impact may require more attention due to the high predicted frequency of occurrence of attacks to a particular vulnerability. In

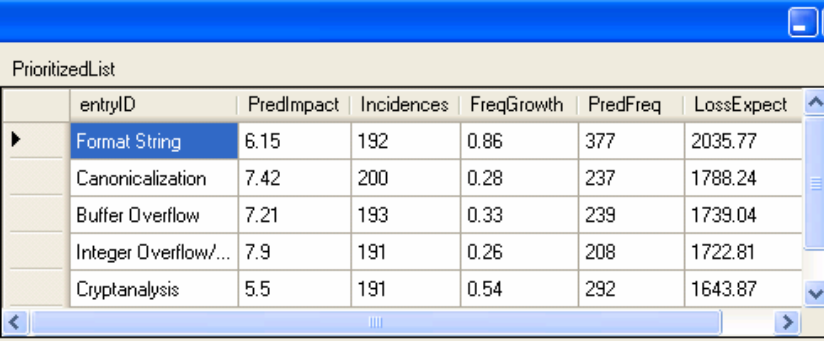
Figure 35, the prioritized list of threats identified and predicted for January of 2001 shows 'Format String' as the risk element of highest threat, with a loss expectation value of 2266.78. Even though the impact factor for 'Format String' (6.15) is less than that of 'Canonicalization' (7.42), 'Format String' would still be considered a greater threat because of its higher predicted number of occurrences (377) when compared to 'Canonicalization' (237). In looking at the aggregated predicted impact scores, the clusters to which they most resemble, and the confidence intervals for each cluster as depicted in Table 15 of section 3.7, we observed that 'Canonicalization', 'Buffer Overflow', and 'Integer Overflow/Underflow' would be classified to the same cluster – 'Cluster 4'. 'Cryptanalysis' and 'Format String' would be classified as Cluster 5 data. As can be seen in Figure 35, the difference in priority is based on the predicted frequency of threats to exercise these vulnerabilities. Since our prediction of impact is based on the CVSS score, we note that the values for Prime III vulnerabilities warrant that all listed vulnerabilities be addressed during the development phase of the software development life cycle. The CVSS score range from 0 – 9, where 0 indicates no damage potential and 9 signifies a high collateral damage. All scores for the prediction depicted in Figure 35 are above the mid-level point of 5 and would be considered at least medium to high risk vulnerabilities. In order of priority based on the loss expectation 'Format String', 'Canonicalization', 'Buffer Overflow', 'Integer Overflow/Underflow' and 'Cryptanalysis' should be considered in the order listed. Based on the confidence interval for the predictions, we would surmise that 'Canonicalization', 'Buffer Overflow' and 'Integer Overflow/Underflow' would be of the same level of importance and though 'Cryptanalysis' and 'Format String' are of the same classification, due to the frequency of

predicted exercise of the ‘Format String’ vulnerability that it take precedence in mitigation.

Figure 36 shows the prediction of impact based on the same vulnerability list but with historical data from 2003 – 2005 to predict January 2006 data. To obtain the clusters for classifying this data, we applied EM clustering algorithm (NVDCL-EM10). It can be seen that though the predicted impact scores have become less, the frequency of occurrence of threats have at least tripled. It should also be noted that the vulnerability whose loss expectation implies greatest collateral damage has also changed from ‘Format String’ to ‘Buffer Overflow’. ‘Canonicalization’ and ‘Integer Overflow/Underflow’ now command the same level of attention and though their impact scores (4.56) are slightly below the medium risk level, the high number associated with frequency of occurrence (1,784) requires that countermeasures to address these vulnerabilities are addressed, even before attention is given to ‘Format String’ vulnerability even though it has a higher impact score. The full prediction and confidence interval information for Prime III is in Appendix E-3.

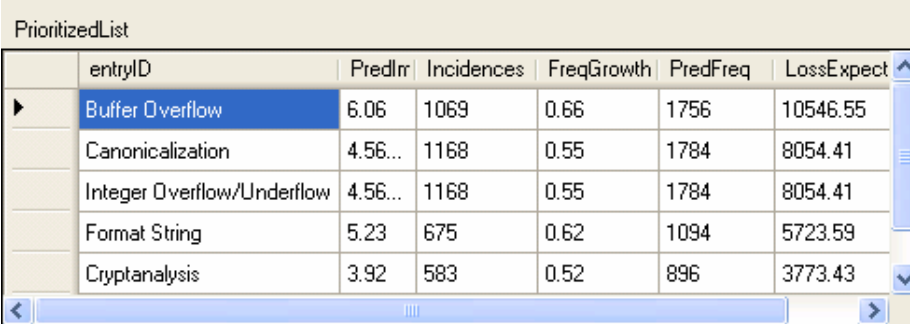
Training Data by Clusters (1996 - 2001)					With 95% Confidence Interval
CVSS_Score	StdDev	Margin of Error	Cluster	Lower	Upper
8.2541	1.8124	0.1002	Cluster 2	8.1539	8.3543
7.0872	1.7984	0.1315	Cluster 4	6.9557	7.2186
7	0	0	Cluster 7	7	7
5.5393	2.0795	0.1474	Cluster 5	5.3919	5.6867
5.3409	1.9665	0.107	Cluster 1	5.2339	5.4478
4.3934	2.0171	0.1118	Cluster 3	4.2816	4.5052
4.3105	1.6957	0.1226	Cluster 6	4.1879	4.4331

Table 15 - Confidence Interval Derivation



entryID	PredImpact	Incidences	FreqGrowth	PredFreq	LossExpect
Format String	6.15	192	0.86	377	2035.77
Canonicalization	7.42	200	0.28	237	1788.24
Buffer Overflow	7.21	193	0.33	239	1739.04
Integer Overflow/...	7.9	191	0.26	208	1722.81
Cryptanalysis	5.5	191	0.54	292	1643.87

Figure 35 - Prime III Loss Expectation (Prioritized List) – Predictions



entryID	PredImpact	Incidences	FreqGrowth	PredFreq	LossExpect
Buffer Overflow	6.06	1069	0.66	1756	10546.55
Canonicalization	4.56...	1168	0.55	1784	8054.41
Integer Overflow/Underflow	4.56...	1168	0.55	1784	8054.41
Format String	5.23	675	0.62	1094	5723.59
Cryptanalysis	3.92	583	0.52	896	3773.43

Figure 36 - Prime III Loss Expectation (Prioritized List) – 2006 Predictions

Given this ordered list with empirical values showing the vulnerabilities with their associated impact values and predicted number of occurrences, we have provided an objective measure that can be used to justify dealing with the vulnerabilities in question. Unlike vulnerability identification systems that list in no particular order of importance the vulnerabilities and threats of a particular effort, SSRAM's prediction of the impact factor and the frequency of occurrence provides an objective means for determining and justifying which threats to ameliorate. SSRAM also provides an empirical foundation for focusing testing later in the development process to make sure that the development effort took steps to lessen the chances of these threats being successfully exercised. The analysts can now determine, based on the tools at hand, what needs to be done, what resources need to be acquired and which vulnerabilities they want to concentrate efforts and resources. In essence they can better perform analysis analogous to cost benefit

analysis and empirically assess the opportunity cost of the risk elements they chose to address.

5 SUMMARY, FUTURE WORK AND CONCLUSIONS

5.1 Summary

When responding to system vulnerabilities, efforts tend towards reactive measures that support the hardening of systems and their connected networks. Even though these efforts are necessary, they do not address the fact that the majority of the security vulnerabilities are due to software vulnerabilities, as reported by NIST [Curphey 2004]. Efforts to ameliorate system vulnerabilities should in effect start early in the software development life cycle, so that security is built in and not bolted onto the system upon completion of development. To this end, risks to a software development effort have to be determined and assessed early in the development life cycle. Attempts to date to assess the risks that could apply to a development effort have been largely qualitative in nature. Though helpful, these qualitative approaches have drawbacks such as the difficulty in duplicating results and transferring assessment lessons to other projects. The traditional quantitative approach of annualized loss expectancy (ALE) and the exposure factor (EF) carries with it the burden of knowing what asset costs to use. The impact effect is measured by the cost of asset loss, which is not easily determined during the design phase of development. If the cost of asset loss is available, it may not adequately reflect the impact that is a result of the exercise of a given threat. Other efforts that provide objective results are implemented late in the software development life cycle,

usually during the testing phase, thereby incurring additional costs for dealing with vulnerabilities assessed after the system has been built.

In looking at risk assessment in other fields such as insurance, banking, and finance, we find that one of the factors that allows for prediction of risk metrics is the availability of historical data. One of the deterrents to creating objective measures for assessing risks in software security was purported to be the lack of data to use as a basis for historical predictions. As it stands, there are now open source vulnerability databases, which provide a historical data source that can be used to assess vulnerabilities. We chose the NVD, and in so doing we were able to categorize the data based on the K-Means and EM algorithm to obtain natural groupings of the entries based on the data about each entry. These algorithms portrayed different approaches to categorizing data.

We compared the performance of the models between two mining structures, one with the description entries, a comment-like attribute without discrete values, and one without. This performance was measured by dividing the NVD data into two segments, one for training and the other for testing. We used the test data to find an indication of how likely it is for each test case to exist within a determined cluster. Based on the average case likelihood scores obtained, we saw that the clustering algorithms that took into account the description vectors performed better than those without. In addition to this, we examined the cohesion of each cluster with respect to the centroid of the cluster by looking at the mean, standard deviation, maximum and minimum values for each of the clusters within each mining model. We validated our choice by comparing the cohesion values of the algorithms.

Within each structure we compared EM versus K-Means models and chose those with higher average case likelihood scores. The average case likelihood score measures the likelihood that a test case would be similar to one of the cases within the training model. EM consistently outperformed KM models for our data. Choosing the algorithm with the best score, we labeled each entry of the training data with the cluster it belonged to. To determine what fields should be used to classify the data we looked at those fields with entropy values > 0.05 and < 0.95 . We then ran two multiple regression analysis, which at $\alpha = 0.05$ significance showed that the variables loss type, vulnerability type and exploit range were significant to predict cluster node and CVSS score. Upon naming each entry with its cluster, we investigated different classification algorithms to see which best classified the data under investigation. We again used two mining structures to compare results of classification mining models, one with the description terms and another without the description terms. Furthermore, we looked at the accuracy of each of the classification models and with a step-wise approach chose the algorithm with good accuracy and shorter performance time. With CVSS score used as the impact factor we adapted the traditional loss expectancy model by using CVSS score as opposed to asset cost as the impact factor to determine loss expectation. In addition, we constructed confidence intervals for our predictions based on the training data sets used for classifying new entries at a 95% level of confidence. With the ability to predict CVSS score with statistical significance, our study suggests an empirical approach to determining the impact of a given risk or threat and provides a way to objectively compare the impact and frequency of occurrence of identified threats.

5.2 Future Work

Any significant research effort invariably opens more questions than it solves. This endeavor was no different. The following are suggestions for extending the research:

- Validate SSRAM with an industry project. Our research looked at the Prime III development effort but did not validate the results of its work. We propose to run SSRAM in a development environment and assess its effectiveness.
- Investigate the usage of SSRAM to assess risk on component based systems. As development efforts rely more and more on other components, empirically assessing the risk of utilizing components within a development effort is necessary.
- Perform comparative studies. We propose to do comparative studies using other vulnerability databases and other data mining algorithms for classification and prediction of impact factors.
- Strengthen predictive values. We will investigate factors, along with time, which will strengthen the predictive values of the impact scores. Along with this, we propose to look at the trend of vulnerabilities reported based on security factors such as impact scores, loss type, vulnerability type, and exploit range.
- Investigate the characteristic of the clusters within SSRAM. We will investigate the characteristics of the clusters determined within SSRAM to better describe the classification of each entry. As it stands, SSRAM classifies each entry to a generic cluster – “Cluster 1 ... Cluster N”.
- Investigate other variables that can be used to determine clusters. The R^2 statistic of 27.9% in Figure 24 shows that there is a need to explore other variables that can

be used to predict the cluster node. Though Loss Type, Vulnerability Type and Exploit Range are significant predictors of cluster node, they are not enough to explain the model.

- Generate countermeasures based on the vulnerabilities assessed. SSRAM would be enhanced by the provision of countermeasures to enact based on the vulnerabilities assessed.
- Integrate SSRAM with a vulnerability identification system. The use of SSRAM is predicated on the identification of vulnerabilities. The integration of SSRAM to an identification system would reduce the amount of work necessary to implement SSRAM and overall produce a more robust risk assessment methodology.

5.3 Conclusions

In creating a model that predicts impact factors, we found that there was no statistical difference between the actual scores and the predicted scores produced by our research.

We were able to estimate the CVSS score and use it as the impact factor estimation that allows loss expectation to be quantitatively derived based on historical data. Along with that we also produced confidence intervals for the predicted impact scores.

This study shows that the database of vulnerabilities is adequate for use as a basis for predicting risk metrics such as impact factor. We were also able to provide a categorization of vulnerabilities based on development environment factors along with descriptive terms of the vulnerabilities. This categorization allowed us to be able to classify new vulnerabilities based on their similarities to historical entries.

This research has shown that the predictions based on these categorizations were not significantly different from the actual impact scores for the time under research.

	Data Availability	Assess Risk Elements	Can be used at design level of development	Generalize & Duplicate Results	Categorization based on Development Environmental Factors	Identify Risks
SSRAM	Yes	Yes	Yes	Yes	Yes	No
Traditional Risk Impact Estimation	No	Yes	No	No	No	No
Software Reliability	Yes	No	Yes	Yes	N/A	No
Qualitative Assessment approaches (Boehm, RiskIT, Risk Management Framework)	N/A	Yes	Yes	No	?	No
Testing & Assessment Approaches	Yes	Yes	No	Yes	No	No
Identification Approaches	Yes	No	Yes	Yes	No	Yes

Table 19 – Comparison of SSRAM with other Risk Analysis Methodologies

This research shows that our approach (SSRAM) when compared to other risk analysis methodologies is capable of being used to objectively assess risk elements early in the software development life cycle due to its ability to classify risk elements to categories that are determined based on the development factors (Table 19). SSRAM provides a basis for objectively assessing threats or vulnerabilities early in the software development life cycle. The prediction of both the impact value and the number of occurrences of a given threat in a prioritized format provides objective evidence that can direct risk mitigation during the development of software. Assessment lessons can be transferred to other projects and the predictions from SSRAM can be used to do comparative analysis of different projects with similar parameters. Since the assessment is done early in the development cycle, these vulnerabilities can be mitigated during the development effort

without the added cost usually incurred when they are discovered and assessed during later periods in the software development life cycle such as the testing phase.

Given the pivotal role risk assessment has in the development of reliable and secure systems, SSRAM provides a historically validated risk assessment model for analyzing risks so that an objective justification of the direction and choice of risk elements can be made.

BIBLIOGRAPHY

- [Addison and Vallabh 2002] Controlling Software Project Risks – an Empirical Study of Methods used by Experienced Project Managers. *Proceedings of South African Institute of Computer Scientists and Information Technologists SAICSIT* (September 2002), 128-140
- [Alberts 2006] “Common Elements of Risk” CMU SEI 2006
“<http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06tn014.pdf>”
- [Bartlett II et al] Bartlett II, J., Kotrlik, J. W., Higgins, C. C., “Organizational Research: Determining Appropriate Sample Size in Survey Research” *Information Technology, Learning and Performance Journal* (Spring 2001), 43-50.
- [Boehm 1989] Boehm B., *Software Risk Management*, IEEE-Computer Science Press, 1989
- [Boehm 2001] Boehm, B.; Educating software engineering students to manage risk” , 2001. *Proceedings of the 23rd International Conference Software Engineering* (12-19 May 2001), 591 – 600
- [Boehm and Demarco 1997] Boehm, B.W.; and DeMarco, T.; Software risk management *IEEE Software*, Volume 14, Issue 3, (May-June 1997), 17 – 19
- [Bradley, Fayyad, Reina 1998] Bradley, P. S., Fayyad, U., and Reina, C., “Scaling EM (Expectation-Maximization) Clustering to Large Databases,” Technical Report MSR-TR-98-35, Microsoft Research. (1998)
- [Carr et al. 1993] Carr M., Konda S ., Monarch I., Ulrich, F., and Walker, C., “Taxonomy-Based Risk Identification,” *CMU/SEI-93-TR=6*, Software Engineering Institute, Pittsburgh, PA 15213, 1993.
- [Cheswick et al. 2003] Cheswick, B, Kocher, P., McGraw, G., and Rubin, A.. “Bacon Ice Cream: The Best Mix of Proactive and Reactive Security?” *IEEE Security and Privacy* 1.4. (2003)
- [Curphey 2004] Curphey, M. “Software Security Testing: Let's Get Back to Basics” *Security* (October 2004) <http://www.softwaremag.com/L.cfm?Doc=2004-09/2004-09software-security-testing> last accessed 6/15/07

- [Farahmand 2003] Farahmand, F., Navathe, S. B., Sharp, G., Phillip P., Enslow H., Managing Vulnerabilities of Information Systems to Security Incidents *International Conference on Entertainment Computing ICEC* Pittsburgh, PA, (2003).
- [First 2005] CVSS v1 Complete Documentation <http://www.first.org/cvss/v1/guide.html> last accessed 5/2006
- [First 2008] CVSS v2 Complete Documentation <http://www.first.org/cvss/cvss-guide.html#12.1> last accessed 7/22/08
- [FIPS 2004] Federal Information Processing Standard (FIPS) 199 - <http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf> - last accessed 3/27/07
- [GAO AIMD-00-33] Information Security Risk Assessment: Practices of Leading Organizations (GAO AIMD-00-33, 1999) <http://www.gao.gov/special.pubs/ai00033.pdf> last accessed 1/24/07
- [Haimes 2004] Haimes, Y. V. *Risk Modeling, Assessment, and Management 2nd ed.* Wiley Press, 2004
- [Hetzel 1988] Hetzel, William C., *The Complete Guide to Software Testing, 2nd ed.* Publication info: Wellesley, Mass. : QED Information Sciences, 1988. ISBN: 0894352423. Physical description: ix, 280 p. : ill ; 24 cm
- [Howard and Leblanc] Howard, Michael & LeBlanc, David C. *Writing Secure Code, 2nd ed.* Redmond, WA: Microsoft Press, 2002
- [Howard et al. 2005] Howard M., LeBlanc D., and Viega J., *19 Deadly Sins of Software Security*, McGraw-Hill, (2005).
- [Humphrey et al. 2004] Humphrey, W., Davis, N., Redwine Jr., S. T., Zibulski, G., McGraw, G., “Processes for Producing Secure Software – Summary of US National Cybersceurity Summit subgroup Report” *IEEE Security & Privacy* (May/June 2004)
- [Janardhanudu] Janardhanudu, G. “White Box Testing” <https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/white-box/259-BSI.html> last accessed 3/9/2006
- [Jain et al. 1996] Jain, A.K.; Mao, J., Mohiuddin, K.M.; “Artificial neural networks: a tutorial” *Computer Volume 29, Issue 3*, (March 1996),:31 - 44
- [Jurjens 2001] Jan Jurjens. “Towards Secure Systems Development with UMLsec,” Proceedings of FASE 2001. Springer Lecture Notes in Computer Science 2001

- [Karunanithi et al. 1992] Karunanithi, N.; Whitley, D.; Malaiya, Y.K Using neural networks in reliability prediction *Software, IEEE Volume 9, Issue 4*, (July 1992), :53 – 59
- [Khoshgoftar 1995] Khoshgoftar T.M., Szabo R.M., and Guasti P.J., “Exploring the Behaviour of Neural Network Software Quality Models,” *Software Engineering Journal.*, (May 1995).
- [Klugman et al. 2004] Klugman, S. A., Panjer, H. H., Willmot, G. E., *Loss Models From Data to Decisions 2nd edition Wiley Series in Probability and Statistics John Wiley & Sons Inc.* (2004)
- [Kontio 1997] Kontio, J., The Riskit Method for Software Risk Management, version 1.00 CS-TR-3782 / UMIACS-TR- 97-38, 1997. Computer Science Technical Reports. University of Maryland. College Park, MD
- [Kontio 1999] Kontio, J.; Risk management in software development: a technology overview and the riskit method *Software Engineering, 1999. Proceedings of the 1999 International Conference* (May 16-22 1999), 679 — 680
- [Kontio et al. 1998] J. Kontio, G. Getto, and D. Landes. Experiences in improving risk management processes using the concepts of the Riskit method. In *Proceedings. ACM SIGSOFT Int’l Symp. Foundations Softw. Eng.*, (1998), 163–174
- [Lodderstedt et al. 2002] Lodderstedt T., Basin D., Doser, J.; SecureUML: A UML-Based Modeling Language for Model-Driven Security
http://kisogawa.inf.ethz.ch/WebBIB/publications-softech/papers/2002/0_secuml_uml2002.pdf last accessed 07/03/2007
- [Lyu et al. 1996] Michael R. Lyu (editor) *Handbook of Software Reliability Engineering* IEEE Computer Society Press and McGraw-Hill Book Company 1996
<http://www.cse.cuhk.edu.hk/~lyu/book/reliability/> last accessed 3/30/07
- [McGraw 2002] McGraw, G., “Managing Software Security Risks” *Computer* (April 2002)
- [McGraw 2006] McGraw, G., *Software Security Building Security In Addison-Wesley Software Security Series*, Boston, MA.(2006)
- [Mead and Stehney 2005] Mead, N. R.; Stehney, T. “ Security Quality Requirements Engineering (SQUARE) Methodology”
- [Mead 2005] Mead, Nancy R., Hough, Eric D. , Stehney II, Theodore R. *Software Quality Requirements Engineering (SQUARE) methodology*
<http://www.sei.cmu.edu/pub/documents/05.reports/pdf/05tr009.pdf> November 2005

