SECURITY VULNERABILITY CATAGORIES IN MAJOR SOFTWARE SYSTEMS

Omar H. Alhazmi, Sung-Whan Woo, Yashwant K. Malaiya
Colorado State University
omar| woo|malaiya@cs.colostate.edu

ABSTRACT
The security vulnerabilities in software systems can be categorized by either the cause or severity. Several software vulnerabilities datasets for major operating systems and web servers are examined. The goal is to identify the attributes of each category that can potentially be exploited for enhancing security. Linking a vulnerability type to a severity level can help us prioritize testing to develop more effective testing plans. Instead of using an ad hoc security testing approach, testing can be directed to vulnerabilities with higher risk. Modeling vulnerabilities by category can be used to improve the post-release maintenance and patching processes by providing estimation for the number of vulnerabilities of individual types and their severity levels. We also show that it is possible to apply vulnerability discovery models to individual categories which can project the types of vulnerabilities to be expected in near future.

KEYWORDS: Security, Vulnerability, Model, Taxonomy.

1. Introduction

Software vulnerabilities are the major security threats to networked software systems. A *vulnerability* is defined as "a defect which enables an attacker to bypass security measures" [1]. The vulnerabilities found are disclosed by the finders using some of the common reporting mechanisms available. The databases for the vulnerabilities are maintained by organizations such as National Vulnerabilities Database [2], MITRE Corporation [3], Security Focus [4] and individual software developers. The databases also record some of the significant attributes of the vulnerabilities.

While there have been many studies attempting to identify causes of vulnerabilities and potential counter-measures, the development of systematic quantitative methods to characterize security has begun only recently. There has been an extensive debate comparing the security attributes of open source and commercial software [5]. This debate demonstrates that for a careful interpretation of the data, rigorous quantitative modeling methods are needed. The likelihood of a system being compromised depends on the probability that a newly discovered vulnerability will be exploited. Thus, the risk is better represented by the not yet discovered vulnerabilities and the vulnerabilities discovery rate rather than by the vulnerabilities that have been discovered in the past and remedied by patches. Possible approaches for a quantitative perspective of exploitation trends are discussed in [6], [7]. Probabilistic examinations of intrusions have been presented by several researchers [8][9]. The vulnerabilities discovery process in operating systems has just recently been examined by Rescorla [10] and by Alhazmi and Malaiya [11], [12], [13].

It has been estimated that 10% of vulnerabilities cause 90% of exposures [14]. The risk of some vulnerabilities is minor while it is quite significant for others. The exploitations of some of the vulnerabilities have been widely reported. The Code Red worm [15], which exploited a vulnerability in IIS (described in Microsoft Security Bulletin MS01-033, June 18, 2001), appeared on July 13, 2001, and soon spread world-wide in unpatched systems is an example of a very critical vulnerability.

The goal of this work is to assess the vulnerabilities of software systems using some of their main attributes, by two approaches, modeling and analysis. Modeling can be used to predict future vulnerabilities and their attributes. On the other hand, analysis can link categories attributes to severity levels attributes, so testers can design test cases to target those vulnerabilities that are likely to be more critical. The analysis will look into the distribution of vulnerabilities on categories and identify the mistakes that have lead to having more vulnerabilities from a specific type or severity, the distribution of vulnerabilities can lead to better diagnosis of development flaws and mistakes.

There have been some studies that have studied vulnerabilities and its discovery rate [11], [12][13]. These studies treat vulnerabilities as equally important entities, in this study we expand the prospective to look into specific vulnerabilities categories and severity.

For categorizing vulnerabilities by vulnerability type, several proposed taxonomies have aimed toward a universal taxonomy of vulnerabilities [16], [17]. These vulnerabilities taxonomies focused on goals like: Mutual exclusiveness, exhaustiveness and repeatability, those qualities will provide an unambiguous taxonomy, vulnerability taxonomy is considered an evolving area of

research; nevertheless, there is a taxonomy that has gained recognition from organizations such as MITRE Corporation [3] and the National Vulnerabilities Database (NVD) managed by National Institute of Standards and Technology(NIST) which has shown to be acceptable [2].

The other important taxonomy is to classify vulnerabilities by their severities, here also where many attempts have tried to provide objective classifications, a heuristic-based classifications looks at several attributes of the vulnerability is Common Vulnerabilities scoring system (CVSS). CVSS is being adapted by the NIST NVD it provides a standard for vulnerabilities severity classification, the system give each vulnerability a value from 1-10, the higher the number the more severe the vulnerability [24].

2. Vulnerabilities Discovery Models

Use of reliability growth models is now common in software reliability engineering [18]; SRGMs show that as bugs are found and removed, fewer bugs remain [19]. Therefore, the bug finding rate gradually drops and the cumulative number of bugs eventually approaches saturation. Such growth models are used to determine when a software system is ready to be released and what future failure rates can be expected.

