Security Metrics for Software Systems

Ju An Wang, Hao Wang,
Minzhe Guo, and Min Xia
Southern Polytechnic State University
1100 South Marietta Parkway
Marietta, GA 30060-2896, USA
01-678-915-3718
{jwang, hwang mguo,
mxia}@spsu.edu

ABSTRACT
Security metrics for software products provide quantitative measurement for the degree of trustworthiness for software systems. This paper proposes a new approach to define software security metrics based on vulnerabilities included in the software systems and their impacts on software quality. We use the Common Vulnerabilities and Exposures (CVE), an industry standard for vulnerability and exposure names, and the Common Vulnerability Scoring System (CVSS), a vulnerability scoring system designed to provide an open and standardized method for rating software vulnerabilities, in our metric definition and calculation. Examples are provided in the paper, which show that our definition of security metrics is consistent with the common practice and real-world experience about software quality in trustworthiness.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics – product metrics, security measures; D.2.19 [**Software Engineering**]: Software Quality – Methods for SQA, Measurement applied to SQA

General Terms
Software Security

Keywords
Software vulnerabilities, Security metrics, Software quality

1. INTRODUCTION
Software is essential to the operation of the Nation's critical infrastructure. Vulnerabilities in software can jeopardize intellectual property, consumer trust, and business operations and services. Additionally, a broad spectrum of critical applications and infrastructure, from process control systems to commercial application products, depend on secure, reliable software. It is estimated that 90 percent of reported security incidents result from exploits against defects in the design or code of software. Therefore, ensuring the integrity of software is critical to protecting the infrastructure from threats and vulnerabilities, and reducing overall risk to cyber attacks. In order to ensure system reliability, integrity, and safety, it is critical to include provisions for built-in security of the enabling software. With the advances of computer hardware, the security and dependability of a computing system rely heavily on its software. The current state of the arts of software technology has not reached the same level as its hardware counterpart in terms of reliability and security.

Metrics are quantifiable measurement. Security metrics are quantitative indicators for the security attributes of an information system or technology. Metrics helps us understand quality and consistency. Metrics provides a universal way to exchange ideas, to measure the product or service quality, and to improve a process. We cannot improve security if we cannot measure it. However, measuring security is hard because the discipline itself is still in the early stage of development. To date there are few documented resources and existing work on software security metrics. There are a great variety of different vulnerabilities existing for different kinds of software. Each vulnerability or exposure has different impact on the quality and security attributes of the software product such as confidentiality, integrity, availability, and so on. Another challenge is to validate the defined security metrics, comparing different metrics definitions. Finally, lack of tool support represents yet another challenge in our research. We strongly believe that it is essential to automate the process of security management to make it more efficient and less error-prone. We would like to implement a software tool delivering the security metrics for a given software system automatically, or at least semi-automatically with a user friendly graphical user interface. We also expect our approach is general enough to measure security metrics for reusable software components as well as software systems.

Software security involves internal weakness and external attacks. The external threat agents often break a software system by exploiting its internal weakness, i.e., the software vulnerabilities. Therefore, our research focuses on software vulnerabilities that become fundamental indicators for the level of trustworthiness of the software. There are a great variety of software vulnerabilities discovered over times. Our approach is to select representative weakness that reflect the software security level. We use the Common Vulnerabilities and Exposures (CVE) [6-9] lists to identify the weakness included in the software system during its lifecycle. Obviously more vulnerabilities discovered in a software system would lead higher potential risks for the software system. Considering the fact that different vulnerabilities may have

ACMSE '09 March 19-21, 2009, Clemson, SC, USA.

different consequences to security, we want to assess the severity of vulnerabilities, focusing on their likelihood to be exploited.

The remainder of the paper is organized as follows. Section 2 introduce software vulnerability concepts and terminology. Section 3 presents our security metrics formula for software systems. Section 4 compares two common browsers, Internet Explorer and Firefox, in terms of their security metrics. Finally in Section 5 we draw conclusions and discuss some research topics.