- [Mead 2006] Mead, Nancy R. Software Engineering Institute Carnegie Mellon University <https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/requirements/232.html>. Last accessed 1/24/07
- [Michael and Radosevich 2005] Michael, C., and Radosevich “Risk-Based and Functional Security Testing” Building Security In https://buildsecurityin.us-cert.gov/portal/article/bestpractices/security_testing/overview.xml#Risk-Based-Testing
- [Microsoft 2006] <http://www.microsoft.com/downloads/details.aspx?familyid=570dccd9-596a-44bc-bed7-1f6f0ad79e3d&displaylang=en> accessed last - 3/9/07
- [Microsoft 2008-1] ms-help://MS.VSCC.v80/MS.VSIPCC.v80/MS.SQLSVR.v9.en/uas9/html/61eb4861-8a6a-4214-a4b8-1dd278ad7a68.htm accessed last – 5/20/08
- [Microsoft 2008-2]] ms-help://MS.VSCC.v80/MS.VSIPCC.v80/MS.SQLSVR.v9.en/uas9/html/3b53e011-3b1a-4cd1-bdc2-456768ba31b5.htm accessed last – 5/20/08
- [Microsoft 2008-3] ms-help://MS.VSCC.v80/MS.VSIPCC.v80/MS.SQLSVR.v9.en/uas9/html/95ffe66f-c261-4dc5-ad57-14d2d73205ff.htm accessed last – 5/20/08
- [Microsoft 2008-4] ms-help://MS.VSCC.v80/MS.VSIPCC.v80/MS.SQLSVR.v9.en/uas9/html/61eb4861-8a6a-4214-a4b8-1dd278ad7a68.htm accessed last – 7/1/08
- [Microsoft 2008-5] ms-help://MS.VSCC.v80/MS.VSIPCC.v80/MS.SQLSVR.v9.en/uas9/html/3b53e011-3b1a-4cd1-bdc2-456768ba31b5.htm accessed last – 5/20/08
- [Mkpong-Ruffin and Umphress 2007] Mkpong-Ruffin, I., Umphress, D. A., “High-Leveraged Techniques for Software Security” CrossTalk The Journal of Defense Software Engineering (March 2007)
- [Neumann 2002] D. E. Neumann. An enhanced neural network technique for software risk analysis. IEEE Transactions Software Eng., 28(9):904–912, 2002.
- [NIST 2002-10] National Institute of Standards and Technology, “Software Errors Cost U.S. Economy \$59.5 Billion Annually” (NIST 2002-10). http://www.nist.gov/public_affairs/releases/n02-10.htm (2002).
- [NVD] National Vulnerability Database www.nist.nvd.gov last accessed 4/7/07
- [OWASP] http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project last accessed 4/5/07

- [Pan 1999] 16. Pan, Jiantao. "Software Testing – 18- 849b Dependable Embedded Systems." Carnegie Mellon University, 1999 ,www.ece.cmu.edu/~koopman/des_s99/sw_testing>.
- [Prime III] <http://www.primevotingsystem.com/> last access 7/29/08
- [Rosenberg et al. 1998] Linda Rosenberg, Ted Hammer, Jack Shaw International Symposium on Software Reliability November 1998
http://satc.gsfc.nasa.gov/support/ISSRE_NOV98/software_metrics_and_reliability.html - last accessed 3/30/07
- [Sherer 1995] Sherer, S.A., "Software Fault Prediction," J. Systems and Software, vol. 2 no. 2, (May 1995).
- [Sargent 2003] Sargent, R. G., "Validation and Verification of Simulation Models" *Proceedings of the 2003 Winter Simulation Conference* S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.
- [Schneier 2000] Schneier, B. *Secrets and Lies: Digital Security in a Networked World*. New York: John Wiley & Sons, 2000
- [Sindre and Opdahl] Sindre, G., and Opdahl, A.L.,. "Templates for Misuse Case Description." *Proc. Of the Seventh International Workshop on Requirements Engineering, Foundation for Software Quality (REFSQ 2001)*, (4-5 June 2001), Switzerland.
- [SP 800-30] Stoneburner, G.; Goguen, A.; Feringa, A., NIST SP-300 Risk Management Guide for Information Technology Systems - (July 2002)
- [SP 800-53] Ross, R., Katzke, S., Johnson, A., Swanson, Stoneburner, G., Rogers, G., Lee, A., -, "Information Security" National Institute of Standards and Technology (NIST) Special Publication 800-53 Recommended Security Controls for Federal Information Systems
- [SP 800-55] Swanson, M., Bartol, N., Sabato, J., Hash, J., and Graffo, L., "COMPUTER SECURITY" – NIST SP 800-55 Security Metrics Guide for Information Technology Systems (July 2003)
- [SP800-12] An Introduction to Computer Security – The NIST Handbook – Chapter 7 Computer Security Risk Management
- [SP800-100] Bowen, P. hash, J., Wilson, M. "Information Security Handbook: A Guide for Managers" –National Institute of Standards and Technology (NIST) Special Publication 800-100 (October 2006)

- [Steel et al.. 2005] Steel, C., Nagappan, R., Lai, R., Core Security Patterns: Best Practices and Strategies for J2EE Web Services, and Identity Management. Prentice Hall, (2005)
- [Tang & MacLennan 2005] Tang, Z., MacLennan, J., Data Mining with SQL Server 2005 Wiley, (2005)
- [Tsipenyuk et al.. 2005] Tsipenyuk, K., Chess, B., and McGraw, G., “Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors,” IEEE Security and Privacy (November/December 2005)
- [Verdon and McGraw 2004] Verdon, D, McGraw, G., “Risk Analysis in Software Design.” IEEE Security and Privacy 2.4 (2004)
- [Viega and McGraw 2001] Viega, J., and McGraw, G.,. Building Secure Software: How to Avoid Security Problems the Right Way. Addison-Wesley Boston, MA.(2001)
- [Voas et al.. 1997] Voas, J., McGraw, G., Kassab, L., Voas, L., “A ‘Crystal Ball’ for Software Liability” IEEE Computer (June 1997) 29 – 36
- [Voas et al. 1998] Voas, J., and McGraw, G.,. *Software Fault Injection: Inoculating Programs Against Errors*, 47-48. New York, NY: John Wiley & Sons, 1998.
- [Walker et al.. 2006] Walker, Robert J., Holmes, R., Hedgeland, I., Kapur, P., Smith, A., “A Lightweight Approach to Technical Risk Estimation via Probabilistic Impact Analysis” *International Conference on Software Engineering Proceedings of the 2006 international workshop on Mining software repositories* Shanghai, China (2006)
- [Wallace 1991] Wallace, D.R., Kuhn, D.R., Cherniavsky, J.C., "Report on a Workshop on the Assurance of High Integrity Software," *Proceedings of the Sixth Annual Conference on Computer Assurance (COMPASS 1991)*, NIST, Gaithersburg, MD, June 24-27, 1991 , The Institute of Electrical and Electronics Engineers, (1991).
- [Weingberg and Schulman 1974] Weinberg, G. M., Schulman, E. M., “Goals and Performance in Computer Programming,” Human Factors, (1974), 16(1), 70-77.
- [Young 2001] Young, Peter C. and Tippins, Steven C. Managing Business Risk: An Organization-Wide Approach to Risk Management. New York, NY: American Management Association, (2001) (ISBN: 0-814-40461-8).

APPENDICES

Appendix A-1 – The NVD Schema

This appendix gives the NVD Schema from which SSRAM derives its schema.

Table structural data and attribute types used in SSRAM are based on the NVD Schema shown in this appendix.

```
<xs:simpleType name="CVSSVector" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation>
      simpleType to describe the CVSS Base Vector
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="(AV:[RL]/AC:[HL]/Au:(R|NR)/C:[NPC]/I:[NPC]/A:[NPC]/B:[NCIA])\ (Approximated)?" />
  </xs:restriction>
</xs:simpleType><xs:simpleType name="dateType" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation>
      Defines date format for NVD. Dates follow the mask "yyyy-mm-dd"
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:date">
    <xs:pattern value="(19|20)\d\d-((01|03|05|07|08|10|12)-(0[1-9]||1-2]\d|3[01])|(04|06|09|11)-(0[1-9]||1-2]\d|30)|02-(0[1-9]||1\d|2\d))" />
  </xs:restriction>
</xs:simpleType><xs:simpleType name="trueOnlyAttribute" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation>
      simpleType used for attributes that are only present when they are true.
      Such attributes appear only in the form attribute_name="1".
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="1" />
  </xs:restriction>
</xs:simpleType><xs:simpleType name="zeroToTen" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation>
      simpleType used when scoring on a scale of 0-10, inclusive
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:decimal">
    <xs:minInclusive value="0" fixed="true" />
    <xs:maxInclusive value="10" fixed="true" />
  </xs:restriction>
</xs:simpleType><xs:simpleType name="urlType" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation>
      Restricts urls in NVD beyond the xs:anyURI restrictions.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:anyURI">
    <xs:whiteSpace value="collapse" />
    <xs:pattern value="((news|ht|f|tp(s)?):\/\/([^\:]|[:\/?])+(:\/)?)+\/" />
  </xs:restriction>
</xs:simpleType>
```

```

</xs:restriction>
</xs:simpleType><xs:element name="nvd" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation>
      Root element. Contains only "entry" child elements.
      Attributes for this element describe the version of the XML feed being read.
      Attributes:
        "nvd_xml_version" (required) => the schema and DTD version number currently supported by this document
        "pub_date" (required) => the date this document was compiled
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>
            Documents one CVE entry. The child elements should always appear
            in the sequence defined below. These elements are compatible with
            entry elements from the CVE XML feeds.
            Attributes:
              "type" (required) => CVE or CAN
              "name" (required) => full CVE name
              "seq" (required) => sequence number from CVE name
              "nvd_name" => NVD name (if it exists)
              "discovered" => date discovered
              "published" (required) => date published
              "modified" => date modified
              "severity" => severity as determined by NVD analysts: High, Medium, or Low
              "reject" => indicates that this CVE entry has been rejected by CVE or NVD
              "CVSS_score" => CVSS Severity Score
              "CVSS_vector" => CVSS Base Vector
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="desc">
                <xs:annotation>
                  <xs:documentation>
                    Description wrapper tag, parent to any documented descriptions of this CVE entry.
                    While the "desc" tag will always be present, there may be no "descript" child tags.
                    Only one "descript" tag will exist for each description source (i.e. CVE, NVD, ...).
                  </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="descript" minOccurs="0" maxOccurs="2">
                      <xs:annotation>
                        <xs:documentation>
                          Contains a specific description of this CVE entry from source
                          indicated by the "source" attribute.
                        </xs:documentation>
                      </xs:annotation>
                      <xs:complexType mixed="true">
                        <xs:attribute name="source" use="required">
                          <xs:simpleType>
                            <xs:restriction base="xs:NMTOKEN">
                              <xs:enumeration value="cve" />
                              <xs:enumeration value="nvd" />
                            </xs:restriction>
                          </xs:simpleType>
                        </xs:attribute>

```

```

        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="impacts" minOccurs="0">
  <xs:annotation>
    <xs:documentation>
      Impact wrapper tag (may or may not be present). Only one "impact" tag will exist
      for each impact explanation source.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="impact">
        <xs:annotation>
          <xs:documentation>
            Contains a specific impact explanation of this CVE entry from
            source indicated by the "source" attribute.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType mixed="true">
          <xs:attribute name="source" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="nvd" />
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="sols" minOccurs="0">
  <xs:annotation>
    <xs:documentation>
      Solution wrapper tag (may or may not be present). Only one "sol" tag will exist
      for each solution explanation source.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="sol">
        <xs:annotation>
          <xs:documentation>
            Contains a specific solution explanation of this CVE entry from
            source indicated by the "source" attribute.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType mixed="true">
          <xs:attribute name="source" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="nvd" />
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:sequence>

```

```

    </xs:complexType>
  </xs:element>
  <xs:element name="loss_types" minOccurs="0">
    <xs:annotation>
      <xs:documentation>
        Loss type tag (may or may not be present). Contains one loss type child for each loss
        type of this CVE entry.
        Potential loss types are:
        "avail" => availability
        "conf" => confidentiality
        "int" => integrity
        "sec_prot" => security protection
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="avail" minOccurs="0" />
        <xs:element name="conf" minOccurs="0" />
        <xs:element name="int" minOccurs="0" />
        <xs:element name="sec_prot" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              Security Protection tag with one attribute for each security protection type.
              Potential security protection types are:
              "admin" => gain administrative access
              "user" => gain user access
              "other" => other
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:attribute name="admin" type="trueOnlyAttribute" />
            <xs:attribute name="user" type="trueOnlyAttribute" />
            <xs:attribute name="other" type="trueOnlyAttribute" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="vuln_types" minOccurs="0">
    <xs:annotation>
      <xs:documentation>
        Vulnerability type tag (may or may not be present). Contains one vulnerability type
        child for each vulnerability type of this CVE entry.
        Potential vulnerability types are:
        "access" => Access validation error
        "input" => Input validation error
        "design" => Design error
        "exception" => Exceptional condition error
        "env" => Environmental error
        "config" => Configuration error
        "race" => Race condition error
        "other" => other
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="access" minOccurs="0" />
        <xs:element name="input" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              Input validation error tag with one attribute for each input validation error type.

```

Potential input validation error types are:

"bound" => Boundary condition error

"buffer" => Buffer overflow

</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:attribute name="bound" type="trueOnlyAttribute" />

<xs:attribute name="buffer" type="trueOnlyAttribute" />

</xs:complexType>

</xs:element>

<xs:element name="design" minOccurs="0" />

<xs:element name="exception" minOccurs="0" />

<xs:element name="env" minOccurs="0" />

<xs:element name="config" minOccurs="0" />

<xs:element name="race" minOccurs="0" />

<xs:element name="other" minOccurs="0" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="range" minOccurs="0">

<xs:annotation>

<xs:documentation>

Vulnerability range tag (may or may not be present). Contains one vulnerability range child for each vulnerability range of this CVE entry.

Potential vulnerability ranges are:

"local" => Locally exploitable

"remote" => Remotely exploitable

"user_init" => User accesses attacker

</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:sequence>

<xs:element name="local" minOccurs="0" />

<xs:element name="remote" minOccurs="0" />

<xs:element name="user_init" minOccurs="0" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="refs">

<xs:annotation>

<xs:documentation>

Reference wrapper tag (always present). External references to this CVE entry are contained within this tag.

</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:sequence>

<xs:element name="ref" minOccurs="0" maxOccurs="unbounded">

<xs:annotation>

<xs:documentation>

Individual reference to this CVE entry. Text is the name of this vulnerability at this particular reference.

Attributes:

"source" (required) => Name of reference source

"url" (required) => hyperlink to reference

"sig" => indicates this reference includes a tool signature

"adv" => indicates this reference is a Security Advisory

"patch" => indicates this reference includes a patch for this vulnerability

</xs:documentation>

</xs:annotation>

<xs:complexType mixed="true">


```

        <xs:attribute name="source" type="xs:string" use="required" />
        <xs:attribute name="url" type="urlType" use="required" />
        <xs:attribute name="sig" type="trueOnlyAttribute" />
        <xs:attribute name="adv" type="trueOnlyAttribute" />
        <xs:attribute name="patch" type="trueOnlyAttribute" />
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="vuln_soft" minOccurs="0">
    <xs:annotation>
    <xs:documentation>
Vulnerable software wrapper tag (may or may not be present). Software affected by this CVE
entry are listed within this tag.
    </xs:documentation>
    </xs:annotation>
    <xs:complexType>
    <xs:sequence>
        <xs:element name="prod" maxOccurs="unbounded">
            <xs:annotation>
            <xs:documentation>
Product wrapper tag. Versions of this product that are affected by this
vulnerability are listed within this tag.
Attributes:
"name" => Product name
"vendor" => Vendor of this product
            </xs:documentation>
            </xs:annotation>
            <xs:complexType>
            <xs:sequence>
                <xs:element name="vers" maxOccurs="unbounded">
                    <xs:annotation>
                    <xs:documentation>
Represents a version of this product that is affected by this vulnerability.
Attributes:
"num" => This version number
                    </xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                    <xs:attribute name="num" type="xs:string" use="required" />
                    <xs:attribute name="prev" type="trueOnlyAttribute" />
                    </xs:complexType>
                    </xs:element>
                </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required" />
            <xs:attribute name="vendor" type="xs:string" use="required" />
            </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="type" use="required">
<xs:simpleType>
    <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="CAN" />
        <xs:enumeration value="CVE" />
    </xs:restriction>
</xs:simpleType>

```

"prev" => Indicates that versions previous to this version number are also affected by this vulnerability

```

</xs:attribute>
<xs:attribute name="name" use="required">
<xs:simpleType>
  <xs:restriction base="xs:ID">
    <xs:pattern value="(CAN|CVE)\-\d\d\d\d\d\d\d\d" />
  </xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="seq" use="required">
<xs:simpleType>
  <xs:restriction base="xs:NMTOKEN">
    <xs:pattern value="\d\d\d\d\d\d\d\d" />
  </xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="nvd_name" type="xs:string" />
<xs:attribute name="discovered" type="dateType" />
<xs:attribute name="published" type="dateType" use="required" />
<xs:attribute name="modified" type="dateType" />
<xs:attribute name="severity">
<xs:simpleType>
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="High" />
    <xs:enumeration value="Medium" />
    <xs:enumeration value="Low" />
  </xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="reject" type="trueOnlyAttribute" />
<xs:attribute name="CVSS_score" type="zeroToTen" />
<xs:attribute name="CVSS_vector" type="CVSSVector" />
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="nvd_xml_version" type="xs:NMTOKEN" use="required" />
<xs:attribute name="pub_date" type="dateType" use="required" />
</xs:complexType>
</xs:element>

```

Appendix A-2 – An Example of an NVD entry

This appendix shows an example of an NVD entry that had to be parsed. In all, 20,804 entries were loaded into SSRAM from 1996 – 2006 to be used for training and testing of data. The historical data used was divided into 2 groups. 1996 – 2001 for training to predict 2002 data, and 2003-2005 to predict 2006.

```
<entry type="CVE" name="CVE-2006-0948" seq="2006-0948" published="2006-08-21" modified="2006-08-22" severity="High" CVSS_score="7.0" CVSS_vector="(AV:L/AC:L/Au:NR/C:C/I:C/A:C/B:N)">
  <desc>
    <descript source="cve">AOL 9.0 Security Edition revision 4184.2340, and probably other versions, uses insecure permissions (Everyone/Full Control) for the "America Online 9.0" directory, which allows local users to gain privileges by replacing critical files.</descript>
  </desc>
  <sols>
    <sol source="nvd">AOL has released fixes to address this issue. These fixes can be automatically applied by logging in to the service.</sol>
  </sols>
  <loss_types>
    <sec_prot admin="1" />
  </loss_types>
  <vuln_types>
    <design />
  </vuln_types>
  <range>
    <local />
  </range>
  <refs>
    <ref source="BID" url="http://www.securityfocus.com/bid/19583" patch="1">19583</ref>
    <ref source="BUGTRAQ" url="http://www.securityfocus.com/archive/1/archive/1/443622/100/0/threaded" adv="1">20060818 Secunia Research: AOL Insecure Default Directory Permissions</ref>
    <ref source="FRSIRT" url="http://www.frsirt.com/english/advisories/2006/3317" adv="1" patch="1">ADV-2006-3317</ref>
    <ref source="SECTrack" url="http://securitytracker.com/id?1016717" patch="1">1016717</ref>
```

```
<ref source="SECUNIA" url="http://secunia.com/advisories/18734" adv="1"
patch="1">18734</ref>
<ref source="XF" url="http://xforce.iss.net/xforce/xfdb/28445" patch="1">aol-default-
insecure-permissions(28445)</ref>
</refs>
- <vuln_soft>
- <prod name="AOL Security Edition" vendor="AOL">
  <vers num="9.0 4184.2340" />
</prod>
</vuln_soft>
</entry>
```

Appendix A-3 – Script for Parsing XML document

The code shown in this appendix was used to load the xml document feeds and parse the xml document into corresponding tables in SSRAM.

```
use ssram
-- -----Insert XML file into nvdtmp
INSERT nvdtmp
    SELECT CONVERT(xml, BulkColumn, 2) FROM
        OPENROWSET(Bulk 'F:\Dissertation\Databases\National Vulnerability
Database\nvdcve-2002.xml', SINGLE_BLOB) [rowsetresults]
--    OPENROWSET(Bulk 'F:\Dissertation\Databases\National Vulnerability
Database\nvdcve-recent.xml', SINGLE_BLOB) [rowsetresults]

Go
-- ----- Drop and create SSRAM tables
USE [SSRAM]
GO
--Drop entryDesc

IF EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_entryDesc_entry]') AND parent_object_id =
OBJECT_ID(N'[dbo].[entryDesc]'))
ALTER TABLE [dbo].[entryDesc] DROP CONSTRAINT [FK_entryDesc_entry]
GO
USE [SSRAM]
GO
/***** Object: Table [dbo].[entryDesc]      Script Date: 02/05/2008
10:43:52 *****/
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[entryDesc]') AND type in (N'U'))
    DROP TABLE [dbo].[entryDesc]
---Drop entrySols

USE [SSRAM]
GO
IF EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_entrySols_entry]') AND parent_object_id =
OBJECT_ID(N'[dbo].[entrySols]'))
    ALTER TABLE [dbo].[entrySols] DROP CONSTRAINT
[FK_entrySols_entry]
GO
USE [SSRAM]
GO
/***** Object: Table [dbo].[entrySols]      Script Date: 02/05/2008
10:55:14 *****/
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[entrySols]') AND type in (N'U'))
    DROP TABLE [dbo].[entrySols]
```

```

---drop lossType2

USE [SSRAM]
GO
IF EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_lossType2_entry]') AND parent_object_id =
OBJECT_ID(N'[dbo].[lossType2]'))
ALTER TABLE [dbo].[lossType2] DROP CONSTRAINT [FK_lossType2_entry]
GO
USE [SSRAM]
GO
/***** Object: Table [dbo].[lossType2] Script Date: 02/05/2008
10:56:52 *****/
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[lossType2]') AND type in (N'U'))
DROP TABLE [dbo].[lossType2]

--Drop nvdtmp

USE [SSRAM]
GO
/***** Object: Table [dbo].[nvdtmp] Script Date: 02/05/2008
10:58:49 *****/
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[nvdtmp]') AND type in (N'U'))
DROP TABLE [dbo].[nvdtmp]
--
---DROP range

USE [SSRAM]
GO
IF EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_range_entry]') AND parent_object_id =
OBJECT_ID(N'[dbo].[range]'))
ALTER TABLE [dbo].[range] DROP CONSTRAINT [FK_range_entry]
GO
USE [SSRAM]
GO
/***** Object: Table [dbo].[range] Script Date: 02/05/2008
10:59:59 *****/
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[range]') AND type in (N'U'))
DROP TABLE [dbo].[range]

--- DROP refs

USE [SSRAM]
GO
IF EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_refs_entry]') AND parent_object_id =
OBJECT_ID(N'[dbo].[refs]'))
ALTER TABLE [dbo].[refs] DROP CONSTRAINT [FK_refs_entry]
GO

```

```

USE [SSRAM]
GO
/***** Object:  Table [dbo].[refs]      Script Date: 02/05/2008 11:01:17
*****/
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[refs]') AND type in (N'U'))
DROP TABLE [dbo].[refs]

--DROP vuln_soft

USE [SSRAM]
GO
IF EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_vuln_soft_entry]') AND parent_object_id =
OBJECT_ID(N'[dbo].[vuln_soft]'))
ALTER TABLE [dbo].[vuln_soft] DROP CONSTRAINT [FK_vuln_soft_entry]
GO
USE [SSRAM]
GO
/***** Object:  Table [dbo].[vuln_soft]  Script Date: 02/05/2008
11:03:26 *****/
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[vuln_soft]') AND type in (N'U'))
DROP TABLE [dbo].[vuln_soft]

---DROP vuln_types

USE [SSRAM]
GO
IF EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_vuln_types_entry]') AND parent_object_id =
OBJECT_ID(N'[dbo].[vuln_types]'))
ALTER TABLE [dbo].[vuln_types] DROP CONSTRAINT [FK_vuln_types_entry]
GO
USE [SSRAM]
GO
/***** Object:  Table [dbo].[vuln_types]  Script Date: 02/05/2008
11:04:17 *****/
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[vuln_types]') AND type in (N'U'))
DROP TABLE [dbo].[vuln_types]

---DROP entry

-----ENTRY TABLE DROP AND CREATE
/***** Object:  Table [dbo].[entry]      Script Date: 02/05/2008
10:42:30 *****/
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[entry]') AND type in (N'U'))
    DROP TABLE [dbo].[entry]