2. SOFTWARE VULNERABILITIES

Vulnerability evaluation plays a central role for security posture and risk management. Vulnerability refers to flaws or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy. Any flaw or weakness in an information system could be exploited to gain unauthorized access to, damage or compromise the information system. In order to evaluate vulnerability, we need well-defined security metrics to measure the severity level of a vulnerability based on scientific, systematic, and quantitative approaches. Without well-defined security metrics, companies find themselves difficult to compare and select different security options accurately. Cost-benefit analysis and ROI (Return on Investment) calculations are becoming standard pre-requisites for any information security product sale or purchase.

The CVSS (Common Vulnerability Scoring System) [1] provides a tool to quantify the severity and risk of a vulnerability to an information asset in a computing environment. It was designed by NIST (National Institute of Standard and Technology) and a team of industry partners. CVSS metrics for vulnerabilities are divided into three groups: Base metrics measure the intrinsic and fundamental characteristics of vulnerabilities that do not change over time or in different environments. Temporal metrics measure those attributes of vulnerabilities that change over time but do not change among user environments. Environmental metrics measure those vulnerability characteristics that are relevant and unique to a particular user's environment.

There are six base metrics that capture the most fundamental features of a vulnerability:

(1) Access Vector (AV): It measures how the vulnerability is exploited, for instance, locally or remotely. The more remote an attacker can be to attack an information asset, the greater the vulnerability score.

(2) Access Complexity (AC): It measures the complexity of the attack required to exploit the vulnerability once an attacker has gained access to the target system. The lower the required complexity, the higher the vulnerability score.

(3) Authentication (Au): It measures the number of times an attacker must authenticate to a target in order to exploit a vulnerability. The fewer authentication instances that are required, the higher the vulnerability score.

(4) Confidentiality Impact (CC): It measures the impact on confidentiality of a successfully exploited vulnerability. Increased confidentiality impact increases the vulnerability score.

(5) Integrity Impact (IC): It measures the impact on integrity of a successfully exploited vulnerability. Increased integrity impact increases the vulnerability score.

(6) Availability Impact (AC): It measures the impact on availability of a successfully exploited vulnerability. Increased availability impact increases the vulnerability score.

The temporal metrics in CVSS represent the time dependent features of the vulnerabilities, including exploitability in terms of their technical details, the remediation status of the vulnerability, and the availability of exploit code or techniques. The environmental metrics represent the implementation and environment specific features of the vulnerability. There are three environmental metrics as defined below, which capture the characteristics of a vulnerability that are associated with a user's IT environment.

The scoring process first calculates the base metrics according to the base equation, which delivers a score ranging from 0 to 10, and creates a vector. The vector is a text string that contains the values assigned to each metric, and it is used to communicate exactly how the score for each vulnerability is derived. Optionally, the base score can be refined by assigning values to the temporal and environmental metrics. If the temporal score is needed, the temporal equation will combine the temporal metrics with the base score to produce a temporal score ranging from 0 to 10. Similarly, if an environmental score is needed, the environmental equation will combine the environmental metrics with the temporal score to produce an environmental score ranging from 0 to 10. For the purpose of this paper, we give below the base metric equations only.

It is important to point out that the CVSS approach applies to individual vulnerability only. To the best of our knowledge, we did not find research results in literature covering the security metrics for software systems or software products, which may have multiple vulnerabilities. Our approach discussed in the following sections represents a first attempt towards a system view of vulnerability quantification at the software product level.

3. SOFTWARE SECURITY METRICS

Rigorous measurement of software security can provide substantial help in the evaluation and improvement of software products and processes. However, little agreement exists about the meaning of software security and how to define software security. We define software security metrics based on the representative weakness of the software as shown in the formulas below:

$$SM(s) = \sum_{n=1}^{m} \left(P_n \times W_n \right), \qquad (1)$$

Where *SM(s)* stands for the security metrics for the software *s*, and W_i (i = 1, 2, …, m) are the severity of those representative weakness in the software *s*. Note a software product may have many weaknesses and flaws. Here "representative" refers to those weaknesses that lead most vulnerabilities that may be exploited by attackers. Suppose the weakness corresponding to W_n has *k* vulnerabilities and their corresponding CVSS [1] base scores are $V_1, V_2, \ldots, V_k$. The severity of this weakness, W_n, is defined as the average score of them, as demonstrated in the formula (2) below.

$$W_n = \frac{\sum_{i=1}^{K} V_i}{K} \qquad (2)$$

In formula (1), each P_i (i = 1, 2, …, m) represents the risk of the corresponding weakness. We use the percentage each representative weakness occurs in the overall weakness occurrences to calculate P_i as the formula (3) below.

$$P_n = \frac{R_n}{\sum_{i=1}^{m} R_i} \qquad (3)$$

Where R_n is the frequency of occurrences for each representative weakness over a span of time in months, as illustrated in formula (4) below, where K is the number of weaknesses, and M is the number of months.

$$R_n = \frac{K}{M} \qquad (4)$$

To make the value of software security metrics *SM(s)* to range from 0 to 10, we require that the following formula (5) hold for P_n.

$$\sum_{n=1}^{m} P_n = 1 \qquad (5)$$

As shown in the formulas above, we define software security metrics based on the *representative* weaknesses of the software. For a given piece of software, we first find out those typical weaknesses reported in Common Weakness Enumeration (CWE) [15] related to the software and calculate the number of vulnerabilities caused by these weaknesses. Some weakness causes more vulnerabilities than others. We pick up those weaknesses that cause most vulnerabilities as our "representative weaknesses". After identifying the representative weaknesses for the software, we incorporate the severity of representative weaknesses into the security metrics. The severity of a vulnerability is captured by calculating the percentage of occurrences of this vulnerability compared with the total occurrences of all vulnerabilities. We use the average of CVSS [1] base scores that are from the CVE [7, 9] lists in a specific version of the software. In the equation (2), V represents the CVSS base score for the vulnerability in the CVE list. The parameter K in equation (4), however, represents the number of weakness as showed in [8].

The examples given in the following sections demonstrate how to obtain software security metrics based on their vulnerabilities.

4. SAMPLE APPLICATION

4.1 Mozilla Firefox 2

Let's first find out the top five weaknesses listed in [8] leading to most vulnerabilities as shown in the following table:

Name of representative weakness in Mozilla Firefox 2	*Number of vulnerabilities caused by the corresponding weakness*
1. Input Invalidation	13
2. Cross-site scripting (XSS)	14
3. Insufficient Information	12
4.Resource Management Error	12
5. Permission, Privilege, and Access Control	10

Then for each weakness, we find out those vulnerabilities and their CVSS base scores.

1. Input Invalidation		**2. Cross-site scripting (XSS)**	
CVE ID	CVSS BASE SCORE	CVE ID	CVSS BASE SCORE
1.cve-2008-2933	2.6	1.cve-2008-4066	4.3
2.cve-2008-2809	4.0	2.cve-2008-4065	4.3
3.cve-2008-2805	5.0	3.cve-2008-2800	4.3
4.cve-2008-2806	7.5	4.cve-2008-2808	4.3
5.cve-2008-0414	4.3	5.cve-2008-1234	4.3
6.cve-2007-5691	4.3	6.cve-2008-1243	4.3
7.cve-2007-5339	4.3	7.cve-2008-0416	4.3
8.cve-2007-4841	9.3	8.cve-2008-0415	4.3
9.cve-2007-1362	4.3	9.cve-2007-6589	4.3
10.cve- 2007-2292	4.3	10.cve- 2007-5947	4.3
11.cve-2006-6971	5.o	11.cve-2007-5947	4.3
12.cve-2006-2894	4.0	12.cve-2007-5415	4.3
13.cve-2007-5340	4.3	13.cve-2007-3670	4.3
		14.cve-2007-0995	4.3