```

```

/***** Object: Table [dbo].[entry]      Script Date: 02/05/2008
10:43:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[entry](
    [type] [varchar](5) COLLATE SQL_Latin1_General_CP1_CI_AS NOT
NULL,
    [name] [varchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT
NULL,
    [seq] [varchar](12) COLLATE SQL_Latin1_General_CP1_CI_AS NOT
NULL,
    [nvd_name] [varchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [discovered] [datetime] NULL,
    [published] [datetime] NOT NULL,
    [modified] [datetime] NULL,
    [severity] [varchar](10) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [reject] [varchar](10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [CVSS_score] [real] NULL,
    [CVSS_vector] [varchar](25) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    CONSTRAINT [PK_entry] PRIMARY KEY CLUSTERED
(
    [name] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF

```

```

-- CREATE entryDesc
USE [SSRAM]
GO
/***** Object: Table [dbo].[entryDesc]  Script Date: 02/05/2008
10:53:52 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[entryDesc](
    [entry_name] [varchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
    [descript] [varchar](max) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [source] [varchar](5) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,

```



```

        CONSTRAINT [PK_entryDesc] PRIMARY KEY CLUSTERED
    (
        [entry_name] ASC
    )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
    ) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description',
    @value=N'associated with entry.name (foreign key)'
    ,@level0type=N'SCHEMA', @level0name=N'dbo', @level1type=N'TABLE',
    @level1name=N'entryDesc', @level2type=N'COLUMN',
    @level2name=N'entry_name'

GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description',
    @value=N'Contains a specific description of this CVE entry from source
    indicated by the "source" attribute' ,@level0type=N'SCHEMA',
    @level0name=N'dbo', @level1type=N'TABLE', @level1name=N'entryDesc',
    @level2type=N'COLUMN', @level2name=N'descript'

GO
EXEC sys.sp_addextendedproperty @name=N'MS_Description',
    @value=N'enumerated type "cve" or "nvd" - required for each descript'
    ,@level0type=N'SCHEMA', @level0name=N'dbo', @level1type=N'TABLE',
    @level1name=N'entryDesc', @level2type=N'COLUMN', @level2name=N'source'

GO
USE [SSRAM]
GO
ALTER TABLE [dbo].[entryDesc] WITH CHECK ADD CONSTRAINT
    [FK_entryDesc_entry] FOREIGN KEY([entry_name])
    REFERENCES [dbo].[entry] ([name])

-----entrySols Table

USE [SSRAM]
GO
/***** Object: Table [dbo].[entrySols]      Script Date: 02/05/2008
10:55:56 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[entrySols](
    [entry_name] [varchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS
    NOT NULL,
    [sol] [varchar](max) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [source] [varchar](5) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    CONSTRAINT [PK_entrySols] PRIMARY KEY CLUSTERED
    (

```

```

        [entry_name] ASC
    )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
    ) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
USE [SSRAM]
GO
ALTER TABLE [dbo].[entrySols] WITH CHECK ADD CONSTRAINT
[FK_entrySols_entry] FOREIGN KEY([entry_name])
REFERENCES [dbo].[entry] ([name])

-----create lossType

USE [SSRAM]
GO
/***** Object: Table [dbo].[lossType2]      Script Date: 02/05/2008
10:57:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[lossType2] (
    [entry_name] [varchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
    [loss_type] [varchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
    [adminSP] [char](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [userSP] [char](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [otherSP] [char](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    CONSTRAINT [PK_lossType2] PRIMARY KEY CLUSTERED
(
    [entry_name] ASC,
    [loss_type] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
USE [SSRAM]
GO
ALTER TABLE [dbo].[lossType2] WITH CHECK ADD CONSTRAINT
[FK_lossType2_entry] FOREIGN KEY([entry_name])
REFERENCES [dbo].[entry] ([name])

-----nvdtm

USE [SSRAM]

```

```

GO
/***** Object:  Table [dbo].[nvdtmp]      Script Date: 02/05/2008
10:58:18 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[nvdtmp] (
    [XmlCol] [xml] NULL
) ON [PRIMARY]

-----range

USE [SSRAM]
GO
/***** Object:  Table [dbo].[range]      Script Date: 02/05/2008
11:00:42 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[range] (
    [entry_name] [varchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
    [exploit_range] [varchar](15) COLLATE
SQL_Latin1_General_CP1_CI_AS NOT NULL,
    CONSTRAINT [PK_range] PRIMARY KEY CLUSTERED
(
    [entry_name] ASC,
    [exploit_range] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
USE [SSRAM]
GO
ALTER TABLE [dbo].[range] WITH CHECK ADD CONSTRAINT [FK_range_entry]
FOREIGN KEY([entry_name])
REFERENCES [dbo].[entry] ([name])

----refs

USE [SSRAM]
GO
/***** Object:  Table [dbo].[refs]      Script Date: 02/05/2008 11:01:37
*****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```

SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[refs](
    [entry_name] [varchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
    [ref] [nvarchar](max) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [source] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [url] [nvarchar](100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT
NULL,
    [sig] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [adv] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [patch] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    CONSTRAINT [PK_refs] PRIMARY KEY CLUSTERED
(
    [entry_name] ASC,
    [url] ASC
)WITH (IGNORE_DUP_KEY = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
USE [SSRAM]
GO
ALTER TABLE [dbo].[refs] WITH CHECK ADD CONSTRAINT [FK_refs_entry]
FOREIGN KEY([entry_name])
REFERENCES [dbo].[entry] ([name])

-----vuln_soft

USE [SSRAM]
GO
/***** Object: Table [dbo].[vuln_soft] Script Date: 02/05/2008
11:03:47 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[vuln_soft](
    [entry_name] [varchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
    [prodName] [nvarchar](25) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
    [prodVendor] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
    [versionNum] [nvarchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
    [preVersion] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [edition] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    CONSTRAINT [PK_vuln_soft] PRIMARY KEY CLUSTERED
(

```

```

        [entry_name] ASC,
        [prodName] ASC,
        [prodVendor] ASC,
        [versionNum] ASC
    )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
    ) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
USE [SSRAM]
GO
ALTER TABLE [dbo].[vuln_soft] WITH CHECK ADD CONSTRAINT
[FK_vuln_soft_entry] FOREIGN KEY([entry_name])
REFERENCES [dbo].[entry] ([name])

--vuln_types

USE [SSRAM]
GO
/***** Object: Table [dbo].[vuln_types]      Script Date: 02/05/2008
11:04:49 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[vuln_types](
    [entry_name] [varchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
    [vuln_type] [varchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
    [input_bound] [char](1) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [input_buffer] [char](1) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    CONSTRAINT [PK_vuln_types] PRIMARY KEY CLUSTERED
(
        [entry_name] ASC,
        [vuln_type] ASC
    )WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
    ) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
USE [SSRAM]
GO
ALTER TABLE [dbo].[vuln_types] WITH CHECK ADD CONSTRAINT
[FK_vuln_types_entry] FOREIGN KEY([entry_name])
REFERENCES [dbo].[entry] ([name])

```

```

USE [SSRAM]
GO
/***** Object:  Table [dbo].[denormNVD]      Script Date: 02/05/2008
11:07:28 *****/
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[denormNVD]') AND type in (N'U'))
DROP TABLE [dbo].[denormNVD]

USE [SSRAM]
GO
/***** Object:  Table [dbo].[denormNVD]      Script Date: 02/05/2008
11:08:06 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[denormNVD] (
    [name] [varchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT
NULL,
    [discovered] [datetime] NULL,
    [published] [datetime] NOT NULL,
    [cvss_score] [real] NULL,
    [descript] [varchar](max) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [sol] [varchar](max) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [loss_type] [varchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [adminSP] [char](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [userSP] [char](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [otherSP] [char](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [vuln_type] [varchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [input_bound] [char](1) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [input_buffer] [char](1) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [exploit_range] [varchar](15) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
    [ref] [nvarchar](max) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [source] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [sig] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [adv] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [patch] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [prodName] [nvarchar](25) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [versionNum] [nvarchar](15) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [preVersion] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [edition] [nchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]

```

```

GO
SET ANSI_PADDING OFF

-- -----Parse Entry-----

-- to be used by the sp_xml_prepared document to shred the nvd xml file
DECLARE @idoc int
DECLARE @doc xml

-- Get the nvd xml content for parsing
-- note that set @ doc accepts any valid expression, a select
statement had to be put in parenthesis to work
set @doc = (select XmlCol from dbo.nvdtmp)

-- create and internal representation of the XML document
EXEC sp_xml_preparedocument @idoc OUTPUT, @doc

-- Use OPENXML to provide rowset consisting of entries for nvd
insert entry
select * from OPENXML(@idoc, '/nvd/entry')
with entry

EXEC sp_xml_removedocument @idoc

```

```

----- Parse EntryDesc -----
use ssram
-- to be used by the sp_xml_prepared document to shred the nvd xml file
DECLARE @idoc int
DECLARE @doc xml

-- Get the nvd xml content for parsing
-- note that set @ doc accepts any valid expression, a select
statement had to be
-- put in parenthesis to work
set @doc = (select top 1 XmlCol from dbo.nvdtmp)

-- create and internal representation of the XML document
EXEC sp_xml_preparedocument @idoc OUTPUT, @doc

-- Use OPENXML to provide rowset consisting of descriptions for each
entry in the nvd
-- Note the use of 'with' and '../../@name to get the grandparent node
attribute and the use
-- of ntext 'text () ' to get the element value for the description
insert entryDesc
select * from OPENXML(@idoc, '/nvd/entry/desc/descript', 3)
with (entry_name varchar (15) '../../@name',
      descript          ntext 'text ()',
      source            varchar (max) '@source')

EXEC sp_xml_removedocument @idoc

```

```

-- ----- Parse into entrySols -----

```

```

use ssram
-- to be used by the sp_xml_prepared document to shred the nvd xml file
DECLARE @idoc int
DECLARE @doc xml

-- Get the nvd xml content for parsing
-- note that set @ doc accepts any valid expression, a select
statement had to be
-- put in parenthesis to work
-- set @doc = (select top 1 XmlCol from dbo.nvdtmp)
set @doc = (select XmlCol from dbo.nvdtmp)

-- create and internal representation of the XML document
EXEC sp_xml_preparedocument @idoc OUTPUT, @doc

-- Use OPENXML to provide rowset consisting of descriptions for each
entry in the nvd
-- Note the use of 'with' and '../../@name to get the grandparent node
attribute and the use
-- of ntext 'text () ' to get the element value for the description
insert entrySols
select * from OPENXML(@idoc, '/nvd/entry/sols/sol', 3)
    with (entry_name varchar (15) '../../@name',
          sol ntext 'text ()',
          source varchar (5) '@source')

EXEC sp_xml_removedocument @idoc

```

```

-- -----Parse LossType -----
use ssram
-- to be used by the sp_xml_prepared document to shred the nvd xml file
DECLARE @idoc int
DECLARE @doc xml

-- Get the nvd xml content for parsing
-- note that set @ doc accepts any valid expression, a select
statement had to be
-- put in parenthesis to work
--set @doc = (select top 1 XmlCol from dbo.nvdtmp)
set @doc = (select XmlCol from dbo.nvdtmp)

-- create and internal representation of the XML document
EXEC sp_xml_preparedocument @idoc OUTPUT, @doc

-- Note use of meta property @mp:localname to get the value for the
loss_type element
-- for availability loss type
insert lossType2
    select * from OPENXML(@idoc, '/nvd/entry/loss_types/avail',9)
        with (entry_name varchar (15) '../../@name',
              loss_type varchar (15) '@mp:localname',

```



```

            adminSP          char (1)      '@admin',
            userSP           char (1)      '@user',
            otherSP          char (1)      '@other')
-- for confidentiality loss type
insert lossType2
    select * from OPENXML(@idoc, '/nvd/entry/loss_types/conf',9)
        with (entry_name varchar (15) '../../@name',
            loss_type      varchar (15) '@mp:localname',
            adminSP        char (1)      '@admin',
            userSP         char (1)      '@user',
            otherSP        char (1)      '@other')

-- for integrity loss type
insert lossType2
    select * from OPENXML(@idoc, '/nvd/entry/loss_types/int',9)
        with (entry_name varchar (15) '../../@name',
            loss_type      varchar (15)      '@mp:localname',
            adminSP        char (1)      '@admin',
            userSP         char (1)      '@user',
            otherSP        char (1)      '@other')

-- for security protection loss type
insert lossType2
    select * from OPENXML(@idoc, '/nvd/entry/loss_types/sec_prot',9)
        with (entry_name varchar (15) '../../@name',
            loss_type      varchar (15)      '@mp:localname',
            adminSP        char (1)      '@admin',
            userSP         char (1)      '@user',
            otherSP        char (1)      '@other')

--
---- admin attribute for security protection loss type
--update lossType2
--set adminSP = LS.adminSP
--from
--(
--    select * from OPENXML(@idoc, '/nvd/entry/loss_types/sec_prot', 3)
--        with (entry_name varchar (15) '../../@name',
--            adminSP      nchar (10)    '@admin')
--    ) as LS
--where lossType2.entry_name = LS.entry_name
--
---- user attribute for security protection loss type
--update lossType2
--set userSP = LS.userSP
--from
--(
--    select * from OPENXML(@idoc, '/nvd/entry/loss_types/sec_prot', 3)
--        with (entry_name varchar (15) '../../@name',
--            userSP      nchar (10)    '@user')
--    ) as LS
--where lossType2.entry_name = LS.entry_name
--
---- other attribute for security protection loss type
--update lossType2

```

```
--set userSP = LS.userSP
--from
--(
--    select * from OPENXML(@idoc, '/nvd/entry/loss_types/sec_prot', 3)
--        with (entry_name varchar (15) '../../@name',
--              otherSP      nchar (10)   '@other')
--) as LS
--where lossType2.entry_name = LS.entry_name
--
```

```
EXEC sp_xml_removedocument @idoc
```

```
-- -----Parse Vuln Types-----
```

```
use ssram
-- to be used by the sp_xml_prepared document to shred the nvd xml file
DECLARE @idoc int
DECLARE @doc xml
```

```
-- Get the nvd xml content for parsing
-- note that set @ doc accepts any valid expression, a select
statement had to be
-- put in parenthesis to work
--set @doc = (select top 1 XmlCol from dbo.nvdtmp)
set @doc = (select XmlCol from dbo.nvdtmp)
```

```
-- create and internal representation of the XML document
EXEC sp_xml_preparedocument @idoc OUTPUT, @doc
```

```
---- Use OPENXML to provide rowset consisting of descriptions for each
entry in the nvd
```

```
--insert vuln_types
--select * from OPENXML(@idoc, '/nvd/entry/vuln_types', 3)
--    with (entry_name varchar (15) '../@name',
--          vuln_type      varchar (10)   '/access',
--          input          varchar (5)   '/input',
--          design         varchar (10)  '/design',
--          exception      varchar (10)  '/exception',
--          env            varchar (10)  '/env',
--          config         varchar (10)  '/config',
--          race           varchar (10)  '/race',
--          other          varchar (10)  '/other')
```

```
-- update the other nodes for vuln_types
insert vuln_types
    select * from OPENXML(@idoc, '/nvd/entry/vuln_types/access',9)
        with (entry_name varchar (15) '../../@name',
              vuln_type      varchar (15) '@mp:localname',
              input_bound    char (1)    '@bound',
```

```

        input_buffer      char   (1)      '@buffer' )

insert vuln_types
select * from OPENXML(@idoc, '/nvd/entry/vuln_types/input',9)
    with (entry_name varchar (15) '../../@name',
          vuln_type   varchar (15) '@mp:localname',
          input_bound char   (1)    '@bound',
          input_buffer char   (1)    '@buffer' )

insert vuln_types
select * from OPENXML(@idoc, '/nvd/entry/vuln_types/design',9)
    with (entry_name varchar (15) '../../@name',
          vuln_type   varchar (15) '@mp:localname',
          input_bound char   (1)    '@bound',
          input_buffer char   (1)    '@buffer' )

insert vuln_types
select * from OPENXML(@idoc, '/nvd/entry/vuln_types/exception',9)
    with (entry_name varchar (15) '../../@name',
          vuln_type   varchar (15) '@mp:localname',
          input_bound char   (1)    '@bound',
          input_buffer char   (1)    '@buffer' )

insert vuln_types
select * from OPENXML(@idoc, '/nvd/entry/vuln_types/env',9)
    with (entry_name varchar (15) '../../@name',
          vuln_type   varchar (15) '@mp:localname',
          input_bound char   (1)    '@bound',
          input_buffer char   (1)    '@buffer' )

insert vuln_types
select * from OPENXML(@idoc, '/nvd/entry/vuln_types/config',9)
    with (entry_name varchar (15) '../../@name',
          vuln_type   varchar (15) '@mp:localname',
          input_bound char   (1)    '@bound',
          input_buffer char   (1)    '@buffer' )

insert vuln_types
select * from OPENXML(@idoc, '/nvd/entry/vuln_types/race',9)
    with (entry_name varchar (15) '../../@name',
          vuln_type   varchar (15) '@mp:localname',
          input_bound char   (1)    '@bound',
          input_buffer char   (1)    '@buffer' )

insert vuln_types
select * from OPENXML(@idoc, '/nvd/entry/vuln_types/other',9)
    with (entry_name varchar (15) '../../@name',
          vuln_type   varchar (15) '@mp:localname',
          input_bound char   (1)    '@bound',
          input_buffer char   (1)    '@buffer' )

EXEC sp_xml_removedocument @idoc

```

```
-- ----- Parse Exploit range -----
```

```

use ssram
-- to be used by the sp_xml_prepared document to shred the nvd xml file
DECLARE @idoc int
DECLARE @doc xml

-- Get the nvd xml content for parsing
-- note that set @ doc accepts any valid expression, a select
statement had to be
-- put in parenthesis to work
--set @doc = (select top 1 XmlCol from dbo.nvdtmp)
set @doc = (select XmlCol from dbo.nvdtmp)

-- create and internal representation of the XML document
EXEC sp_xml_preparedocument @idoc OUTPUT, @doc

-- Use OPENXML to provide rowset consisting of the vulnerability range
for each entry in the nvd
--insert range
--select * from OPENXML(@idoc, '/nvd/entry/range', 3)
--    with (entry_name          varchar (15) '../@name',
--          local              nchar (1)    'local',
--          local_network      nchar (1)    'local_network',
--          network            nchar (1)    'network',
--          user_init          nchar (1)    'user_init')
--

-- update the local nodes for range -- locally exploitable range
insert range
    select * from OPENXML(@idoc, '/nvd/entry/range/local',9)
        with (entry_name varchar (15) '../@name',
              exploit_range varchar (15) '@mp:localname')

-- for local network exploitable range
insert range
    select * from OPENXML(@idoc, '/nvd/entry/range/local_network',9)
        with (entry_name varchar (15) '../@name',
              exploit_range varchar (15) '@mp:localname')

-- for remote exploitable range
insert range
    select * from OPENXML(@idoc, '/nvd/entry/range/remote',9)
        with (entry_name varchar (15) '../@name',
              exploit_range varchar (15) '@mp:localname')

-- for user initiated exploit - where user accesses the attacker
insert range
    select * from OPENXML(@idoc, '/nvd/entry/range/user_init',9)
        with (entry_name varchar (15) '../@name',
              exploit_range varchar (15) '@mp:localname')

EXEC sp_xml_removedocument @idoc

```

```

-- ----- Parse ref node -----

use ssram
-- to be used by the sp_xml_prepared document to shred the nvd xml file
DECLARE @idoc int
DECLARE @doc xml

-- Get the nvd xml content for parsing
-- note that set @ doc accepts any valid expression, a select
statement had to be
-- put in parenthesis to work
--set @doc = (select top 1 XmlCol from dbo.nvdtmp)
set @doc = (select XmlCol from dbo.nvdtmp)
----Used to see what the contents would be like 2/5/08
---- may have to do a nested select or join it with itself to remove
duplicates
----EXEC sp_xml_preparedocument @idoc OUTPUT, @doc
----
----
----select * from OPENXML(@idoc, '/nvd/entry/refs/ref', 3)
---- with (entry_name varchar (15)      '../../@name',
----         ref          nvarchar(max)   'text ()',
----         source       nvarchar (50)    '@source',
----         url          nvarchar (100)   '@url',
----         sig          nchar (1)        '@sig',
----         adv          nchar (1)        '@adv',
----         patch        nchar (1)        '@patch')

-- create and internal representation of the XML document
EXEC sp_xml_preparedocument @idoc OUTPUT, @doc

-- USED TO SHRED INTO TABLE
-- Use OPENXML to provide rowset consisting of external references for
each entry in the nvd
-- Note the use of 'with' and '../../@name to get the grandparent node
attribute and the use
-- of ntext 'text () ' to get the element value for the description
insert refs
select * from OPENXML(@idoc, '/nvd/entry/refs/ref', 3)
      with (entry_name varchar (15)      '../../@name',
            ref          nvarchar(max)   'text ()',
            source       nvarchar (50)    '@source',
            url          nvarchar (100)   '@url',
            sig          nchar (1)        '@sig',
            adv          nchar (1)        '@adv',
            patch        nchar (1)        '@patch')

EXEC sp_xml_removedocument @idoc


```

```

use ssram
-- to be used by the sp_xml_prepared document to shred the nvd xml file
DECLARE @idoc int

```

```

DECLARE @doc xml

-- Get the nvd xml content for parsing
-- note that set @ doc accepts any valid expression, a select
statement had to be
-- put in parenthesis to work
set @doc = (select top 1 XmlCol from dbo.nvdtmp)

-- create and internal representation of the XML document
EXEC sp_xml_preparedocument @idoc OUTPUT, @doc

-- Use OPENXML to provide rowset consisting of software (product and
versions)affected by each entry in the nvd
-- Note the use of 'with' and '../.../@name to get the great-
grandparent node (if it works)attribute and the use
insert vuln_soft
select distinct * from OPENXML(@idoc, '/nvd/entry/vuln_soft/prod/vers',
3)
    with (entry_name  varchar (15)      '../.../@name',
          prodName    nvarchar (25)    '../@name',
          prodVendor  nvarchar (50)    '../@vendor',
          versionNum  nvarchar (15)    '@num',
          preVersion  nchar (1)        '@prev',
          edition     nchar (1)        '@edition')

EXEC sp_xml_removedocument @idoc

```

Appendix A-4 – Validation of Data Upload

This appendix shows the entries that were used to validate that the NVD XML data were parsed and loaded correctly into SSRAM

Records Validated

Record Number	Actual File	NVDtmp File
1-	CVE-2006-0948	CVE-2006-0948
5	CVE-2006-3124	CVE-2006-3124
25	CVE-2006-4257	CVE-2006-4257
45	CVE-2006-4277	CVE-2006-4277
65	CVE-2006-4298	CVE-2006-4298
85	CVE-2006-4319	CVE-2006-4319
105	CVE-2006-4349	CVE-2006-4349
125	CVE-2006-4369	CVE-2006-4369
Last	CVE-2006-4380	CVE-2006-4380

Table 20 - Data Entry Validation Entries

At the end of the first round of validation, we found out that our representation of the loss type and vulnerability type nodes did not adequately reflect the different types of security protection loss and input vulnerabilities. As such, additional fields were added to the *loss_type* and *vuln_type* tables and parsing script corrected to reflect the changes made. We compared again for the previous data and 7 additional records given below to confirm that the data was uploaded correctly.