3. Insufficient Information		**4. Resource Management Error**	
CVE ID	CVSS BASE SCORE	CVE ID	CVSS BASE SCORE
1.cve-2008-4062	10.0	1.cve-2008-2798	10.0
2.cve-2008-2806	7.5	2.cve-2008-2799	10.0
3.cve-2008-2785	9.3	3.cve-2008-2811	10.0
4.cve-2007-5959	9.3	4.cve-2008-4062	10.0
5.cve-2007-3845	9.3	5.cve-2008-2419	4.3
6.cve-2007-3734	9.3	6.cve-2008-1380	9.3
7.cve-2007-3735	9.3	7.cve-2008-1236	6.8
8.cve-2007-3737	9.3	8.cve-2008-1237	6.8
9.cve-2007-3738	9.3	9.cve-2008-0413	9.3
10.cve- 2007-0994	6.8	10.cve- 2008-0419	9.3
11.cve-2007-0775	3.7	11.cve-2007-5896	7.1
12.cve-2007-6398	6.8	12.cve-2008-0412	9.3

5. Permission, Privilege, and Access Control	
CVE ID	CVSS BASE SCORE
1.cve-2008-4060	7.5
2.cve-2008-4059	7.5
3.cve-2008-4058	7.5
4.cve-2008-3836	7.5
5.cve-2008-3835	7.5
6.cve-2008-2802	7.5
7.cve-2008-2803	6.8
8.cve-2008-2810	6.8
9.cve-2007-3285	6.8
10.cve- 2007-0802	6.4

Next, we identify the weakness, the vulnerabilities, and their frequencies of occurrences in the software.

Weakness	Amount of CVE caused (K)	Time span (mm/yyyy) (M)	Probability of vulnerability occurrence (Rn)=K/M
1. Input Invalidation	K=13	02/2007-07/2008 M=17(Months)	R1=13/17
2. Cross-site scripting (XSS)	K=14	02/2007-09/2008 M=19(Months)	R2=14/19
3. Insufficient	K=12	12/2006-09/2008	R3=12/21

Information		M=21(Months)	
4.Resource Management Error	K=12	11/2007-09/2008 M=10(Months)	R4=12/10
5. Permission, Privilege, and Access Control	K=10	02/2007-09/2008 M=19(Months)	R5=10/19

Based on these data, we could calculate the average of CVSS base scores for those vulnerabilities and generate the percentage of each weakness in the software:

Weaknesses	Severity of the weakness(The average of CVSS base scores for the vulnerabilities caused by the weakness) (Wn)	The percentage of each weakness in the software (Pn)=Rn / (R1+R2+….Rn)
1. Input Invalidation	W1=4.86	R1/(R1+R2+R3+R4+R5)=P1 P1=25935/128853
2. Cross-site scripting	W2=4.30	R2/(R1+R2+R3+R4+R5)=P2 P2=24990/128853
3. Insufficient Information	W3=8.32	R3/(R1+R2+R3+R4+R5)=P3 P3=19380/128853
4.Resource Management Error	W4=8.51	R4/(R1+R2+R3+R4+R5)=P4 P4=40698/128853
5. Permission, Privilege, and Access Control	W5=7.18	R5/(R1+R2+R3+R4+R5)=P5 P5=17850/128853

Finally, we could calculate the security metric score based the formula (1):

The final score=W1*P1+W2*P2+W3*P3+W4*P4+W5*P5 = 6.7.

4.2 Microsoft Internet Explorer 6

For Microsoft IE 6, we choose three representative weaknesses listed as the following table:

Weakness	The amount of vulnerabilities that are caused by the weakness
1. Buffer Error	8
2.Code Injection	11
3. Resource Management Error	10

For each weakness, we find out those vulnerabilities and their CVSS base scores.

1. Buffer Error		2.Code Injection	
CVE ID	CVSS BASE SCORE	CVE ID	CVSS BASE SCORE
1.cve-2008-3014	9.3	1.cve-2008-1085	9.3
2.cve-2008-3012	9.3	2.cve-2008-1086	9.3
3.cve-2007-5348	9.3	3.cve-2008-1368	4.3
4.cve-2008-1442	9.3	4.cve-2008-0076	9.3
5.cve-2007-4790	7.5	5.cve-2008-0078	9.3
6.cve-2007-3481	5.0	6.cve-2007-5456	7.5
7.cve-2007-2222	9.3	7.cve-2007-3892	7.5
8.cve-2003-1484	4.3	8.cve-2007-3550	7.8

We can get the resource management error CVSS base scores for the following CVEs: 1.cve-2008-3476: 9.3; 2.cve-2008-3475: 9.3; 3.cve-2008-3013: 9.3; 4.cve-2008-2254: 9.3; 5.cve-2008-2255: 9.3; 6.cve-2008-2257: 9.3; 7.cve-2008-2258: 9.3; 8.cve-2008-0077: 9.3; 9.cve-2008-3903: 6.8; and 10.cve-2008-3041: 9.3.

Next, we identify the weakness, the vulnerabilities, and their frequencies of occurrences in the software.

Weakness	The total amount of vulnerabilities (K)	Span of time (mm/yyyy) (M)	The probability of each weakness's occurrence (Rn)=K/M
1. Buffer Error	K=8	12/2003-09/2008 M=57(Months)	R1=8/57
2.Code Injection	K=11	12/2004-04/2008 M=40(Months)	R2=11/40
3.Resource Management Error	K=10	08/2007-10/2008 M=14(Months)	R3=10/14

Based on these data, we could calculate the average of CVSS base scores for those vulnerabilities and generate the percentage of each weakness in the software:

Weaknesses	Severity of weakness (The average of CVSS base score for the vulnerabilities (Wn)	Percentage of each weakness occurred in the software (Pn)=Rn / (R1+R2+….Rn)
1. Buffer Error	W1=7.91	R1/(R1+R2+R3)=P1 P1=2240/18029
2.Code Injection	W2=8.22	R2/(R1+R2+R3)=P2 P2=4389/18029
3.Resource Management Error	W3=9.05	R3/(R1+R2+R3)=P3 P3=11400/18029

Finally, we could calculate the security metric score based the formula (1):

The final score=W1*P1+W2*P2+W3*P3 = 8.7.

4.3 Microsoft Internet Explorer 7

For Microsoft IE 7, we choose three representative weaknesses listed as the following table:

Weakness	The amount of vulnerabilities that are caused by the weakness
1. Input Invalidation	6
2.Code Injection	8
3. Resource Management Error	12

For each weakness, we find out those vulnerabilities and their CVSS base scores.

1. Input Invalidation		2.Code Injection	
CVE ID	CVSS BASE SCORE	CVE ID	CVSS BASE SCORE
1.cve-2008-2256	9.3	1.cve-2008-1085	9.3
2.cve-2008-2259	9.3	2.cve-2008-0076	9.3
3.cve-2008-4071	5.0	3.cve-2008-0078	9.3
4.cve-2008-1544	5.8	4.cve-2007-5344	6.8
5.cve-2008-1545	4.3	5.cve-2007-5456	7.5
6.cve-2007-3896	9.3	6.cve-2007-3892	7.5
		7.cve-2007-3550	7.8
		8.cve-2007-1751	9.3

Similarly, we can get the resource management error CVSS base scores for the following CVEs: 1.cve-2008-4381: 5.0; 2.cve-2008-4127: 4.3; 3.cve-2008-3902: 9.3; 4.cve-2008-2254: 9.3; 5.cve-2008-2255: 9.3; 6.cve-2008-2257: 9.3; 7.cve-2008-2258: 9.3; 8.cve-2008-0077: 9.3; 9.cve-2008-3903: 6.8; 10.cve-2008-5347: 6.8; 11.cve-2007-3893: 6.8; and 12.cve-2007-3041: 9.3.

Next, we identify the weakness, the vulnerabilities, and their frequencies of occurrences in the software.

Weakness	The total amount of vulnerabilities (K)	Span of time (mm/yyyy) (M)	The probability of each weakness's occurrence (Rn)=K/M
1. Input Invalidation	K=6	10/2007-09/2008 M=11(Months)	R1=6/11
2.Code Injection	K=8	06/2007-04/2008 M=10(Months)	R2=8/10
3.Resource Management Error	K=12	08/2007-10/2008 M=14(Months)	R3=12/14

Based on these data, we could calculate the average of CVSS base scores for those vulnerabilities and generate the percentage of each weakness in the software:

Weaknesses	Severity of weakness (The average of CVSS base score for the vulnerabilities (Wn)	Percentage of each weakness occurred in the software (Pn)=Rn / (R1+R2+….Rn)
1. Input Invalidation	W1=7.17	R1/(R1+R2+R3)=P1 P1=210/848
2.Code Injection	W2=8.35	R2/(R1+R2+R3)=P2 P2=308/848
3.Resource Management Error	W3=7.90	R3/(R1+R2+R3)=P3 P3=330/848

Finally, we could calculate the security metric score based the formula (1):

The final score=W1*P1+W2*P2+W3*P3 **=** 7.9.