Record Number	Actual File	NVDtmp File
2	CVE-2006-2122	CVE-2006-2122
22	CVE-2006-4254	CVE-2006-4254
42	CVE-2006-4274	CVE-2006-4274
62	CVE-2006-4295	CVE-2006-4295
82	CVE-2006-4316	CVE-2006-4316
102	CVE-2006-4346	CVE-2006-4346
122	CVE-2006-4366	CVE-2006-4366

Appendix B-1 – Creating Term Vector

We felt that the description attribute of each entry had information that would help in clustering NVD data. This was a non-trivial problem. In order to be able to use this comment-like attribute we had to create term vectors that associated terms with entries, so that we could look at the frequency of occurrence of those terms. To create a term vector so that the comment-like description of vulnerability terms could be used as part of the data mining process involved the following steps.

1. Create a text mining dictionary for the vector table based on the terms in the description attribute for all of the entries.
2. Build Term Vectors based on an association of entries to text mining dictionary terms.

Creating a Text Mining Dictionary for the Vector Table

- 1.1 Create a new DTS (SSIS) package
- 1.2 Rename the package to BuildDictionary.dtsx
- 1.3 Go to Data Flow tab and add a new Data Flow task
- 1.4 In the data flow task, add a “OLE DB Source” transform
 - Connection: create a new for localhost.SSRAM
 - Table: desc
 - Columns:
 -
- 1.5 Add a Data Conversion transform and connect from the OLE DB Source transform
- 1.6 Add a “Term Extraction” transform and connect from the Data Conversion transform
 - Term Type: Noun and Noun Phrase
 - Score Type: TFIDF –
TFIDF - Specify that the score is the TFIDF value of the term. The TFIDF score is the product of Term Frequency and Inverse Document Frequency, defined as: *TFIDF of a Term T = (frequency of T) * log((#rows in Input) / (#rows having T))*
 - Parameters: Frequency=2, Length=10
- 1.7 Add a “Sort” transform and connect it.
 - Sort “Term” in ascending order
 - Don’t pass through Score column

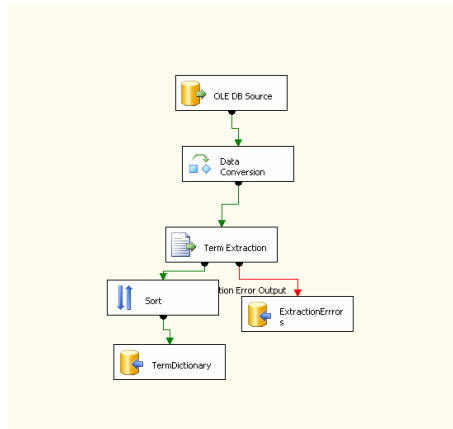
1.8 Add an “OLE DB Destination” transform and connect it.

- Use the connection: localhost.SSRAM
- Click “New” and name it “TermDictionary”
- In Mappings, connect the column, “Term”

1.9 Execute the package

- It automatically enters into debugging mode
- It may take a few minutes

1.10 Stop debugging



Build term vectors

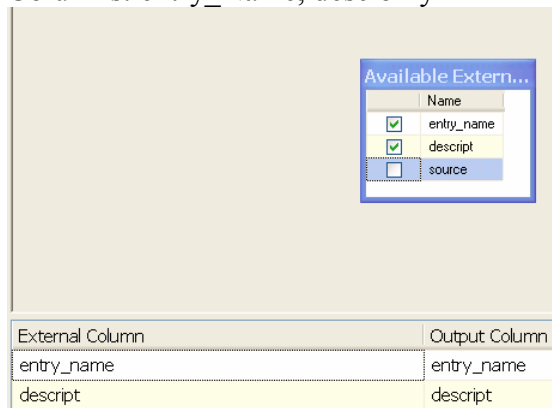
1.11 Create a new DTS (SSIS) package

1.12 Rename the package to BuildTermVectors.dtsx

1.13 Go to Data Flow tab and add a new Data Flow task

1.14 In the data flow task, add a “OLE DB Source” transform

- Connection: create a new for localhost.SSRAM
- Table: entryDesc
- Columns: entry_Name, desc only



1.15 Add a “Term Lookup” transform and connect from the previous transform

- Reference table: TermDictionary

1.16 Add a “Sort” transform and connect it.

- Sort “entry_name” in ascending order, then, “Term” in ascending order, no duplicates

- 1.17 Add an “OLE DB Destination” transform and connect it.
 - Use the connection: localhost.TDM
 - Click “New” and name it “TermVectors”
 - In Mappings, make sure to connect all columns, “Term”, “Frequency”, “ID”
- 1.18 Execute the package
 - It automatically enters into debugging mode
 - It may take a few minutes
- 1.19 Stop debugging
 - PassThru column: entry_name
 - Lookup input column: descript

Reference Table | Term Lookup | Advanced

Specify the connection manager to the reference transformation uses.

OLE DB connection manager:

L1014.SSRAM

Reference table name:

[dbo].[TermDictionary]

Reference Table | Term Lookup | Advanced

Available Input C...

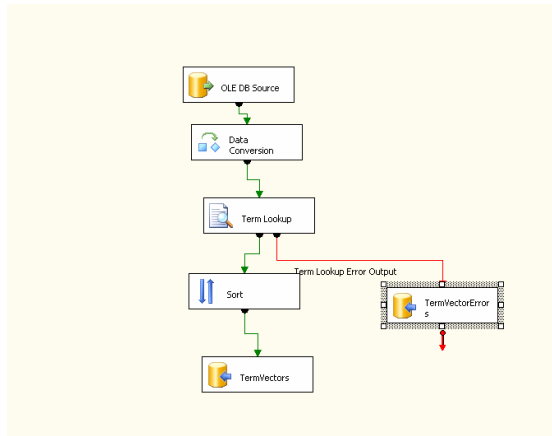
Name
<input checked="" type="checkbox"/> entry_name
<input type="checkbox"/> descript

Available Reference C...

Name
<input checked="" type="checkbox"/> descriptionTerms
Score

Pass-Through Column Output Column Alias

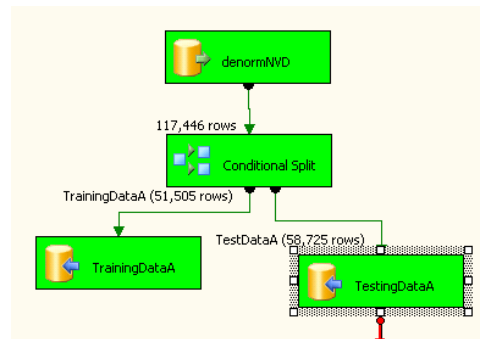
entry_name	entry_name



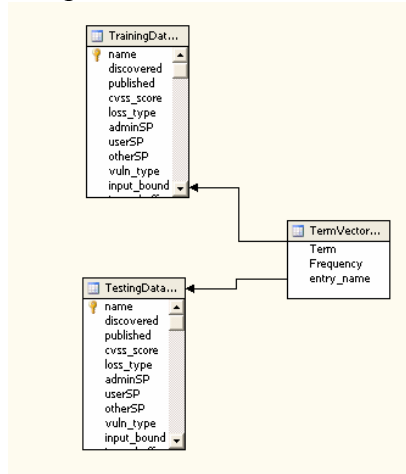
Appendix B-2 – Determining Clustering Algorithm

This appendix shows the process of preparing the data for clustering and setting up the clustering algorithm.

- Create a new prepareSampleData by splitting the data for training and testing. In our case, we used those entries reported between 1996 through 2001 for training and created the test data as those entries reported in 2002.



- - Setup new DataSourceView to be used for Clustering



- Create clustering data mining structure with and without description components
 - With Description
 - Following Fields suggested

Columns related to loss_type:

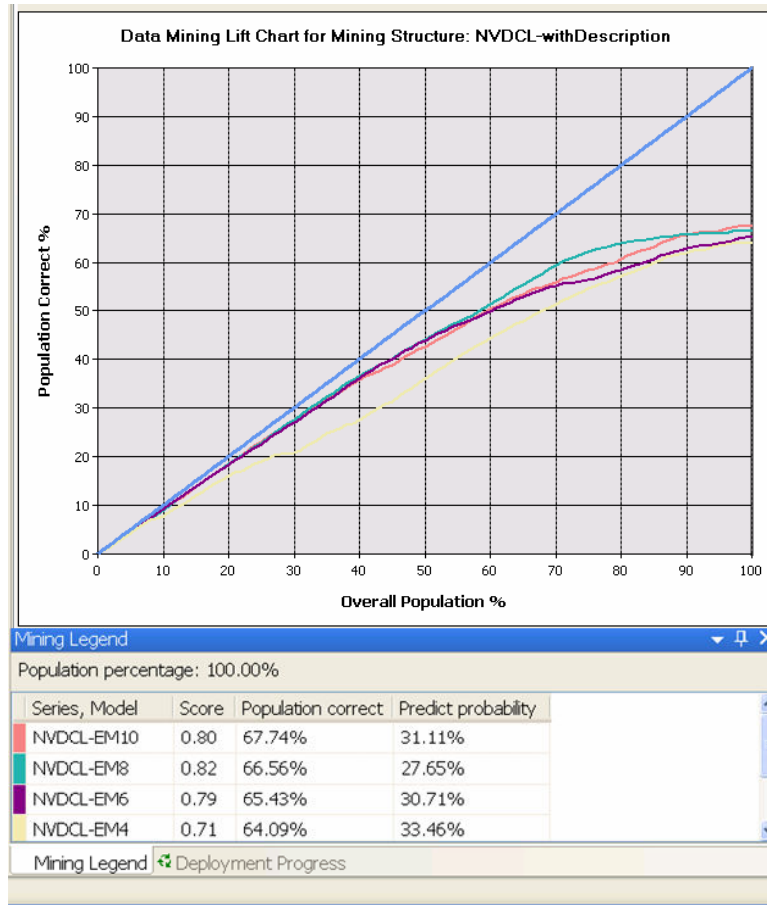
Column Name	Score	Input
cvss_score	0.429	x
adminSP	0.196	x
vuln_type	0.105	x
input_buffer	0.072	x
otherSP	0.061	x
userSP	0.043	
exploit_range	0.027	
input_bound	0.011	
patch	0.005	
sig	0.001	
preVersion	0.001	
adv	0.001	
edition	0.000	
ref	0.000	
discovered	0.000	
versionNum		
prodName		
source		
published		
name		

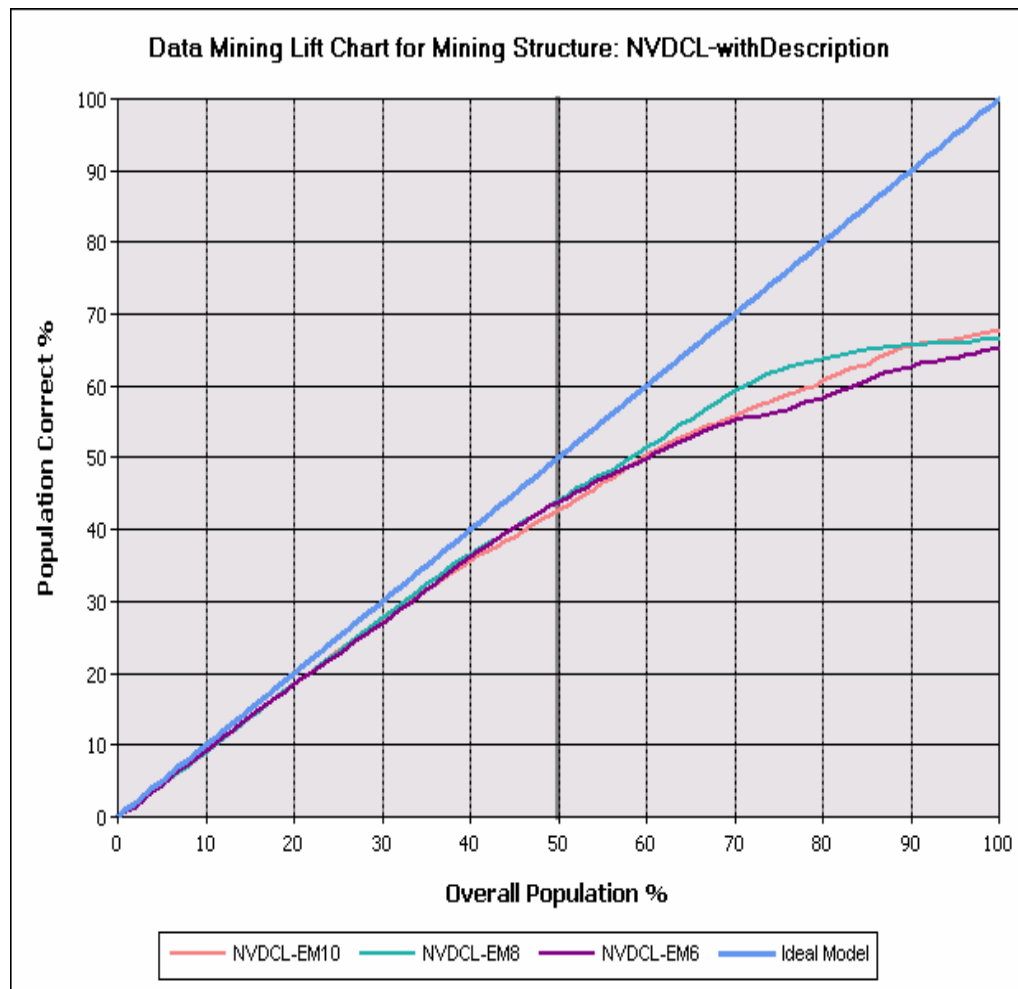
•

Mining model structure:

Tables/Columns	Key	Input	Predictable
TrainingData_96_01			
adminSP	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
adv	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
cvss_score	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
discovered	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
edition	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
exploit_range	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
input_bound	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
input_buffer	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
loss_type	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
name	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
otherSP	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
patch	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
preVersion	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
prodName	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
published	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ref	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
sig	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
source	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
userSP	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
versionNum	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
vuln_type	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
TermVectors			
Frequency	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Term	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

- Created 4 Clustering using EM algorithm with 10,8,6 and 4 nodes.
 - The X-axis of the chart represents the percentage of the test dataset that is used to compare the predictions
 - the Y-axis now represents the percentage of predictions that are correct.



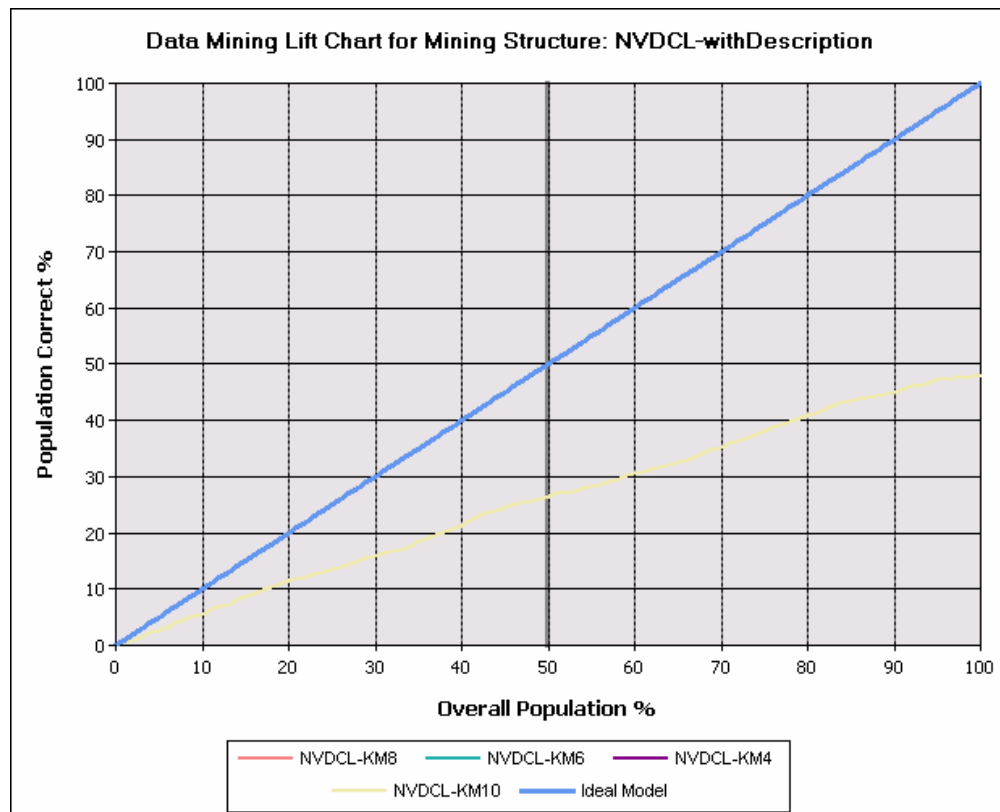


Mining Legend ✕

Population percentage: 50.00%

Series, Model	Score	Population correct	Predict probability
NVDCL-EM10	0.80	42.72%	50.65%
NVDCL-EM8	0.82	44.05%	56.31%
NVDCL-EM6	0.79	43.98%	55.06%
Ideal Model		50.00%	

- With NVDCL-EM4 consistently underperforming the others, it was replaced with a KMeans algorithm for further comparison

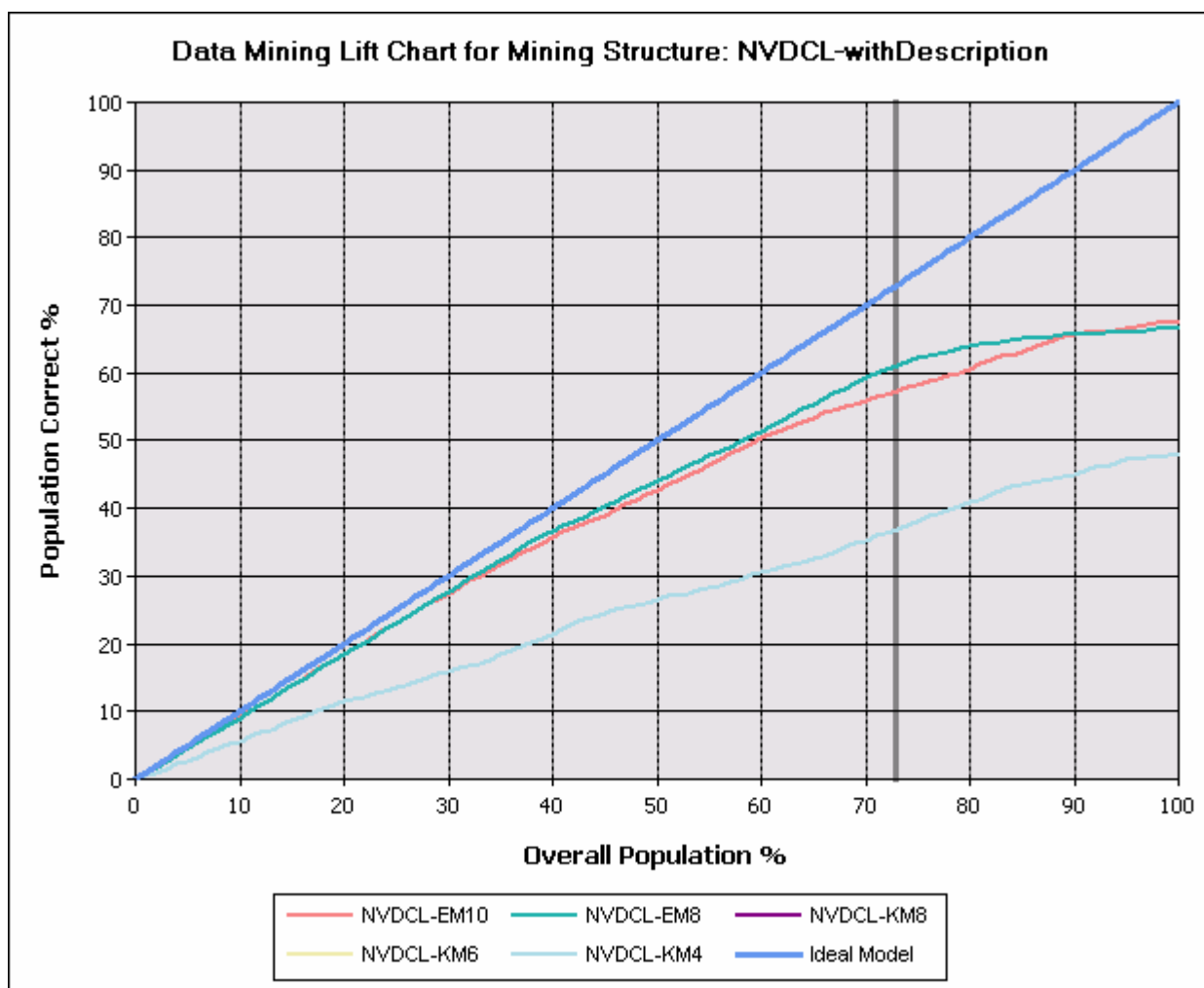


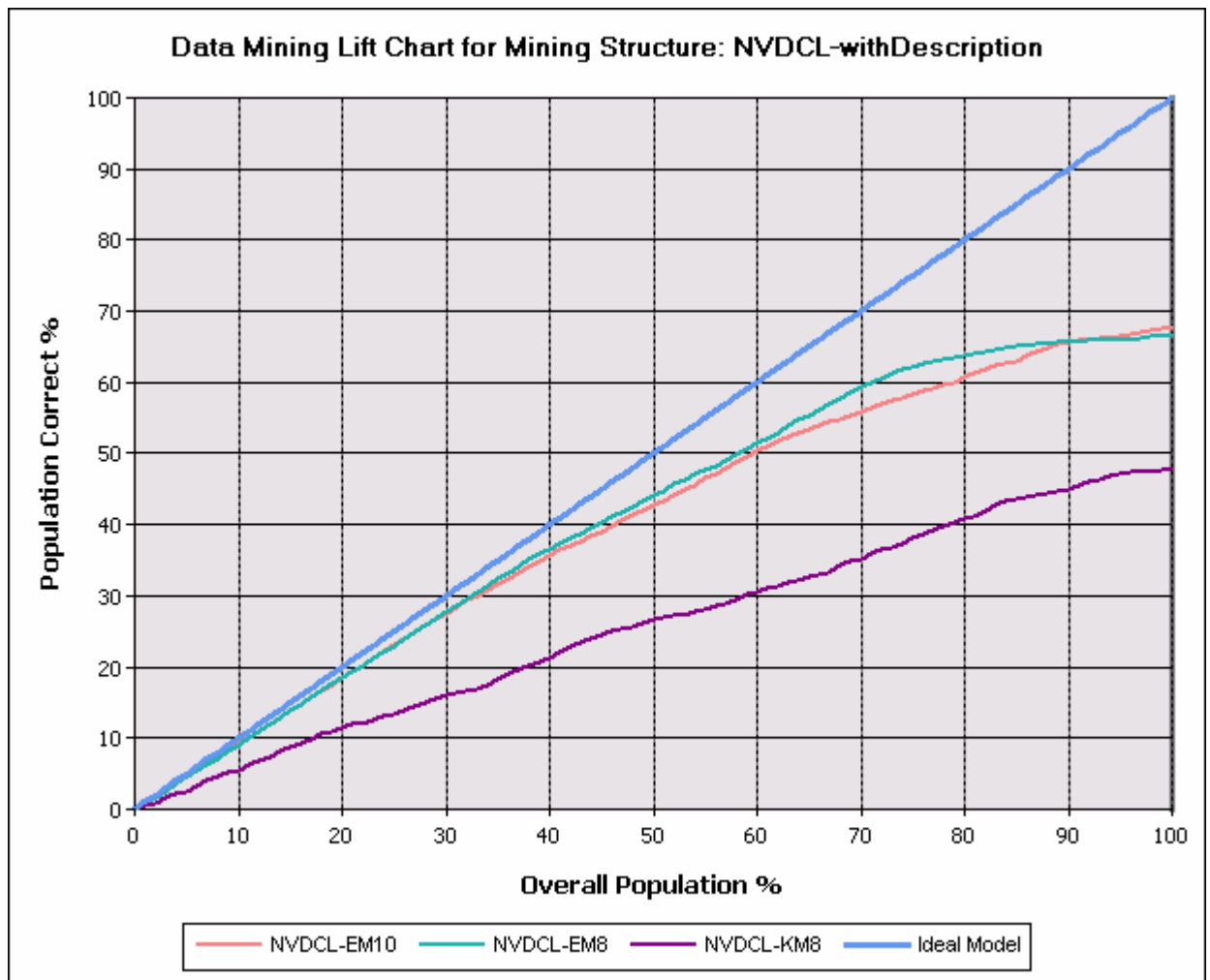
Mining Legend ✕

Population percentage: 100.00%

Series, Model	Score	Population correct	Predict probability
NVDCL-KM8	0.51	47.74%	43.24%
NVDCL-KM6	0.51	47.74%	42.82%
NVDCL-KM4	0.51	47.74%	41.36%
NVDCL-KM10	0.51	47.74%	44.27%
Ideal Model		100.00%	

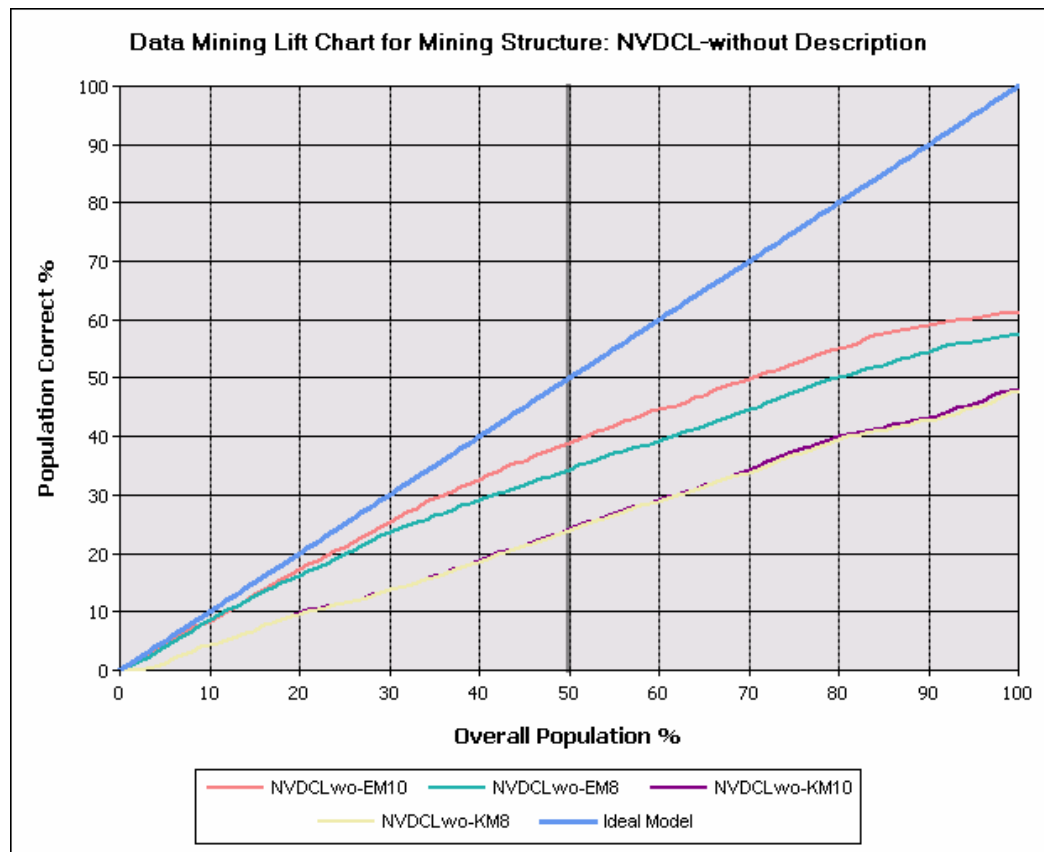
The K-Means algorithm regardless of number of nodes used had the same score and population predicted as correct were the same, so it did not matter which one we used.





The KMeans models consistently underperformed the EM models –
Choosing the EM-8 models.

- Without Description



Mining Legend x

Population percentage: 50.00%

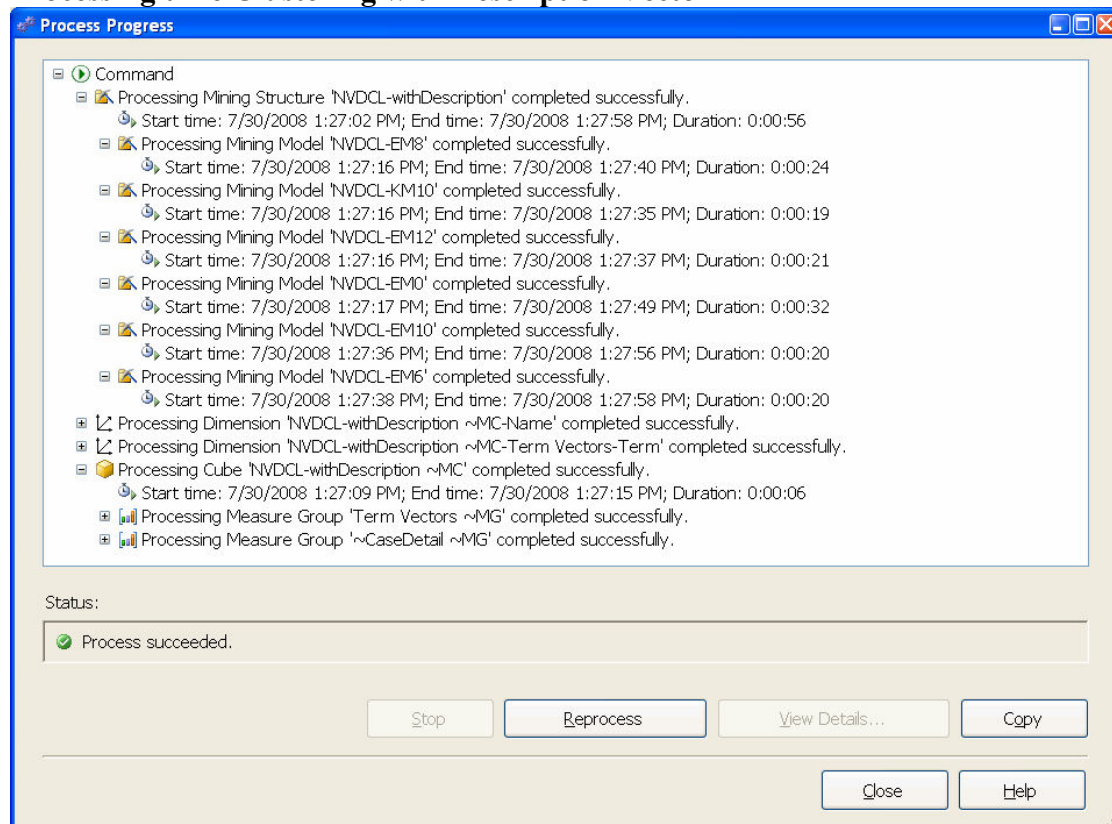
Series, Model	Score	Population correct	Predict probability
NVDCLwo-EM10	0.73	39.01%	51.78%
NVDCLwo-EM8	0.66	34.32%	42.17%
NVDCLwo-KM10	0.48	24.08%	43.89%
NVDCLwo-KM8	0.48	23.82%	43.13%
Ideal Model		50.00%	

<

>

Mining Legend ↺ Deployment Progress

Processing time Clustering with Description Vector



We looked at the processing time for clustering given the description vector, given all of the algorithms, it took 56 seconds.

Appendix B-3 – Query for Identifying Clusters in Training Data

The following query was run on the analysis server to predict and assign clusters to each entry.

```
SELECT
    t.[name],
    t.[discovered],
    t.[published],
    t.[cvss_score],
    t.[loss_type],
    t.[adminSP],
    t.[userSP],
    t.[otherSP],
    t.[vuln_type],
    t.[input_bound],
    t.[input_buffer],
    t.[exploit_range],
    t.[ref],
    t.[source],
    t.[sig],
    t.[adv],
    t.[patch],
    t.[prodName],
    t.[versionNum],
    t.[preVersion],
    t.[edition],
    (Cluster()) as
[ClusterNode],
    (ClusterProbability()) as
[ClusterProbability]
From
    [NVDCL-EM-10]
PREDICTION JOIN
    SHAPE {
        OPENQUERY ([SSRAM],
            'SELECT
                [name],
                [discovered],
                [published],
                [cvss_score],
                [loss_type],
                [adminSP],
                [userSP],
                [otherSP],
                [vuln_type],
                [input_bound],
                [input_buffer],
                [exploit_range],
                [ref],
                [source],
                [sig],
                [adv],
                [patch],
                [prodName],
                [versionNum],
                [preVersion],
                [edition]
            FROM
                [dbo].[TrainingData_96_01]
            ORDER BY
                [name]') }
        APPEND
        ({OPENQUERY ([SSRAM],
            'SELECT
                [Term],
                [Frequency],
                [entry_name]
            FROM
                [dbo].[TermVectors]
            ORDER BY
                [entry_name]') }
        RELATE
            [name] TO [entry_name])
        AS
            [TermVectors] AS t
    ON
        [NVDCL-EM-10].[Discovered] =
t.[discovered] AND
        [NVDCL-EM-10].[Published] =
t.[published] AND
        [NVDCL-EM-10].[Cvss Score] =
t.[cvss_score] AND
        [NVDCL-EM-10].[Loss Type] =
t.[loss_type] AND
        [NVDCL-EM-10].[Admin SP] =
t.[adminSP] AND
        [NVDCL-EM-10].[User SP] =
t.[userSP] AND
        [NVDCL-EM-10].[Other SP] =
t.[otherSP] AND
        [NVDCL-EM-10].[Vuln Type] =
t.[vuln_type] AND
        [NVDCL-EM-10].[Input Bound] =
t.[input_bound] AND
        [NVDCL-EM-10].[Input Buffer] =
t.[input_buffer] AND
```

```
[NVDCL-EM-10].[Version Num] =
t.[versionNum] AND
[NVDCL-EM-10].[Pre Version] =
t.[preVersion] AND
[NVDCL-EM-10].[Edition] =
t.[edition] AND
[NVDCL-EM-10].[Term
Vectors].[Term] =
t.[TermVectors].[Term] AND
[NVDCL-EM-10].[Term
Vectors].[Frequency] =
t.[TermVectors].[Frequency]
```

CVE-ID	\$CLUSTER	cvss_score	loss_type	published	vuln_type	exploit_range	source	prodName	versionNum
CVE-1999-1549	Cluster 1	3.3	conf	11/16/1999 12:...	design	remote	TheAimsGroup	Lynx	2.8
CVE-1999-1549	Cluster 1	3.3	conf	11/16/1999 12:...	design	local	Security Focus	Lynx	2.7
CVE-1999-1549	Cluster 1	3.3	conf	11/16/1999 12:...	design	local	Security Focus	Lynx	2.8
CVE-1999-1549	Cluster 1	3.3	conf	11/16/1999 12:...	design	remote	Security Focus	Lynx	2.7
CVE-1999-1549	Cluster 1	3.3	conf	11/16/1999 12:...	design	remote	Security Focus	Lynx	2.8
CVE-1999-1550	Cluster 1	3.3	conf	11/8/1999 12:0...	design	local	TheAimsGroup	BigIP	2.0
CVE-1999-1550	Cluster 1	3.3	conf	11/8/1999 12:0...	design	remote	TheAimsGroup	BigIP	2.0
CVE-1999-1550	Cluster 4	3.3	conf	11/8/1999 12:0...	design	local	TheAimsGroup	BigIP	2.0
CVE-1999-1550	Cluster 1	3.3	conf	11/8/1999 12:0...	design	remote	TheAimsGroup	BigIP	2.0
CVE-1999-1550	Cluster 4	3.3	conf	11/8/1999 12:0...	design	local	TheAimsGroup	BigIP	2.0
CVE-1999-1550	Cluster 1	3.3	conf	11/8/1999 12:0...	design	remote	TheAimsGroup	BigIP	2.0
CVE-1999-1550	Cluster 1	3.3	conf	11/8/1999 12:0...	design	local	Security Focus	BigIP	2.0
CVE-1999-1550	Cluster 1	3.3	conf	11/8/1999 12:0...	design	remote	Security Focus	BigIP	2.0
CVE-1999-1551	Cluster 2	3.3	avail	3/2/1999 12:00...	input	remote	BUGTRAQ	IMail	5.0
CVE-1999-1551	Cluster 3	3.3	avail	3/2/1999 12:00...	input	remote	BUGTRAQ	IMail	6.0
CVE-1999-1551	Cluster 3	3.3	avail	3/2/1999 12:00...	input	remote	BID	IMail	5.0
CVE-1999-1551	Cluster 3	3.3	avail	3/2/1999 12:00...	input	remote	BID	IMail	6.0
CVE-1999-1551	Cluster 3	3.3	avail	3/2/1999 12:00...	input	remote	XF	IMail	5.0
CVE-1999-1551	Cluster 3	3.3	avail	3/2/1999 12:00...	input	remote	XF	IMail	6.0
CVE-1999-1553	Cluster 2	10	avail	5/1/1999 12:00...	input	remote	BUGTRAQ	XCmail	0.99.6
CVE-1999-1553	Cluster 2	10	sec_prot	5/1/1999 12:00...	input	remote	BUGTRAQ	XCmail	0.99.6
CVE-1999-1553	Cluster 2	10	avail	5/1/1999 12:00...	input	remote	BID	XCmail	0.99.6
CVE-1999-1553	Cluster 2	10	sec_prot	5/1/1999 12:00...	input	remote	BID	XCmail	0.99.6
CVE-1999-1553	Cluster 2	10	sec_prot	5/1/1999 12:00...	input	remote	XF	XCmail	0.99.6
CVE-1999-1553	Cluster 2	10	sec_prot	5/1/1999 12:00...	input	remote	XF	XCmail	0.99.6
CVE-1999-1555	Cluster 2	7	sec_prot	6/14/1999 12:0...	design	local	Security Focus	Internet Mail Abi...	4.0

Appendix B-4 – Classified Data Matrix

Classification models are evaluated based on the number of test records that they correctly and incorrectly predict. Columns correspond to the actual values and rows depict the predicted values. The metric to measure the performance of each model – Accuracy, defined as follows

$$Accuracy = \frac{\text{Number of correct Predictions}}{\text{Total Number of predictions}}$$

The rest of this appendix shows the confusion matrix used to determine the accuracy of the classification models.

Classifier without Description

Counts for DT ClusterClassifier on [Cluster Node]	Classification Confusion Matrix								
	Predicted	Cluster 1 (Actual)	Cluster 6 (Actual)	Cluster 4 (Actual)	Cluster 3 (Actual)	Cluster 8 (Actual)	Cluster 5 (Actual)	Cluster 7 (Actual)	Cluster 2 (Actual)
	Cluster 1	2708	5	0	6	42	4	0	102
	Cluster 6	767	1243	0	256	2	162	0	54
	Cluster 4	31	13	167	2	41	23	71	2
	Cluster 3	0	48	0	117	0	5	7	79
	Cluster 8	0	0	0	0	0	0	0	0
	Cluster 5	32	92	9	18	0	1099	13	91
	Cluster 7	47	6	394	5	289	2	762	25
	Cluster 2	22	24	5	592	6	17	6	657
Accuracy .66									
Counts for NB ClusterClassifier on [Cluster Node]									
	Predicted	Cluster 1 (Actual)	Cluster 6 (Actual)	Cluster 4 (Actual)	Cluster 3 (Actual)	Cluster 8 (Actual)	Cluster 5 (Actual)	Cluster 7 (Actual)	Cluster 2 (Actual)
	Cluster 1	3158	496	60	145	74	164	41	157
	Cluster 6	307	680	1	89	0	63	2	29
	Cluster 4	35	14	257	5	75	32	197	6
	Cluster 3	19	129	10	513	0	7	4	103
	Cluster 8	2	0	9	0	17	0	15	6
	Cluster 5	47	89	6	15	2	1017	7	59

	Cluster 7	28	2	226	8	202	2	578	22
	Cluster 2	11	21	6	221	10	27	15	628
Accuracy 0.67									
Counts for NN ClusterClassifier on [Cluster Node]									
	Predicted	Cluster 1 (Actual)	Cluster 6 (Actual)	Cluster 4 (Actual)	Cluster 3 (Actual)	Cluster 8 (Actual)	Cluster 5 (Actual)	Cluster 7 (Actual)	Cluster 2 (Actual)
	Cluster 1	3052	260	46	77	70	149	31	137
	Cluster 6	421	889	0	154	0	73	1	28
	Cluster 4	34	3	207	3	68	32	138	7
	Cluster 3	11	146	6	566	2	5	3	174
	Cluster 8	0	1	6	1	2	1	4	1
	Cluster 5	50	89	16	15	3	1027	11	70
	Cluster 7	27	10	285	10	226	2	662	29
	Cluster 2	12	33	9	170	9	23	9	564
Accuracy 0.86									

Classifier with Description

Classification Confusion Matrix									
Counts for DTCluster Classifier with Desc on [Cluster Node]									
	Predicted	Cluster 1 (Actual)	Cluster 6 (Actual)	Cluster 4 (Actual)	Cluster 3 (Actual)	Cluster 8 (Actual)	Cluster 5 (Actual)	Cluster 7 (Actual)	Cluster 2 (Actual)
	Cluster 1	2708	5	0	6	42	4	0	102
	Cluster 6	774	1251	0	256	2	212	0	54
	Cluster 4	0	0	0	0	0	0	0	0
	Cluster 3	0	0	0	0	0	0	0	0
	Cluster 8	0	0	0	0	0	0	0	0
	Cluster 5	32	72	0	18	0	882	3	95
	Cluster 7	79	31	570	7	330	209	843	40
	Cluster 2	14	72	5	709	6	5	13	719
Accuracy	0.63								
Counts for NB ClusterClassifier with Desc on [Cluster Node]									
	Predicted	Cluster 1 (Actual)	Cluster 6 (Actual)	Cluster 4 (Actual)	Cluster 3 (Actual)	Cluster 8 (Actual)	Cluster 5 (Actual)	Cluster 7 (Actual)	Cluster 2 (Actual)
	Cluster 1	3158	409	0	105	44	163	3	129
	Cluster 6	339	809	0	153	0	64	0	33
	Cluster 4	31	13	185	2	41	23	71	2
	Cluster 3	3	83	2	325	0	7	0	83
	Cluster 8	0	0	0	0	0	0	0	0
	Cluster 5	12	84	9	16	0	1033	10	79
	Cluster 7	47	6	376	9	289	2	769	29
	Cluster 2	17	27	3	386	6	20	6	655
Accuracy	0.68								

Counts for NN ClusterClassifier with Desc on [Cluster Node]									
	Predicted	Cluster 1 (Actual)	Cluster 6 (Actual)	Cluster 4 (Actual)	Cluster 3 (Actual)	Cluster 8 (Actual)	Cluster 5 (Actual)	Cluster 7 (Actual)	Cluster 2 (Actual)
	Cluster 1	3482	880	100	196	83	205	25	155
	Cluster 6	33	347	0	41	0	22	0	2
	Cluster 4	0	0	4	0	0	0	0	0
	Cluster 3	2	63	8	287	0	5	15	43
	Cluster 8	0	0	0	0	0	0	0	0
	Cluster 5	25	85	12	16	0	1043	13	85
	Cluster 7	48	11	448	7	291	13	793	22
	Cluster 2	17	45	3	449	6	24	13	703
Accuracy	0.65								

Appendix B-5 – Validation of Clustering Algorithms

Validation Statistics of all models for Training Data 1996-2001. In order that the cohesion of each model could be determined we looked at the average score for each cluster and the standard deviation. This allowed us to see the variability around the mean. We also looked at the range of scores within the clusters by getting the minimum (Min) and maximum (Max) score. The wtd scores were based on the growth rate of the scores as explained in Chapter 3.