With the similar approach, we could calculate the security metrics for Apache Tomcat 4: 4.0, and for Apache Tomcat 5: 4.4. The details were omitted due to space limitation.

5. CONCLUSIONS AND DISCUSSION

It is widely recognized that metrics are important to information security because metrics can be an effective tool for information security professionals to measure the security strength and levels of their systems, products, processes, and readiness to address security issues they are facing. Metrics can also help identify system vulnerabilities, providing guidance in prioritizing corrective actions, and raising the level of security awareness within the organization. With the knowledge of security metrics, an information security professional can answer typical questions like "Are we secure?" and "How secure are we?" in a formal and persuadable manner. For federal agencies, a number of existing laws, rules, and regulations cite security metrics as a requirement. These laws include the Clinger-Cohen Act, Government Performance and Results Act (GPRA), Government Paperwork Elimination Act (GPEA), and Federal Information Security Management Act (FISMA). Moreover, metrics can be used to justify and direct future security investment. Security metrics can also improve accountability to stakeholders and improve customer confidence.

However, the term "security metrics" is often ambiguous and confusing in many contexts of discussion in information security. Some guiding standards and good experiments of security metrics exist, such as FIPS 140-1/2, ITSEC, TCSEC, Common Criteria (CC) and NIST Special Publication 800-55, but they are either too broad without precise definitions, or too narrow to be generalized to cover a great variety of security situations. Metrics are quantifiable measurement. Security metrics are quantitative indicators for the security attributes of an information system or technology. A quantitative measurement is the assignment of numbers to the attributes of objects or processes. For information security professionals, we are interested in measuring the fundamental security attributes of information such as confidentiality, integrity, and availability.

A number of papers have been published in the area of security metrics. CVSS [1] (Common Vulnerability Scoring System) provides a tool to quantify the severity and risk of a vulnerability to an information asset in a computing environment. It was designed by NIST (National Institute of Standard and Technology) and a team of industry partners. CVSS metrics for vulnerabilities are divided into three groups: *Base metrics* measure the intrinsic and fundamental characteristics of vulnerabilities that do not change over time or in different environments. *Temporal metrics* measure those attributes of vulnerabilities that change over time but do not change among user environments. *Environmental metrics* measure those vulnerability characteristics that are relevant and unique to a particular user's environment.

Wang [16] defines software metrics as the quantitative measurements of trust indicating how well a system meets the security requirements. Let T represent the time set, a subset of real numbers. $p(t)$ is a security parameter defined at time $t \in T$. A security metric with respect to parameter p, denoted as S_p, is a real number in [0, 1] on the time interval $[t_1, t_2]$ is defined by the following formula:

$$S_p = \int_{t1}^{t2} p(t)dt \ / (t_2 - t_1) \qquad (1)$$

A system with perfect security is a system with a security metric quantity as 1. Of course, perfect security is unachievable for information systems. The key of information security practice is to reach a goal as close as possible to the perfect security. Although it may be difficult to give the precise definition of $p(t)$, the key idea of taking time into security metrics definition is critical. In this paper, we present a practical approach to define software security metrics taking into consideration of time as well. However, our time here in this paper is a coarse grain of time, namely, we count the occurrences of vulnerabilities over a time span in months.

Another contribution of this paper is to associate software weakness and vulnerabilities with security metrics. Software vulnerabilities exist due to flaws and errors in design, coding, testing, and maintenance of software. These vulnerabilities could be exploited by attackers to compromise the computing system where the software is running on. Therefore, the number of

vulnerabilities and the severity of those vulnerabilities should be important indicators for software security and trustworthiness. The examples provided in the previous section confirm to our argument. The more vulnerabilities a software product has, the lower level of trustworthiness this software product has. The more severe vulnerabilities a software product has, the less secure this software will be. In our examples provided in the previous section, we have a few sample software products and their security metrics calculated as shown in the following table, which seems to match our experience and published security advisories.