EM8Clusters - All										
	Count	Avg Probability	Std Deviation	Avg Score	Max Score	Min Score	Wtd StdDev	Wtd Avg Score	Wtd Max	Wtd Min
Cluster 1	22717	0.81	2.08	7.44	10	1.6	2.25	6.09	10	0.91
Cluster 2	7816	0.54	2.2	5.43	10	1.9	1.43	2.89	10	0.95
Cluster 3	7296	0.51	2.19	5.16	10	1.9	1.64	2.66	10	1.06
Cluster 4	3856	0.51	2.39	5.01	10	2.3	1.52	2.46	10	0.83
Cluster 5	9306	0.62	2.33	4.75	10	2.3	1.84	3.02	10	1.02
Cluster 6	514	0.9	0	7	7	7	0.61	6.29	7	5.59
EM8Clusters										
Probability >.8	Count	Avg Probability	Std Deviation	Avg Score	Max Score	Min Score	Wtd StdDev	Wtd Avg Score	Wtd Max	Wtd Min
Cluster 1	14101	0.94	1.67	7.69	10	1.6	1.64	7.23	10	1.6
Cluster 2	687	0.97	2.26	5.21	10	2.3	2.19	5.03	10	1.86
Cluster 3	1011	0.96	2.4	4.58	10	1.9	2.29	4.39	10	1.59
Cluster 4	631	0.99	2.15	4.51	10	2.3	2.11	4.45	10	1.9
Cluster 5	2143	0.86	2.31	6.32	10	2.3	1.88	5.35	10	1.85
Cluster 6	292	0.97	0	7	7	7	0.01	6.82	7	6.82
EM10Clusters										
All	Count	Avg Probability	Std Deviation	Avg Score	Max Score	Min Score	Wtd StdDev	Wtd Avg Score	Wtd Max	Wtd Min
Cluster 1	9468	0.65	2.02	4.38	10	1.9	1.34	2.78	10	0.72
Cluster 2	11350	0.82	1.24	6.75	10	1.9	1.32	5.51	10	0.9
Cluster 3	11688	0.86	1.59	8.89	10	1.6	2.23	7.72	10	1.03
Cluster 4	6940	0.55	1.91	6.04	10	1.9	1.34	3.23	10	1.01
Cluster 5	4869	0.51	1.91	4.6	10	2.3	1.35	2.38	10	1.2
Cluster 6	6649	0.74	1.73	3.9	10	2.3	1.25	2.82	10	1.12
Cluster 7	541	0.87	0	7	7	7	0.69	6.11	7	5.34

EM10Clusters										
>0.80	Count	Avg Probability	Std Deviation	Avg Score	Max Score	Min Score	Wtd StdDev	Wtd Avg Score	Wtd Max	Wtd Min
Cluster 1	1700	0.9	1.61	3.8	10	1.9	1.71	3.48	10	1.86
Cluster 2	8351	0.9	0.81	6.74	10	1.9	0.78	6.09	10	1.61
Cluster 3	8761	0.94	1.34	9.29	10	1.6	1.46	8.72	10	1.59
Cluster 4	860	0.94	2.27	4.41	10	1.9	2.17	4.13	10	1.57
Cluster 5	429	0.98	1.91	4.45	10	2.3	1.87	4.35	10	2.25
Cluster 6	2990	0.86	1.54	3.28	10	2.3	1.48	2.85	10	1.85
Cluster 7	300	0.96	0	7	7	7	0.02	6.73	7	6.73

EM12Clusters										
All	Count	Avg Probability	Std Deviation	Avg Score	Max Score	Min Score	Wtd StdDev	Wtd Avg Score	Wtd Max	Wtd Min
Cluster 1	15406	0.85	1.79	8.08	10	1.6	2.3	6.86	10	1.49
Cluster 2	6655	0.73	2.07	4.18	10	2.3	1.33	2.87	10	0.96
Cluster 3	2072	0.5	1.93	4.2	10	2.3	1.95	2.26	10	0.61
Cluster 4	2324	0.56	2.79	7.23	10	1.9	1.68	3.51	10	1.16
Cluster 5	4484	0.5	1.06	3.57	10	1.9	1.17	1.81	10	0.82
Cluster 6	5843	0.53	1.37	6.71	10	1.9	0.91	3.46	10	1.19
Cluster 7	5191	0.53	2.36	5.11	10	2.3	1.58	2.74	10	0.89
Cluster 8	9016	0.41	1.97	5.87	10	2.7	1.17	2.48	9.98	0.84
Cluster 9	514	0.84	0	7	7	7	0.87	5.88	7	4.89

EM12Clusters										
>0.8	Count	Avg Probability	Std Deviation	Avg Score	Max Score	Min Score	Wtd StdDev	Wtd Avg Score	Wtd Max	Wtd Min
Cluster 1	11589	0.96	1.73	8.1	10	1.6	1.69	7.81	10	1.6
Cluster 2	3107	0.9	1.56	3.31	10	2.3	1.56	3.03	10	1.85
Cluster 3	471	0.97	2.22	4.85	10	2.3	2.14	4.71	10	2.14
Cluster 4	673	0.95	2.44	4.9	10	1.9	2.4	4.7	10	1.55
Cluster 5	518	0.99	1.93	4.15	10	1.9	1.9	4.11	10	1.9
Cluster 6	435	0.9	1.25	5.84	10	1.9	1.29	5.26	10	1.89
Cluster 7	266	0.98	2.53	4.81	10	2.3	2.54	4.74	10	2.23
Cluster 8	71	0.95	2.16	5.62	10	2.7	2.06	5.35	9.98	2.69
Cluster 9	292	0.95	0	7	7	7	0.02	6.64	7	6.64

KM10Clusters										
All	Count	Avg Probability	Std Deviation	Avg Score	Max Score	Min Score	Wtd StdDev	Wtd Avg Score	Wtd Max	Wtd Min
Cluster 1	12020	1	1.19	7.01	10	2.3	1.19	7.01	10	2.3
Cluster 10	2126	1	1.75	4.14	10	1.9	1.75	4.14	10	1.9
Cluster 2	12774	1	1.64	8.68	10	1.9	1.64	8.68	10	1.9
Cluster 3	6422	1	0.55	3.38	10	1.9	0.55	3.38	10	1.9
Cluster 4	2642	1	0.77	3.41	10	1.9	0.77	3.41	10	1.9
Cluster 5	4423	1	0.73	6.99	10	2.3	0.73	6.99	10	2.3
Cluster 6	3136	1	0.62	2.41	10	1.9	0.62	2.41	10	1.9
Cluster 7	2257	1	1.02	5.25	10	1.6	1.02	5.25	10	1.6
Cluster 8	3522	1	2	4.64	10	1.9	2	4.64	10	1.9
Cluster 9	2183	1	2.04	6.78	10	2.3	2.04	6.78	10	2.3

Training Data by Clusters (1996 - 2001)					With 95% Confidence Interval	
NumInCluster	CVSS_Score	StdDev	Margin of Error	Cluster	Lower	Upper
1395	7.9726	1.7142	0.09	Cluster 1	7.8826	8.0625
867	6.7191	1.4718	0.098	Cluster 6	6.6211	6.817
338	6.1498	2.6781	0.2855	Cluster 8	5.8643	6.4353
651	5.5779	2.266	0.1741	Cluster 2	5.4038	5.752
573	4.8681	2.1551	0.1765	Cluster 3	4.6916	5.0445
431	4.6508	2.4416	0.2305	Cluster 4	4.4203	4.8813
536	4.384	2.1133	0.1789	Cluster 5	4.2051	4.5629
549	3.3	0	0	Cluster 7	3.3	3.3

Appendix B-6 – Processing Time for Classifiers

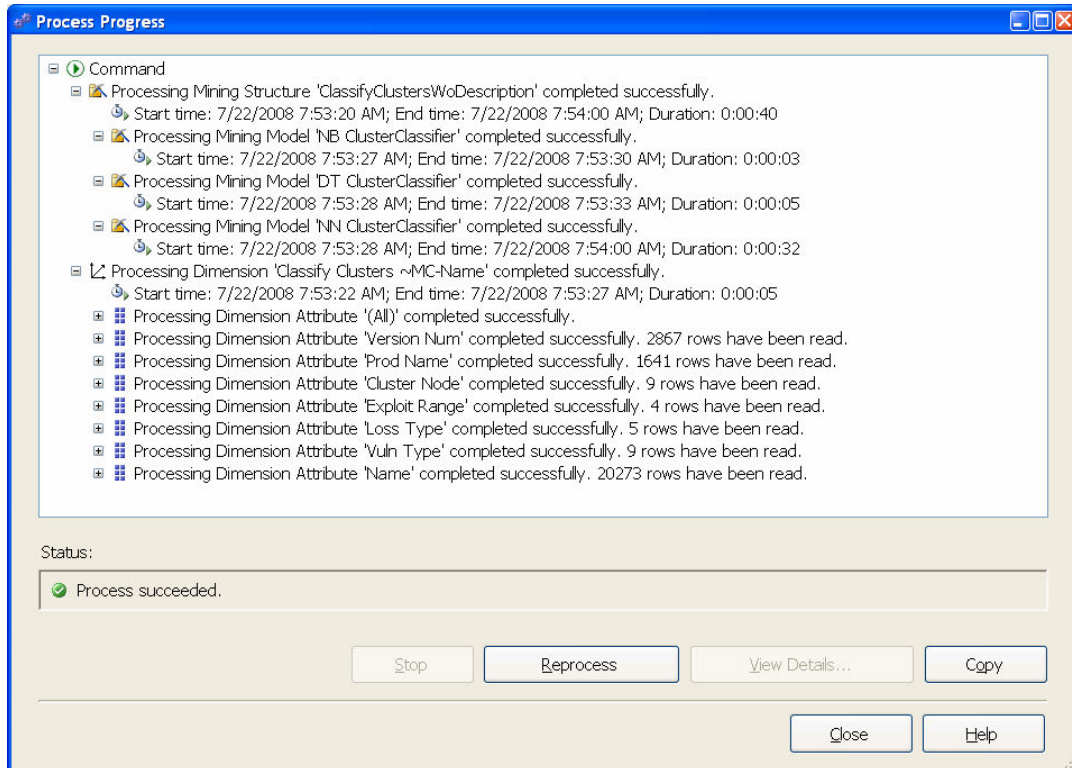


Figure 37 - Processing Time for Classification Algorithm

```
<Batch xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
```

```
<Parallel>
```

```
<Process xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
<Object>
```

```
<DatabaseID>SSRAM22608</DatabaseID>
```

```
<MiningStructureID>Classify Clusters</MiningStructureID>
```

```
</Object>
```

```
<Type>ProcessFull</Type>
```

```
<WriteBackTableCreation>UseExisting</WriteBackTableCreation>
```

```
</Process>
```

```
</Parallel>
```

```
</Batch>
```

Processing Mining Structure 'ClassifyClustersWoDescription' completed successfully.

Start time: 7/22/2008 7:53:20 AM; End time: 7/22/2008 7:54:00 AM; Duration: 0:00:40

Processing Mining Model 'NB ClusterClassifier' completed successfully.

Start time: 7/22/2008 7:53:27 AM; End time: 7/22/2008 7:53:30 AM; Duration: 0:00:03

Processing Mining Model 'DT ClusterClassifier' completed successfully.

Start time: 7/22/2008 7:53:28 AM; End time: 7/22/2008 7:53:33 AM; Duration: 0:00:05

Processing Mining Model 'NN ClusterClassifier' completed successfully.

Start time: 7/22/2008 7:53:28 AM; End time: 7/22/2008 7:54:00 AM; Duration: 0:00:32

Appendix B-7 – Calculating CVSS ‘base’ Score

An example of how the CVSS base score is derived – given the entry in Figure 22 -

CVSS base Score and Vector

BASE METRIC	EVALUATION	SCORE
Access Vector	[Local]	(0.395)
Access Complexity	[Low]	(0.71)
Authentication	[None]	(0.704)
Confidentiality Impact	[Complete]	(0.66)
Integrity Impact	[Complete]	(0.66)
Availability Impact	[Complete]	(0.66)
BASE FORMULA		BASE SCORE

$\text{Impact} = 10.41 * (1 - (0.34 * 0.34 * 0.4)) = 9.83$

$\text{Exploitability} = 20 * 0.395 * 0.71 * 0.704 = 3.95$

$f(\text{Impact}) = 1.176$

$\text{Base Score} = ((0.6 * 9.83) + (0.4 * 3.95) - 1.5) * 1.176$

$= (7.09)$

Note that the score values above are derived from the algorithm shown below from [First 2005]

```
BaseScore = round_to_1_decimal(((0.6*Impact)+(0.4*Exploitability)-1.5)*f(Impact))
```

```
Impact = 10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))
```

```
Exploitability = 20* AccessVector*AccessComplexity*Authentication
```

```
f(impact)= 0 if Impact=0, 1.176 otherwise
```

```
AccessVector      = case AccessVector of
                      requires local access: 0.395
                      adjacent network accessible: 0.646
                      network accessible: 1.0
```

```
AccessComplexity = case AccessComplexity of
                      high: 0.35
                      medium: 0.61
                      low: 0.71
```

Authentication	= case Authentication of
0.45	requires multiple instances of authentication:
0.56	requires single instance of authentication:
	requires no authentication: 0.704
ConfImpact	= case ConfidentialityImpact of
	none: 0.0
	partial: 0.275
	complete: 0.660
IntegImpact	= case IntegrityImpact of
	none: 0.0
	partial: 0.275
	complete: 0.660
AvailImpact	= case AvailabilityImpact of
	none: 0.0
	partial: 0.275
	complete: 0.660

Appendix C-1 – Stored Procedure for Calculating Impact Score Attributes

This appendix shows the code for calculating impact score attributes

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go

-- =====
-- Author:      Idongesit Mkpong-Ruffin
-- Create date: 4/14/2008
-- Description: Calculate impact factor values
-- =====
ALTER PROCEDURE [dbo].[spCalcImpactFactors]
    -- Add the parameters for the stored procedure here
    (
        @endDate datetime = '01/01/2001',
        @timePeriods int = 12,
        @clusterNode nvarchar(155)='Cluster 1',
        @weightedAvgScore float OUTPUT,
        @avgScoreGrowth float OUTPUT,
        @avgFreqGrowth float OUTPUT,
        @SumResult int OUTPUT) -- to see if i can get the information
    returned
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Declare the return variable here
    --DECLARE @SumResult int

    -- Temporary table to store interim data for computation

    DECLARE @tmpClusterInfo table(timeGap int, avgScore float,
    reportedEntries float)

    INSERT INTO @tmpClusterInfo(timeGap, avgScore,
    reportedEntries)
        select datediff(month, published, @endDate)as timeGap,
    avg(cvss_score) as avgScore, count(distinct name) as reportedEntries

    from trainingdataawclusters96_01
    where
        (datediff(month, published, @endDate) > 0)and
        (datediff(month, published, @endDate) <=
    @timePeriods)and
```

```

        clusternode = @clusterNode
        group by datediff(month, published, @endDate)
        order by timegap

-- Total Entries
--      select sum(reportedEntries) from @tmpClusterInfo as
TotalEntries --debug
        set @SumResult = (select sum(reportedEntries) from
@tmpClusterInfo)

        --- get the total entries for coefficientSummation
        set @weightedAvgScore = (select
round(sum(((reportedEntries)/@sumResult)*avgScore),2)
        from @tmpClusterInfo)

-- Score Growth
        -- get the avgGrowth rate of Score for this period within
cluster

        set @avgScoreGrowth = (select
round(avg(((nextRow.avgScore/curRow.avgScore)-1)),2) as avgScoreGrowth
        from @tmpClusterInfo curRow
        left join @tmpClusterInfo nextRow
        on curRow.timegap = nextRow.timeGap-1)

-- Frequency Of Occurrence Growth
        -- get the avgGrowth rate of Score for this period within cluster

        set @avgFreqGrowth = (select
round(avg(((nextRow.reportedEntries/curRow.reportedEntries)-1)),2)
        from @tmpClusterInfo curRow
        left join @tmpClusterInfo nextRow
        on curRow.timegap = nextRow.timeGap-1)

        -- Add the T-SQL statements to compute the return value here
return

END

```

Appendix C-2 – Stored Procedure for Determining Prioritized Listing

Once each entry has been de-normalized so that component elements of the entry can be analyzed and assigned scores, they are aggregated to determine overall impact score for the entry. This appendix shows the code used to effect this in SSRAM.

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go

-- =====
-- Author:          Idongesit Mkpong-Ruffin
-- Create date:
-- Description:
-- =====
ALTER PROCEDURE [dbo].[spPrioritizedList]
    -- Add the parameters for the stored procedure here

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    select distinct entryID, round(avg(predImpact),2) as PredImpact,
        round(avg(predFreq),0) as PredFreq, round(avg(lossExpect),2) as
LossExpect

        from predictions
        group by entryID
        order by LossExpect desc
END
```

Appendix C-3 – Stored Procedure for Persisting Predictions

To be able to prioritize predictions requires that the results of the prediction are stored.

This appendix shows the code use for persisting predictions.

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go

-- =====
-- Author:      Idongesit Mkpong-Ruffin
-- Create date: 5/19/08
-- Description:  Creates persistent storage of predicted information
-- =====
ALTER PROCEDURE [dbo].[spPredictionStore]
    -- Add the parameters for the stored procedure here
    @entryID varchar(30),
    @vulnType varchar(15),
    @lossType varchar(15),
    @exploitRange varchar(15),
    @clusterNode varchar(15),
    @predImpact float,
    @predFreq float,
    @lossExp float

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    insert into
    Predictions(entryID,vulnType,lossType,exploitRange,clusterNode,predImpact,predFreq, lossExpect)

    values(@entryID,@vulnType,@lossType,@exploitRange,@clusterNode,@predImpact,@predFreq,@lossExp);

END
```

Appendix D-1 – Singleton Request Algorithm

To validate that SSRAM could, based on loss type, vulnerability type and exploit range, classify and entry and by extension accurately predict CVSS score, we generated a single request of known clusters based on the code shown in this appendix.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.AnalysisServices.AdomdClient;

namespace SSRAMWinApp1
{
    public partial class frmSingle : Form
    {
        public frmSingle()
        {
            InitializeComponent();

            private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
            {

            }

            private void label2_Click(object sender, EventArgs e)
            {

            }

            private void label1_Click(object sender, EventArgs e)
            {

            }

            private void Form1_Load(object sender, EventArgs e)
            {

            }

            private void label1_Click_1(object sender, EventArgs e)
            {

            }
        }
    }
}
```

```

private void btnCalc_Click(object sender, EventArgs e)
{
    string strASConnString =
SSRAMWinApp1.Properties.Settings.Default.AS_SSRAMConnectionString;
    string strDBConnString =
Convert.ToString(SSRAMWinApp1.Properties.Settings.Default.DB_SSRAMConne
ctionString);
    string strClusterLbl = "";
    double dblPredictImpact = 0;
    double dblPredictFreq = 0;
    DateTime dtPredictDate;
    int intPeriods = 0;

    try
    {
        IPredictElementCluster objRequester = new
PredictElementCluster();

        this.txtResult.Text = "";
        //
        string strRequestCluster =
objRequester.Command(strASConnString, this.tbVT.Text.Trim(),
this.tbLT.Text.Trim(), this.tbER.Text.Trim());
        string strRequestCluster =
objRequester.Command(strASConnString,
this.cbVulnType.SelectedItem.ToString().Trim(),
this.cbLossType.SelectedItem.ToString().Trim(),
this.cbExploitRange.SelectedItem.ToString().Trim());

        strClusterLbl = objRequester.getClusterNode;

        ICalculateImpactFactor objRequestImpact = new
CalcImpactFactor();

        this.txtImpactInfo.Text = ""; // to put impact factor
and loss expectation
        dtPredictDate =
Convert.ToDateTime(this.dateTimePicker1.Text.Trim());
        // result = Int32.TryParse("3", NumberStyles.Integer,
null, out int32Val);
        bool blconvertPeriod =
int.TryParse(this.txtMonths.Text, out intPeriods);
        //bool blcp = int.TryParse(this.txtImpactInfo.Text, out
intPeriods);
        string strImpactInfo =
objRequestImpact.Command(strDBConnString, dtPredictDate, intPeriods, strCl
usterLbl);
        int intImpactLength = strImpactInfo.Length;

        //get predicted Impact
        dblPredictImpact = objRequestImpact.getPredictedImpact;
        dblPredictFreq =
objRequestImpact.getPredictedOccurrences;

```

```

        //display results
        this.txtResult.Text = strRequestCluster +
strImpactInfo;
        this.txtImpactInfo.Text = "Predicted Impact: " +
dblPredictImpact.ToString().Trim() + "\r\n" +
        "Predicted Occurrence: " +
dblPredictFreq.ToString().Trim();
    }

    catch (Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.ToString());

        //      try
        //      {
        //          IPredictCustomerMDXRequester objRequester =
new PredictCustomerMDXRequester();

        //          this.m_tbResult.Text = "";
        //          this.m_tbResult.Text =
objRequester.Command(strASConnString, this.m_tbCity.Text.Trim(),
this.m_tbContactTitle.Text.Trim());
        //      }
        //      catch (Exception ex)
        //      {
        //
        System.Windows.Forms.MessageBox.Show(ex.ToString());
        //      }
    }
} // btnCalcClick

private void label1_Click_2(object sender, EventArgs e)
{
}

private void textBox1_TextChanged(object sender, EventArgs e)
{
}

private void label2_Click_1(object sender, EventArgs e)
{
}

private void checkedListBox1_SelectedIndexChanged(object
sender, EventArgs e)
{
}

```

```

        private void label1_Click_3(object sender, EventArgs e)
        {

        }

        private void comboBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {

        }

        private void comboBox1_SelectedIndexChanged_1(object sender,
EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            Close();
        }
    }

```


Appendix D-2 – Data Table Request (Code for Load Data)

This appendix shows the code for loading a table of de-normalized entries for classification and prediction of impact scores.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.AnalysisServices.AdomdClient;

namespace SSRAMWinApp1
{
    public partial class frmTableRequest : Form
    {
        // Adding the ManufactureEntryDataTable
        public static void ManufactureEntryDataTable(DataSet ds)
        {
            DataTable entries = new DataTable("Entries");
            entries.Columns.Add(new DataColumn("EntryID",
typeof(string)));
            entries.Columns.Add(new DataColumn("VulnType",
typeof(string)));
            entries.Columns.Add(new DataColumn("LossType",
typeof(string)));
            entries.Columns.Add(new DataColumn("ExploitRange",
typeof(string)));

            ds.Tables.Add(entries);

        } // ManufactureEntryDataTable(DataSet ds)

        // Adding the ManufacturePredictDataTable
        public static void ManufacturePredictDataTable(DataSet ds)
        {
            DataTable predictions = new DataTable("Predictions");
            predictions.Columns.Add(new DataColumn("EntryID",
typeof(string)));
            predictions.Columns.Add(new DataColumn("VulnType",
typeof(string)));
            predictions.Columns.Add(new DataColumn("LossType",
typeof(string)));
            predictions.Columns.Add(new DataColumn("ExploitRange",
typeof(string)));
            predictions.Columns.Add(new DataColumn("EntryCluster",
typeof(string)));
        }
    }
}
```

```

        predictions.Columns.Add(new DataColumn("PredictedImpact",
typeof(double)));
        predictions.Columns.Add(new
DataColumn("PredictedFrequency", typeof(int)));
        predictions.Columns.Add(new DataColumn("LossExpect",
typeof(double)));

        ds.Tables.Add(predictions);

    } // ManufacturePredictDataTable(DataSet ds)

    public static void ManufactureOrderedTable(DataSet ds)
    {
        DataTable entries = new DataTable("Ordered");
        entries.Columns.Add(new DataColumn("PredictedImpact",
typeof(double)));
        entries.Columns.Add(new DataColumn("PredictedFrequency",
typeof(double)));
        entries.Columns.Add(new DataColumn("LossExpectation",
typeof(double)));

        ds.Tables.Add("ordered");

        // ds.Tables.Add(prioritizedPredictions);

    } // ManufactureOrderedTable(DataSet ds)

    public frmTableRequest()
    {
        InitializeComponent();
    }

    private void btnLoadDta_Click(object sender, EventArgs e)
    {
        try
        {
            //// used as test data to query - may need to change
            this to get those based on the client's choice
            string selectQuery = "select distinct name as EntryID,"
+
                                "vuln_type as VulnType,loss_type
as LossType, exploit_range as ExploitRange " +
                                "from test02classifier where
month(published) = 1" +
                                "and year(published) = 2002";

            SqlConnection dtaConnection = new
SqlConnection(strDBConnString);

            SqlDataAdapter daCmd = new SqlDataAdapter(selectQuery,
dtaConnection);

```

```

        ///ManufactureEntryDataTable(ds); //Add the Entries
table to the dataset
        daCmd.Fill(ds, "Entries");
        this.vulnEntriesDataGridView.AutoGenerateColumns =
true;

        this.vulnEntriesDataGridView.DataSource = ds;
        this.vulnEntriesDataGridView.DataMember = "Entries";

        // do I need to close the connection?
        dtaConnection.Close();

        ManufacturePredictDataTable(ds);

    } // end try
    catch (Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.ToString());
    } //end catch

}

private void label1_Click(object sender, EventArgs e)
{
}

private void btnPredict_Click(object sender, EventArgs e)
{
    string strClusterLbl = "";
    double dblPredProb = 0;

    SqlConnection conn = new SqlConnection(strDBConnString);

    //delete content of predictions
    string del = "delete Predictions";
    SqlCommand cmdDel = new SqlCommand(del, conn);

    conn.Open();

    cmdDel.ExecuteScalar();
    conn.Close();

    foreach (DataRow row in ds.Tables["Entries"].Rows)
    {
        string strVulnType = row["VulnType"].ToString();
        string strLossType = row["LossType"].ToString();
        string strExploitRange =
row["ExploitRange"].ToString();

```

```

        IPredictElementCluster objRequester = new
PredictElementCluster();
        string strRequestCluster =
objRequester.Command(strASConnString, strVulnType.Trim(),
                                strLossType.Trim(),
strExploitRange.Trim());
        strClusterLbl = objRequester.getClusterNode;
        dblPredProb = objRequester.getPredictProbability;

        // Get the predicted values for this entry values
        ICalculateImpactFactor objRequestImpact = new
CalcImpactFactor();

        // will look at this when I get data from interface

        dtPredictDate =
Convert.ToDateTime(this.dateTimePicker1.Text.Trim());
        bool blconvertPeriod =
int.TryParse(this.txtMonths.Text, out intPeriods);
        string strImpactInfo =
objRequestImpact.Command(strDBConnString, dtPredictDate, intPeriods,
strClusterLbl);

        //get predicted Impact
        dblPredictImpact = objRequestImpact.getPredictedImpact;
        dblPredictFreq =
objRequestImpact.getPredictedOccurrences;
        double dblLossExpect = Math.Round(dblPredictImpact *
dblPredictFreq, 2);

        //inserting a new row with the values into the
Predictions Table

        DataRow predictRow = ds.Tables["Predictions"].NewRow();
        predictRow["EntryID"] = row["EntryID"];
        predictRow["VulnType"] = row["VulnType"];
        predictRow["LossType"] = row["LossType"];
        predictRow["ExploitRange"] = row["ExploitRange"];
        predictRow["EntryCluster"] = strClusterLbl;
        predictRow["PredictedImpact"] = dblPredictImpact;
        predictRow["PredictedFrequency"] = dblPredictFreq;
        predictRow["LossExpect"] = dblLossExpect;

        // used to add the new row to the prediction table
        ds.Tables["Predictions"].Rows.Add(predictRow);

        //Persist the data gotten 8/2/2008

        string ins = @"insert into
Predictions(entryID,vulnType,lossType,exploitRange,clusterNode,predImpa
ct,predFreq,lossExpect)

```

```

values(@entryID,@vulnType,@lossType,@exploitRange,@clusterNode,@predImp
act,@predFreq, @lossExp)";

        SqlCommand cmdIns = new SqlCommand(ins, conn);
        // not sure if this is necessary yet
cmdIns.CommandType = CommandType.TableDirect;
        // assign values to the parameters given
        SqlParameter objParameter = null;
        objParameter = cmdIns.CreateParameter();
        // objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "entryID";
        objParameter.Value = row["EntryID"];
        cmdIns.Parameters.Add(objParameter);

        objParameter = cmdIns.CreateParameter();
        // objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "vulnType";
        objParameter.Value = row["VulnType"];
        cmdIns.Parameters.Add(objParameter);

        objParameter = cmdIns.CreateParameter();
        // objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "lossType";
        objParameter.Value = row["LossType"];
        cmdIns.Parameters.Add(objParameter);

        objParameter = cmdIns.CreateParameter();
        // objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "exploitRange";
        objParameter.Value = row["ExploitRange"];
        cmdIns.Parameters.Add(objParameter);

        objParameter = cmdIns.CreateParameter();
        // objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "clusterNode";
        objParameter.Value = strClusterLbl; //
predictRow["EntryCluster"];
        cmdIns.Parameters.Add(objParameter);

        objParameter = cmdIns.CreateParameter();
        // objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "predImpact";

```

```

        objParameter.Value = dblPredictImpact; //
predictRow["PredictedImpact"];
        cmdIns.Parameters.Add(objParameter);

        objParameter = cmdIns.CreateParameter();
        // objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "predFreq";
        objParameter.Value = dblPredictFreq; //
predictRow["PredictedFrequency"];
        cmdIns.Parameters.Add(objParameter);

        objParameter = cmdIns.CreateParameter();
        // objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "lossExp";
        objParameter.Value = dblLossExpect; //
predictRow["PredictedFrequency"];
        cmdIns.Parameters.Add(objParameter);
        cmdIns.UpdatedRowSource = UpdateRowSource.Both;

        SqlDataAdapter da = new SqlDataAdapter(cmdIns);

        conn.Open();
        da.InsertCommand = cmdIns;
        // conn.Open();
        // cmdIns.ExecuteScalar();
        // cmdIns.ExecuteNonQuery();
        da.Update(ds, "Predictions");
        conn.Close();

    } // end foreach (DataRow row in ds.Tables["Entries"].Rows)

    this.predictedImpactDataGridView.AutoGenerateColumns =
true;
    this.predictedImpactDataGridView.DataSource = ds;
    this.predictedImpactDataGridView.DataMember =
"Predictions";

    // Aggregate each entry's prediction for a prioritized
listing 8/3/2008
    ManufactureOrderedTable(ds);
    //Create data adapter to fill the dataset table
    SqlDataAdapter daOrdered = new SqlDataAdapter();

    // Setup command for data adapter
    SqlCommand orderedCmd = new SqlCommand("spPrioritizedList",
conn);
    orderedCmd.CommandType = CommandType.StoredProcedure;
    orderedCmd.UpdatedRowSource = UpdateRowSource.None;

```

```

        daOrdered.SelectCommand = orderedCmd;
        daOrdered.Fill(ds, "Ordered");

        this.orderedListDataGridView.AutoGenerateColumns = true;
        this.orderedListDataGridView.DataSource = ds;
        this.orderedListDataGridView.DataMember = "Ordered";
        this.orderedListDataGridView.Visible = true;

    } //private void btnPredict_Click(object sender, EventArgs e)

        private string strASConnString =
SSRAMWinApp1.Properties.Settings.Default.AS_SSRAMConnectionString;
        private string strDBConnString =
Convert.ToString(SSRAMWinApp1.Properties.Settings.Default.DB_SSRAMConne
ctionString);
        private double dblPredictImpact = 0;
        private double dblPredictFreq = 0;
        private DateTime dtPredictDate;
        private int intPeriods = 0;
        private DataSet ds = new DataSet();

        private void button1_Click(object sender, EventArgs e)
        {

            //get the data from the Predictions data table into a
persistent state in the database

            SqlConnection conn = new SqlConnection(strDBConnString);
            //conn.Open();
            try
            {

                //delete content of predictions
                string del = "delete Predictions";
                SqlCommand cmdDel = new SqlCommand(del, conn);

                conn.Open();

                cmdDel.ExecuteScalar();
                conn.Close();

                // string strEntryID, strVulnType, strLossType,
strExploitRange, strClusterNode;
                // double dblPredImpact, dblPredFreq;

                foreach (DataRow row in ds.Tables["Predictions"].Rows)
                {

```

```

        string ins = @"insert into
Predictions(entryID,vulnType,lossType,exploitRange,clusterNode,predImpact,predFreq)

values(@entryID,@vulnType,@lossType,@exploitRange,@clusterNode,@predImpact,@predFreq)";

        SqlCommand cmdIns = new SqlCommand(ins, conn);
        // not sure if this is necessary yet
cmdIns.CommandType = CommandType.TableDirect;
        // assign values to the parameters given
        SqlParameter objParameter = null;
        objParameter = cmdIns.CreateParameter();
        objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "entryID";
        objParameter.Value = row["EntryID"];
        cmdIns.Parameters.Add(objParameter);

        objParameter = cmdIns.CreateParameter();
        objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "vulnType";
        objParameter.Value = row["VulnType"];
        cmdIns.Parameters.Add(objParameter);

        objParameter = cmdIns.CreateParameter();
        objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "lossType";
        objParameter.Value = row["LossType"];
        cmdIns.Parameters.Add(objParameter);

        objParameter = cmdIns.CreateParameter();
        objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "exploitRange";
        objParameter.Value = row["ExploitRange"];
        cmdIns.Parameters.Add(objParameter);

        objParameter = cmdIns.CreateParameter();
        objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "clusterNode";
        objParameter.Value = row["EntryCluster"];
        cmdIns.Parameters.Add(objParameter);

```



```

        objParameter = cmdIns.CreateParameter();
        objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "predImpact";
        objParameter.Value = row["PredictedImpact"];
        cmdIns.Parameters.Add(objParameter);

        objParameter = cmdIns.CreateParameter();
        objParameter.Direction =
System.Data.ParameterDirection.Input; //changed from
ParameterDirection.Input
        objParameter.ParameterName = "predFreq";
        objParameter.Value = row["PredictedFrequency"];
        cmdIns.Parameters.Add(objParameter);
        cmdIns.UpdatedRowSource = UpdateRowSource.None;

        SqlDataAdapter da = new SqlDataAdapter(cmdIns);

        conn.Open();
        da.InsertCommand = cmdIns;

        // conn.Open();
        // cmdIns.ExecuteScalar();
        // cmdIns.ExecuteNonQuery();
        da.Update(ds, "Predictions");
        conn.Close();

        } //foreach (DataRow row in
ds.Tables["Predictions"].Rows)
        } // try
        catch (Exception ex)
        {
            //Console.WriteLine("Error: " + e);
            System.Windows.Forms.MessageBox.Show(ex.ToString());

        } // catch
        finally
        {
            conn.Close();
        }

        Close();