Software Product	Security Metrics
Mozilla Firefox 2	6.7
Microsoft IE 6	8.7
Microsoft IE 7	7.9
Apache Tomcat 4	4.0
Apache Tomcat 5	4.4

We strongly believe that common weakness and exposures (CWE) and common vulnerability enumerations (CVE) provide importance source for software security metrics. Our formulas discussed in Section 3, however, seems to be complicated and it deserves more effort to simplify them.

6. REFERENCES

[1] Peter Mell, Karen Scarfone, and Sasha Romanosky, A Complete Guide to the Common Vulnerability Scoring System (CVSS), Version 2.0, Forum of Incident Response and Security Teams, http://www.first.org/cvss/cvss-guide.html (July 2007).

[2] J. A. Wang, M. Xia, and F. Zhang, "Metrics for Information Security Vulnerabilities, *Journal of Applied Global Research,* Volume 1, No. 1, 2008, pp. 48-58.

[3] J.A.Wang, Fengwei Zhang and Min Xia, "Temporal Metrics for Software Vulnerabilities," in *Proceedings of CSIIRW'08*, May 12 – 14, 2008, Oak Ridge, TN, USA.

[4] J. A. Wang, "Information Security Models and Metrics", in *Proceedings of 43rd ACM Southeast Conference*, Volume 2, pp. 178 – 184. ISBN: 1-59593-059-0. March 2005, Kennesaw, GA.

[5] Elizabeth Chew et.al., Guide for Developing Performance Metrics for Information Security, *NIST Special Publication 800-80*, May 2006.

[6] National Institute of Standards and Technology, National Vulnerability Database, Common Vulnerability Scoring System Calculator, http://nvd.nist.gov/cvss.cfm?calculator (Accessed on October 20, 2008).

[7] National Institute of Standards and Technology , National Vulnerability Database, Search CVE and CCE Vulnerability Database, , http://web.nvd.nist.gov/view/vuln/search?execution=e2s1 (Accessed on October 20, 2008).

[8] The MITRE Corporation, Common Weakness Enumeration, CWE Comprehensive Dictionary(1.0.1), http://cwe.mitre.org/data/slices/2000.html (Accessed on October 20, 2008).

[9] The MITRE Corporation, Common Vulnerability and Exposures, CVE List, http://cve.mitre.org/cve/cve.html (Accessed on October 20, 2008).

[10] The MITRE Corporation, Common Attack Pattern Enumeration and Classification, CAPEC Dictionary (Release 1.1), http://capec.mitre.org/data/dictionary.html (Accessed on October 20, 2008).

[11] Michael Gegick1, Laurie Williams, Mladen Vouk, "Predictive Models for Identifying Software Components Prone to Failure During Security Attacks", Department of Computer Science, North Carolina State University, October 28th, 2008, https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/measurement/1075-BSI.pdf (Accessed by November, 2008)

[12] Chris Wysopal, Software Security Weakness Scoring, Metricon 2.0, August 7, 2007. www.**securitymetrics**.org/content/attach/**Metric**on2.0/Wysopal-**metric**on2.0-software-**weakness**-scoring.ppt (Accessed on October, 2008).

[13] Mell P. and Quinn S, "Automating Compliance Checking, Vulnerability Management, and Security Measurement," 2007 Information Assurance Workshop (IAWS) Presentation, 2007.

[14] NIST, Information Security Automation Program, Automating Vulnerability Management, Security Measurement, and Compliance, Version 1.0 Beta, revised on May 22, 2007.

[15] The MITRE Corporation, Common Weakness Enumeration, http://cwe.mitre.org/ (Accessed on October 20, 2008).

[16] J. A. Wang, "Information Security Models and Metrics", in *Proceedings of 43rd ACM Southeast Conference***,** Volume 2, pp. 178 – 184. ISBN: 1-59593-059-0. March 2005, Kennesaw, GA