    }
}

```

Appendix D-3 – Data for SSRAM Validation

To validate that there was no statistical difference between SSRAM's prediction and the actual result we ran a series of t-test of sample means. We used data reported in January of 2002 as our test cases. The following tables show the predictions and comparison to actual scores using the different classification schemes.

t-Test Results using Naïve Bayes algorithm

	<i>Actual</i>	<i>Predicted</i>
Mean	5.577419	6.23709
Variance	5.520473	2.362795
Observations	31	31
Pearson Correlation	0.68273	
Hypothesized Mean Difference	0	
df	30	
t Stat	-2.1378	
P(T<=t) one-tail	0.020399	
t Critical one-tail	1.697261	
P(T<=t) two-tail	0.040798	
t Critical two-tail	2.042272	

CVE-ID Actual	CVE-ID Predicted	Actual Score	Predicted	Difference
CVE-1999-1081	CVE-1999-1081	3.3	6.292	-2.992
CVE-1999-1091	CVE-1999-1091	3.3	4.8555	-1.5555
CVE-1999-1091	CVE-1999-1091	3.3	5.7886	-2.4886
CVE-2001-0887	CVE-2001-0887	2.3	4.8555	-2.5555
CVE-2001-0891	CVE-2001-0891	7	8.0093	-1.0093
CVE-2001-1457	CVE-2001-1457	7	8.0093	-1.0093
CVE-2002-0002	CVE-2002-0002	7	8.0093	-1.0093
CVE-2002-0005	CVE-2002-0005	10	8.0093	1.9907
CVE-2002-0007	CVE-2002-0007	10	6.72	3.28
CVE-2002-0008	CVE-2002-0008	7	6.72	0.28
CVE-2002-0008	CVE-2002-0008	7	8.0093	-1.0093
CVE-2002-0009	CVE-2002-0009	3.3	5.7886	-2.4886
CVE-2002-0009	CVE-2002-0009	3.3	6.292	-2.992
CVE-2002-0010	CVE-2002-0010	8	5.7886	2.2114
CVE-2002-0010	CVE-2002-0010	8	6.72	1.28
CVE-2002-0010	CVE-2002-0010	8	8.0093	-0.0093
CVE-2002-0011	CVE-2002-0011	3.3	6.292	-2.992
CVE-2002-0038	CVE-2002-0038	3.3	3.3	0
CVE-2002-0043	CVE-2002-0043	7	6.72	0.28
CVE-2002-0044	CVE-2002-0044	4.7	4.8555	-0.1555
CVE-2002-0045	CVE-2002-0045	8	5.7886	2.2114
CVE-2002-0045	CVE-2002-0045	8	8.0093	-0.0093
CVE-2002-0046	CVE-2002-0046	3.3	5.7886	-2.4886
CVE-2002-0047	CVE-2002-0047	3.3	3.3	0
CVE-2002-0077	CVE-2002-0077	7	6.72	0.28
CVE-2002-0077	CVE-2002-0077	7	8.0093	-1.0093
CVE-2002-1594	CVE-2002-1594	7	8.0093	-1.0093
CVE-2002-1595	CVE-2002-1595	3.3	6.292	-2.992
CVE-2002-1596	CVE-2002-1596	3.3	3.3	0
CVE-2002-1597	CVE-2002-1597	3.3	3.3	0
CVE-2002-1600	CVE-2002-1600	3.3	5.7886	-2.4886

t-Test: Paired Two Sample for Means using Neural Networks Classification Algorithm

	<i>Actual</i>	<i>Predicted</i>
Mean	5.653125	6.180540625
Variance	5.525796371	2.337423284
Observations	32	32
Pearson Correlation	0.629684793	
Hypothesized Mean Difference	0	
df	31	
t Stat	1.633197931	
P(T<=t) one-tail	0.056272384	
t Critical one-tail	1.695518742	
P(T<=t) two-tail	0.112544769	
t Critical two-tail	2.039513438	

		Actual	Predicted	Difference
CVE-1999-1081	CVE-1999-1081	3.3	6.292	-2.992
CVE-1999-1091	CVE-1999-1091	3.3	4.8555	-1.5555
CVE-1999-1091	CVE-1999-1091	3.3	5.7886	-2.4886
CVE-2001-0887	CVE-2001-0887	2.3	4.8555	-2.5555
CVE-2001-0891	CVE-2001-0891	7	8.0093	-1.0093
CVE-2001-1457	CVE-2001-1457	7	8.0093	-1.0093
CVE-2002-0002	CVE-2002-0002	7	8.0093	-1.0093
CVE-2002-0005	CVE-2002-0005	10	8.0093	1.9907
CVE-2002-0007	CVE-2002-0007	10	6.72	3.28
CVE-2002-0008	CVE-2002-0008	7	6.292	0.708
CVE-2002-0008	CVE-2002-0008	7	8.0093	-1.0093
CVE-2002-0009	CVE-2002-0009	3.3	5.7886	-2.4886
CVE-2002-0009	CVE-2002-0009	3.3	6.292	-2.992
CVE-2002-0010	CVE-2002-0010	8	5.7886	2.2114
CVE-2002-0010	CVE-2002-0010	8	6.72	1.28
CVE-2002-0010	CVE-2002-0010	8	8.0093	-0.0093
CVE-2002-0011	CVE-2002-0011	3.3	6.292	-2.992
CVE-2002-0038	CVE-2002-0038	3.3	3.3	0
CVE-2002-0043	CVE-2002-0043	7	6.72	0.28
CVE-2002-0044	CVE-2002-0044	4.7	4.8555	-0.1555
CVE-2002-0045	CVE-2002-0045	8	4.8555	3.1445
CVE-2002-0045	CVE-2002-0045	8	5.7886	2.2114
CVE-2002-0045	CVE-2002-0045	8	8.0093	-0.0093
CVE-2002-0046	CVE-2002-0046	3.3	5.7886	-2.4886
CVE-2002-0047	CVE-2002-0047	3.3	3.3	0
CVE-2002-0077	CVE-2002-0077	7	6.72	0.28
CVE-2002-0077	CVE-2002-0077	7	8.0093	-1.0093
CVE-2002-1594	CVE-2002-1594	7	8.0093	-1.0093
CVE-2002-1595	CVE-2002-1595	3.3	6.292	-2.992
CVE-2002-1596	CVE-2002-1596	3.3	3.3	0
CVE-2002-1597	CVE-2002-1597	3.3	3.3	0
CVE-2002-1600	CVE-2002-1600	3.3	5.7886	-2.4886

t-Test: Paired Two Sample for Means using Decision Tree Classifier

	<i>Actual</i>	<i>Predicted</i>
Mean	5.76129	6.102877
Variance	5.636452	2.316325
Observations	31	31
Pearson Correlation	0.658869	
Hypothesized Mean Difference	0	
df	30	
t Stat	-1.06461	
P(T<=t) one-tail	0.147773	
t Critical one-tail	1.697261	
P(T<=t) two-tail	0.295546	
t Critical two-tail	2.042272	

CVE-ID		Actual	Predicted	Difference
CVE-1999-1081	CVE-1999-1081	3.3	5.7886	-2.4886
CVE-1999-1091	CVE-1999-1091	3.3	4.8555	-1.5555
CVE-1999-1091	CVE-1999-1091	3.3	6.292	-2.992
CVE-2001-0887	CVE-2001-0887	2.3	4.8555	-2.5555
CVE-2001-0891	CVE-2001-0891	7	8.0093	-1.0093
CVE-2001-1457	CVE-2001-1457	7	8.0093	-1.0093
CVE-2002-0002	CVE-2002-0002	7	8.0093	-1.0093
CVE-2002-0005	CVE-2002-0005	10	8.0093	1.9907
CVE-2002-0007	CVE-2002-0007	10	6.72	3.28
CVE-2002-0008	CVE-2002-0008	7	6.72	0.28
CVE-2002-0008	CVE-2002-0008	7	8.0093	-1.0093
CVE-2002-0009	CVE-2002-0009	3.3	5.7886	-2.4886
CVE-2002-0010	CVE-2002-0010	8	5.7886	2.2114
CVE-2002-0010	CVE-2002-0010	8	6.292	1.708
CVE-2002-0010	CVE-2002-0010	8	6.72	1.28
CVE-2002-0010	CVE-2002-0010	8	8.0093	-0.0093
CVE-2002-0011	CVE-2002-0011	3.3	5.7886	-2.4886
CVE-2002-0038	CVE-2002-0038	3.3	3.3	0
CVE-2002-0043	CVE-2002-0043	7	6.72	0.28
CVE-2002-0044	CVE-2002-0044	4.7	4.8555	-0.1555
CVE-2002-0045	CVE-2002-0045	8	4.8555	3.1445
CVE-2002-0045	CVE-2002-0045	8	5.7886	2.2114
CVE-2002-0045	CVE-2002-0045	8	8.0093	-0.0093
CVE-2002-0046	CVE-2002-0046	3.3	5.7886	-2.4886
CVE-2002-0047	CVE-2002-0047	3.3	3.3	0
CVE-2002-0077	CVE-2002-0077	7	6.72	0.28
CVE-2002-1594	CVE-2002-1594	7	8.0093	-1.0093
CVE-2002-1595	CVE-2002-1595	3.3	5.7886	-2.4886
CVE-2002-1596	CVE-2002-1596	3.3	3.3	0
CVE-2002-1597	CVE-2002-1597	3.3	3.3	0
CVE-2002-1600	CVE-2002-1600	3.3	5.7886	-2.4886

Mean -0.4385
Std Dev 1.7385
df 35.0000
t -1.51354

Entry-ID	Cluster Node	Actual Score	Predicted Impact	Difference
CVE-1999-1081	Cluster 2	3.3	5.1	-1.770
CVE-1999-1091	Cluster 5	3.3	3.8	-0.518
CVE-1999-1091	Cluster 3	3.3	2.4	0.893
CVE-2001-0887	Cluster 5	2.3	3.8	-1.518
CVE-2001-0887	Cluster 5	2.3	3.8	-1.518
CVE-2001-0891	Cluster 1	7	8.1	-1.058
CVE-2001-1457	Cluster 1	7	8.1	-1.058
CVE-2002-0002	Cluster 1	7	8.1	-1.058
CVE-2002-0005	Cluster 1	10	8.1	1.942
CVE-2002-0007	Cluster 6	10	5.6	4.387
CVE-2002-0008	Cluster 1	7	8.1	-1.058
CVE-2002-0008	Cluster 6	7	5.6	1.387
CVE-2002-0009	Cluster 2	3.3	5.1	-1.770
CVE-2002-0009	Cluster 2	3.3	5.1	-1.770
CVE-2002-0010	Cluster 3	8	2.4	5.593
CVE-2002-0010	Cluster 6	8	5.6	2.387
CVE-2002-0010	Cluster 2	8	5.1	2.930
CVE-2002-0010	Cluster 1	8	8.1	-0.058
CVE-2002-0011	Cluster 2	3.3	5.1	-1.770
CVE-2002-0038	Cluster 7	3.3	3.3	0.000
CVE-2002-0043	Cluster 6	7	5.6	1.387
CVE-2002-0044	Cluster 5	4.7	3.8	0.882
CVE-2002-0044	Cluster 5	4.7	3.8	0.882
CVE-2002-0045	Cluster 5	8	3.8	4.182
CVE-2002-0045	Cluster 2	8	5.1	2.930
CVE-2002-0045	Cluster 1	8	8.1	-0.058
CVE-2002-0045	Cluster 1	8	8.1	-0.058
CVE-2002-0046	Cluster 2	3.3	5.1	-1.770
CVE-2002-0047	Cluster 7	3.3	3.3	0.000
CVE-2002-0077	Cluster 6	7	5.6	1.387
CVE-2002-0077	Cluster 6	7	5.6	1.387
CVE-2002-1594	Cluster 1	7	8.1	-1.058
CVE-2002-1595	Cluster 2	3.3	5.1	-1.770
CVE-2002-1596	Cluster 7	3.3	3.3	0.000
CVE-2002-1597	Cluster 7	3.3	3.3	0.000
CVE-2002-1600	Cluster 2	3.3	5.1	-1.770

Appendix E-1 – Microsoft’s TAMT Listing of Vulnerabilities

This appendix shows the report generated from using Microsoft’s Threat Analysis and Modeling Tool.

Confidentiality Threats	
Unauthorized disclosure of <creates a unique ballot for his/herself> using <speech user interface> by <Validated voter>	
Countermeasures	
<input type="checkbox"/> Canonicalization : Only accept primitive typed identified (e.g., integers) which are mapped to filenames	
Unauthorized disclosure of <creates a unique ballot for his/herself> using <Graphical user interface> by <Validated voter>	
Countermeasures	
<input type="checkbox"/> Canonicalization : Only accept primitive typed identified (e.g., integers) which are mapped to filenames	
Unauthorized disclosure of <completes ballot choices> using <speech user interface> by <Validated voter>	
Countermeasures	
<input type="checkbox"/> Canonicalization : Only accept primitive typed identified (e.g., integers) which are mapped to filenames	
Unauthorized disclosure of <completes ballot choices> using <Graphical user interface> by <Validated voter>	
Countermeasures	
<input type="checkbox"/> Canonicalization : Only accept primitive typed identified (e.g., integers) which are mapped to filenames	
Unauthorized disclosure of <prints ballot entry> using <Printer> by <Validated voter>	
Countermeasures	
<input type="checkbox"/> Canonicalization : Only accept primitive typed identified (e.g., integers) which are mapped to filenames	
Unauthorized disclosure of <deletes his/her's ballot entry> using <speech user interface> by <Validated voter>	
Countermeasures	
<input type="checkbox"/> Canonicalization : Only accept primitive typed identified (e.g., integers) which are	

	mapped to filenames	
Unauthorized disclosure of <deletes his/her's ballot entry> using <Graphical user interface> by <Validated voter>		
	Countermeasures	
	<input type="checkbox"/> Canonicalization : Only accept primitive typed identified (e.g., integers) which are mapped to filenames	
Unauthorized disclosure of <Create Ballot counter> using <Secure counter> by <ballot tally>		
	Countermeasures	
	<input type="checkbox"/> Buffer Overflow : Use safe functions such as strncpy, strncat instead of strcpy, strcat	
	<input type="checkbox"/> Buffer Overflow : Validation on input should be performed on the input	
	<input type="checkbox"/> Canonicalization : Only accept primitive typed identified (e.g., integers) which are mapped to filenames	
	<input type="checkbox"/> Cryptanalysis Attacks : Use well-known implementations of well-known cryptographic algorithms	
	<input type="checkbox"/> Cryptanalysis Attacks : Use cryptographically generated random keys	
	<input type="checkbox"/> Cryptanalysis Attacks : Utilize platform supplied feature to store secret key (e.g., DPAPI)	
	<input type="checkbox"/> Cryptanalysis Attacks : Utilize SSL or IPSec w/ Encryption to establish a secure communication channel	
	<input type="checkbox"/> Format String : Use a managed language	
	<input type="checkbox"/> Integer Overflow/Underflow : Use Language features	
Unauthorized disclosure of <reads ballot counter> using <Secure counter> by <ballot tally>		
	Countermeasures	
	<input type="checkbox"/> Buffer Overflow : Use safe functions such as strncpy, strncat instead of strcpy, strcat	
	<input type="checkbox"/> Buffer Overflow : Validation on input should be performed on the input	
	<input type="checkbox"/> Canonicalization : Only accept primitive typed identified (e.g., integers) which are mapped to filenames	
	<input type="checkbox"/> Cryptanalysis Attacks : Use well-known implementations of well-known cryptographic algorithms	
	<input type="checkbox"/> Cryptanalysis Attacks : Use cryptographically generated random keys	

	<div> <input type="checkbox"/> Cryptanalysis Attacks : Utilize platform supplied feature to store secret key (e.g., DPAPI) </div> <div> <input type="checkbox"/> Cryptanalysis Attacks : Utilize SSL or IPSec w/ Encryption to establish a secure communication channel </div> <div> <input type="checkbox"/> Format String : Use a managed language </div> <div> <input type="checkbox"/> Integer Overflow/Underflow : Use Language features </div>	
Unauthorized disclosure of <updates ballot counter> using <Secure counter> by <ballot tally>		
	<div> Countermeasures </div> <div> <input type="checkbox"/> Buffer Overflow : Use safe functions such as strncpy, strncat instead of strcpy, strcat </div> <div> <input type="checkbox"/> Buffer Overflow : Validation on input should be performed on the input </div> <div> <input type="checkbox"/> Canonicalization : Only accept primitive typed identified (e.g., integers) which are mapped to filenames </div> <div> <input type="checkbox"/> Cryptanalysis Attacks : Use well-known implementations of well-known cryptographic algorithms </div> <div> <input type="checkbox"/> Cryptanalysis Attacks : Use cryptographically generated random keys </div> <div> <input type="checkbox"/> Cryptanalysis Attacks : Utilize platform supplied feature to store secret key (e.g., DPAPI) </div> <div> <input type="checkbox"/> Cryptanalysis Attacks : Utilize SSL or IPSec w/ Encryption to establish a secure communication channel </div> <div> <input type="checkbox"/> Format String : Use a managed language </div> <div> <input type="checkbox"/> Integer Overflow/Underflow : Use Language features </div>	
Unauthorized disclosure of <creates imposter file> using <imposter file> by <ballot tally>		
	<div> Countermeasures </div> <div> <input type="checkbox"/> Buffer Overflow : Use safe functions such as strncpy, strncat instead of strcpy, strcat </div> <div> <input type="checkbox"/> Buffer Overflow : Validation on input should be performed on the input </div> <div> <input type="checkbox"/> Canonicalization : Only accept primitive typed identified (e.g., integers) which are mapped to filenames </div> <div> <input type="checkbox"/> Cryptanalysis Attacks : Use well-known implementations of well-known cryptographic algorithms </div>	

	<div><input type="checkbox"/> Cryptanalysis Attacks : Use cryptographically generated random keys</div> <div><input type="checkbox"/> Cryptanalysis Attacks : Utilize platform supplied feature to store secret key (e.g., DPAPI)</div> <div><input type="checkbox"/> Cryptanalysis Attacks : Utilize SSL or IPSec w/ Encryption to establish a secure communication channel</div> <div><input type="checkbox"/> Format String : Use a managed language</div> <div><input type="checkbox"/> Integer Overflow/Underflow : Use Language features</div>	
<div>Unauthorized disclosure of <reads imposter file> using <imposter file> by <ballot tally></div> <div>Countermeasures</div> <div><input type="checkbox"/> Buffer Overflow : Use safe functions such as strncpy, strncat instead of strcpy, strcat</div> <div><input type="checkbox"/> Buffer Overflow : Validation on input should be performed on the input</div> <div><input type="checkbox"/> Canonicalization : Only accept primitive typed identified (e.g., integers) which are mapped to filenames</div> <div><input type="checkbox"/> Cryptanalysis Attacks : Use well-known implementations of well-known cryptographic algorithms</div> <div><input type="checkbox"/> Cryptanalysis Attacks : Use cryptographically generated random keys</div> <div><input type="checkbox"/> Cryptanalysis Attacks : Utilize platform supplied feature to store secret key (e.g., DPAPI)</div> <div><input type="checkbox"/> Cryptanalysis Attacks : Utilize SSL or IPSec w/ Encryption to establish a secure communication channel</div> <div><input type="checkbox"/> Format String : Use a managed language</div> <div><input type="checkbox"/> Integer Overflow/Underflow : Use Language features</div>		
<div>Unauthorized disclosure of <updates imposter file> using <imposter file> by <ballot tally></div> <div>Countermeasures</div> <div><input type="checkbox"/> Buffer Overflow : Use safe functions such as strncpy, strncat instead of strcpy, strcat</div> <div><input type="checkbox"/> Buffer Overflow : Validation on input should be performed on the input</div> <div><input type="checkbox"/> Canonicalization : Only accept primitive typed identified (e.g., integers) which are mapped to filenames</div> <div><input type="checkbox"/> Cryptanalysis Attacks : Use well-known implementations of well-known</div>		

	cryptographic algorithms	
<input type="checkbox"/>	Cryptanalysis Attacks : Use cryptographically generated random keys	
<input type="checkbox"/>	Cryptanalysis Attacks : Utilize platform supplied feature to store secret key (e.g., DPAPI)	
<input type="checkbox"/>	Cryptanalysis Attacks : Utilize SSL or IPSec w/ Encryption to establish a secure communication channel	
<input type="checkbox"/>	Format String : Use a managed language	
<input type="checkbox"/>	Integer Overflow/Underflow : Use Language features	
Unauthorized disclosure of <Prints total tallies for ballots entered> using <Printer> by <Poll Worker >		
	Countermeasures	
<input type="checkbox"/>	Canonicalization : Only accept primitive typed identified (e.g., integers) which are mapped to filenames	

Appendix E-2 – Prime III Data

ID	Name	VulnType	Loss Type	Exploit Range
Threat-1	Canonicalization	input	Avail	user_init
Threat-1	Canonicalization	input	Avail	local
Threat-1	Canonicalization	input	Int	user_init
Threat-1	Canonicalization	input	Int	local
Threat-1	Canonicalization	input	Conf	user_init
Threat-1	Canonicalization	input	Conf	local
Threat-2	Buffer Overflow	access	Int	user_init
Threat-2	Buffer Overflow	input	Int	user_init
Threat-2	Buffer Overflow	access	Int	local
Threat-2	Buffer Overflow	input	Int	local
Threat-2	Buffer Overflow	access	Avail	user_init
Threat-2	Buffer Overflow	input	Avail	user_init
Threat-2	Buffer Overflow	access	Avail	local
Threat-2	Buffer Overflow	input	Avail	local
Threat-2	Buffer Overflow	access	Conf	user_init
Threat-2	Buffer Overflow	input	Conf	user_init
Threat-2	Buffer Overflow	access	Conf	local
Threat-2	Buffer Overflow	input	Conf	local
Threat-3	Cryptanalysis	env	Avail	local
Threat-3	Cryptanalysis	access	Avail	local
Threat-3	Cryptanalysis	env	Avail	local
Threat-3	Cryptanalysis	access	Avail	local
Threat-3	Cryptanalysis	env	Int	local
Threat-3	Cryptanalysis	access	Int	local
Threat-3	Cryptanalysis	env	Int	local
Threat-3	Cryptanalysis	access	Int	local
Threat-3	Cryptanalysis	env	Conf	local
Threat-3	Cryptanalysis	access	Conf	local
Threat-3	Cryptanalysis	env	Conf	local
Threat-3	Cryptanalysis	access	Conf	local
Threat-3	Cryptanalysis	env	Sec Prot	local
Threat-3	Cryptanalysis	access	Sec Prot	local
Threat-3	Cryptanalysis	env	Sec Prot	local
Threat-3	Cryptanalysis	access	Sec Prot	local
Threat-4	Format String	design	Avail	user_init
Threat-4	Format String	design	Avail	local
Threat-4	Format String	design	Int	user_init
Threat-4	Format String	design	Int	local
Threat-4	Format String	design	Conf	user_init
Threat-4	Format String	design	Conf	local
Threat-5	Integer Overflow/Underflow	input	Avail	user_init
Threat-5	Integer Overflow/Underflow	input	Int	user_init
Threat-5	Integer Overflow/Underflow	input	Conf	user_init

Appendix E-3 – Prime III Predictions

This appendix shows the result of making predictions for our case study using the different classification schemes. We ran the predictions based on two separate historical data sources (1996 – 2001, 2003 – 2005). We performed predictions for 2002 and 2006 based on the same vulnerability listing.

The resulting predictions are shown in this appendix along with the confidence interval for the two separate historical basis for predictions.

A. Using 2003-2005 data as training for 2006 Predictions with Neural Network algorithm for classification

The screenshot shows a software application titled "Vulnerability Listing". It contains two main data tables: "Vulnerability Entries" and "PrioritizedList". Below these is a "Predicted Values" table. At the bottom, there are controls for "LoadData", "PredictDate" (set to 1/ 1/2006), "Periods (Months)" (set to 12), and a "Predict" button.

Vulnerability Entries					PrioritizedList					
EntryID	VulnType	LossType	ExploitRange		entryID	PredImpact	Incidences	FreqGrowth	PredFreq	LossExpect
Buffer Overflow	access	Avail	local		Integer Overflow/	7.55	970	0.78	1727	13038.68
Buffer Overflow	access	Avail	user_init		Canonicalization	5.91	810	0.66	1384	9300.21
Buffer Overflow	access	Conf	local		Buffer Overflow	5.71	785	0.65	1331	8690.62
Buffer Overflow	access	Conf	user_init		Format String	5.23	675	0.62	1094	5723.59
Buffer Overflow	access	Int	local		Cryptanalysis	4.54	642	0.56	1024	5175.36

Predicted Values									
EntryID	VulnType	LossType	ExploitRange	EntryCluster	PredictedImpact	Incidences	FreqGrowthRate	PredictedFreq	
Buffer Overflow	access	Avail	local	Cluster 3	5.2318	675	0.62	1094	
Buffer Overflow	access	Avail	user_init	Cluster 2	7.5499	970	0.78	1727	
Buffer Overflow	access	Conf	local	Cluster 5	2.6159	491	0.42	697	
Canonicalization	input	Conf	local	Cluster 5	2.6159	491	0.42	697	
Canonicalization	input	Conf	user_init	Cluster 2	7.5499	970	0.78	1727	
Cryptanalysis	access	Sec_Prot	local	Cluster 3	5.2318	675	0.62	1094	
Cryptanalysis	env	Avail	local	Cluster 5	2.6159	491	0.42	697	

The prioritized list shows Integer overflow/underflow has the highest loss expectation based on the impact score and predicted frequency. Countermeasures for dealing with integer overflow/underflow, such as making sure that the programming language features that deal with minimizing overflow/underflow, should be employed. Canonicalization, Buffer Overflow and Format String would be considered of the same priority given the

confidence interval for the historical data (2003 – 2005) used for the prediction. Since these are above 5.0, the first four vulnerabilities would be considered medium to high risk elements and, as such, require that countermeasures to ameliorate them be given.

B. Using 2003-2005 data as training for 2006 Predictions with Decision Tree algorithm for classification

Vulnerability Listing

Vulnerability Entries

EntryID	VulnType	LossType	ExploitRange
Buffer Overflow	access	Avail	local
Buffer Overflow	input	Int	user_init
Canonicalization	input	Avail	local
Canonicalization	input	Avail	user_init

PrioritizedList

entryID	PredIm	Incidences	FreqGrowth	PredFreq	LossExpect
Buffer Overflow	6.06	1069	0.66	1756	10546.55
Canonicalization	4.56...	1168	0.55	1784	8054.41
Integer Overflow/Underflow	4.56...	1168	0.55	1784	8054.41
Format String	5.23	675	0.62	1094	5723.59
Cryptanalysis	3.92	583	0.52	896	3773.43

Predicted Values

EntryID	VulnType	LossType	ExploitRange	EntryCluster	PredictedImpact	Incidences	FreqGrowthRate	PredictedFreq
Buffer Overflow	access	Avail	local	Cluster 2	7.5499	970	0.78	1727
Buffer Overflow	input	Conf	local	Cluster 6	3.068	1268	0.43	1813
Canonicalization	input	Avail	local	Cluster 2	7.5499	970	0.78	1727
Canonicalization	input	Conf	local	Cluster 6	3.068	1268	0.43	1813
Cryptanalysis	access	Avail	local	Cluster 3	5.2318	675	0.62	1094
Cryptanalysis	env	Avail	local	Cluster 5	2.6159	491	0.42	697

LoadData

PredictDate

1/ 1/2006

Periods (Months)

12

Predict

Using the decision tree algorithm, Buffer Overflow would be considered the highest risk element, due to its high predicted impact while Canonicalization and Integer Overflow/Underflow would be of the same level of priority. Although Format String has a higher impact factor, the predicted number of occurrences warrants it to be placed at a lower priority. Cryptanalysis in both measures was given the lowest priority.

Training Data by Clusters (2003 - 2005)					With 95% Confidence Interval	
Number in Cluster	Avg Score	Std Deviation	Margin of Error	Cluster Node	Lower	Upper
2083.000	7.599	1.673	0.072	Cluster 2	7.527	7.671
1430.000	6.967	0.831	0.043	Cluster 1	6.924	7.010
1616.000	5.379	2.075	0.101	Cluster 3	5.277	5.480
371.000	4.159	2.088	0.212	Cluster 7	3.947	4.372
17.000	3.300	0.000	0.000	Cluster 8	3.300	3.300
1031.000	3.124	0.776	0.047	Cluster 4	3.077	3.172
1764.000	3.013	0.551	0.026	Cluster 6	2.987	3.039
669.000	2.631	0.932	0.071	Cluster 5	2.561	2.702

Confidence Interval for Predictions on 2006 data based on 2003-2005 data

C. Using 1996-2001 data as training for 2001 Predictions with Neural Network algorithm for classification

EntryID	VulnType	LossType	ExploitRange	entryID	PredImpact	PredFreq	LossExpect
Buffer Overflow	access	Avail	local	Integer Overflow/Underflow	6.16	312	2084.09
Buffer Overflow	access	Avail	user_init	Canonicalization	5.6	262	1594.28
Buffer Overflow	access	Conf	local	Cryptanalysis	5.48	242	1529.39
Buffer Overflow	access	Conf	user_init	Buffer Overflow	5.15	200	1118.95
Buffer Overflow	access	Int	local	Format String	4.7	141	659.46

EntryID	VulnType	LossType	ExploitRange	EntryCluster	PredictedImpact	PredictedFrequency	LossExpect
Buffer Overflow	access	Avail	local	Cluster 4	4.7895	112	536.42
Buffer Overflow	access	Conf	local	Cluster 5	4.7841	132	631.5
Buffer Overflow	access	Conf	user_init	Cluster 3	4.5144	169	762.93
Buffer Overflow	input	Avail	local	Cluster 4	4.7895	112	536.42
Buffer Overflow	input	Conf	user_init	Cluster 2	5.5296	388	2145.48
Buffer Overflow	input	Int	user_init	Cluster 1	8.1702	437	3570.38
Canonicalization	input	Avail	local	Cluster 4	4.7895	112	536.42

LoadData PredictDate 1/1/2001 Periods (Months) 12 Predict

The prioritized list for 2001 shows Integer Overflow/underflow as the highest vulnerability risk element for January 2001. Canonicalization, Cryptanalysis and Buffer Overflow would be interpreted as having the same level of priority given the confidence interval data for the training period.

Using Decision Tree Classification Algorithm

The screenshot shows the 'Vulnerability Listing' application. The 'Predicted Values' table is displayed, showing the results of a decision tree classification algorithm for January 2001. The table includes columns for EntryID, VulnType, LossType, ExploitRange, EntryCluster, PredictedImpact, PredictedFrequency, and LossExpect.

EntryID	VulnType	LossType	ExploitRange	EntryCluster	PredictedImpact	PredictedFrequency	LossExpect
Buffer Overflow	access	Avail	local	Cluster 4	6.6126	173	1143.98
Buffer Overflow	access	Avail	user_init	Cluster 7	7	23	161
Buffer Overflow	access	Conf	local	Cluster 5	5.8512	327	1913.34
Buffer Overflow	access	Conf	user_init	Cluster 2	8.3457	300	2503.71
Buffer Overflow	access	Int	local	Cluster 5	5.8512	327	1913.34
Buffer Overflow	access	Int	user_init	Cluster 2	8.3457	300	2503.71
Buffer Overflow	input	Avail	local	Cluster 4	6.6126	173	1143.98
Buffer Overflow	input	Avail	user_init	Cluster 7	7	23	161

Using decision tree, Format String would be considered the element of highest risk due to the predicted number of occurrences, though Canonicalization, Buffer Overflow and Integer Overflow and Underflow all had higher impact predictions. These three would be considered of the same priority level also.

The screenshot shows the 'Vulnerability Listing' application. The 'Predicted Values' table is displayed, showing the results of a decision tree classification algorithm for January 2002. The table includes columns for EntryID, VulnType, LossType, ExploitRange, EntryCluster, PredictedImpact, Incidences, FreqGrowthRate, PredictedFrequency, and LossExpect.

EntryID	VulnType	LossType	ExploitRange	EntryCluster	PredictedImpact	Incidences	FreqGrowthRate	PredictedFrequency	LossExpect
Buffer Overflow	access	Avail	local	Cluster 4	7.36	385	0.28	493	3628.48
Buffer Overflow	access	Avail	user_init	Cluster 7	7	76	0.18	90	630
Buffer Overflow	access	Conf	local	Cluster 5	5.876	321	0.14	366	2150.62
Buffer Overflow	access	Conf	user_init	Cluster 2	8.3232	516	1.36	1218	10137.66
Canonicalization	input	Avail	local	Cluster 4	7.36	385	0.28	493	3628.48
Canonicalization	input	Avail	user_init	Cluster 7	7	76	0.18	90	630
Canonicalization	input	Conf	local	Cluster 2	8.3232	516	1.36	1218	10137.66
Canonicalization	input	Conf	user_init	Cluster 2	8.3232	516	1.36	1218	10137.66

To actually predict for data not within our training data, we predicted for January 2002.

Using the neural network classification algorithm we found that, Integer

Overflow/Underflow, Canonicalization, and Buffer Overflow were predicted with high impact scores of the same level of priority, while Format String and Cryptanalysis though of lower priority still had impact scores in the medium to high range. In this case, we would recommend that all the vulnerabilities' countermeasures be considered in the priority listed but, in essence, allow the decision makers to understand why they should all be addressed.

Training Data by Clusters (1996 - 2001)					With 95% Confidence Interval	
NumInCluster	CVSS_Score	StdDev	Margin of Error	Cluster	Lower	Upper
1257	8.2541	1.8124	0.1002	Cluster 2	8.1539	8.3543
719	7.0872	1.7984	0.1315	Cluster 4	6.9557	7.2186
107	7	0	0	Cluster 7	7	7
765	5.5393	2.0795	0.1474	Cluster 5	5.3919	5.6867
1298	5.3409	1.9665	0.107	Cluster 1	5.2339	5.4478
1250	4.3934	2.0171	0.1118	Cluster 3	4.2816	4.5052
735	4.3105	1.6957	0.1226	Cluster 6	4.1879	4.4331

Appendix F- Vulnerability Classification

Kindoms [Tsipenyuk et al. 2005]	Vulnerability Class [NVD]	19 Sins [Howard et al. 2005]	OWASP
Input validation and representation	Input validation error – (boundary condition error, buffer overflow)	Buffer overflows, command injection, cross-site scripting, format string problems, integer range errors, SQL injection	Buffer overflows, cross-site scripting flaws, injection flaws, unvalidated input
API abuse	Access validation error	Trusting network address information	Broken access control, insecure storage
Security features		Failing to protect network traffic, failing to store and protect data, failing to use cryptographically strong random numbers, improper file access, improper use of SQL, use of weak password-based systems, unauthenticated key exchange	
Time and State	Race condition	Signal race conditions, use of “magic” URLs and hidden forms	Broken authentication and session management
Errors	Exceptional condition error	Failure to handle errors	Improper error handling
Code quality	Design error	Poor usability	Denial of service
Encapsulation Environment	Environmental error	Information leakage	Insecure configuration management
	Other error		