Table 1: The data sets used

Systems	Lines of code (millions)	OS or Software Type	Known Vulnerabilities	Vulnerability Density (per Ksloc)	Release Date
Windows XP	40	Commercial client	173	0.0043	Oct 2001
Windows 2000	35	Commercial server	243	0.00694	Feb 2000
R H Linux 6.2	17	Open-source server	118	0.00694	Mar 2000
Fedora	81	Open-source server	143	0.0019	Nov 2003
IIS	N/A	Commercial web server	121	N/A	1995
Apache	0.227	Open-source web server	94	0.414	1995

Vulnerabilities are a special class of defects that can permit circumvention of security measures. Some vulnerabilities discovery models were recently proposed by Anderson [5], Rescorla [10], and Alhazmi and Malaiya [11]. The applicability of these models to several operating systems was examined in [20]. The results show that while some of the models fit the data for most operating systems, others do not fit well or provide a good fit only during a specific phase.

Here, we investigate the applicability a vulnerability discovery model on some operating systems and web servers vulnerabilities classified by their category or severity level, the models is a time-based proposed by Alhazmi and Malaiya [11]. This model has been found to fit datasets for several of the major Windows and Linux operating systems, and the two major web servers [21], as determined by goodness of fit and other measures.

This model, referred to as the Alhazmi-Malaiya Logistic model (AML), assumes that the rate of change of the cumulative number of vulnerabilities Ω is governed by two factors, as given in Equation 1 below [11]. The first factor declines as the number of remaining undetected vulnerabilities declines. The other factor increases with the time needed to take into account the rising share of the installed base. The saturation effect is modeled by the first factor. While it is possible to obtain a more complex model, this model provides a good fit to the data, as shown below. Let us assume that the vulnerabilities discovery rate is given by the differential equation:

$$\frac{d\Omega}{dt} = A\Omega(B-\Omega), \qquad (1)$$

where Ω is the cumulative number of vulnerabilities, t is the calendar time, and initially t=0. A and B are empirical constants determined from the recorded data. By solving the differential equation, we obtain

$$\Omega(t) = \frac{B}{BCe^{-ABt}+1}, \qquad (2)$$

where C is a constant introduced while solving Equation 1. Equation 2 gives us a three-parameter model given by the logistic function. In Equation 2, as t approaches infinity, Ω approaches B. Thus, the parameter B represents the total number of accumulated vulnerabilities that will eventually be found.

AML shows that the vulnerabilities discovery rate increases at the beginning, reaches a steady rate and then starts declining. Consequently, the cumulative number of vulnerabilities shows an increasing rate at the beginning as the system begins to attract an increasing share of the installed base. After some time, a steady rate of vulnerabilities finding yields a linear curve. Eventually, as the vulnerabilities discovery rate begins to drop, there is saturation due both to reduced attention and a smaller pool of remaining vulnerabilities.

Vulnerabilities are usually reported using calendar time, the reason for this is that it is easy to record vulnerabilities and link them to the time of discovery. This, however, does not take into consideration the changes occurring in the environment during the lifetime of the system. A major environmental factor is the number of installations, which depends on the share of the installed base of the specific system. It is much more rewarding to exploit vulnerabilities that exist in a large number of computers. Hence, it can be expected that a larger share of the effort going into the discovery of vulnerabilities, both in-house and external, would go toward a system with a larger installed base [11].

3. Vulnerabilities by Type and Severity Classification

In this section we preview the taxonomy that will be used for our analysis; we look into how vulnerabilities are classified by category and by severity level.

3.1 Modeling vulnerabilities by category

Vulnerabilities taxonomy is still an evolving area of research. Several taxonomies have been proposed [16], [17], [22], [23]; an ideal taxonomy should have such desirable properties as mutual exclusiveness, clear and unique definition, repeatability, and coverage of all software vulnerabilities. These characteristics will provide an unambiguous taxonomy; however, meanwhile, there is a taxonomy that has gained recognition from organizations such as MITRE Corporation and the NIST NVD [2] which has shown to be acceptable.

Vulnerabilities can be classified using schemes based on cause, severity, impact and source, etc. In this analysis, we use the classification scheme employed by the National Vulnerability Database of the National Institute of Standards and Technology. This classification is based on the causes of vulnerabilities. The eight classes are as follows [2], [4]:

1. Input Validation Error (IVE) (Boundary condition error (BCE), Buffer overflow (BOF)): Such types of vulnerabilities include failure to verify the incorrect input and read/write involving an invalid memory address.
2. Access Validation Error (AVE): These vulnerabilities cause failure in enforcing the correct privilege for a user.
3. Exceptional Condition Error Handling (ECHE): These vulnerabilities arise due to failures in responding to unexpected data or conditions.
4. Environmental Error (EE): These vulnerabilities are triggered by specific conditions of the computational environment.
5. Configuration Error (CE): These vulnerabilities result from improper system settings.
6. Race Condition Error (RC): These are caused by the improper serialization of the sequences of processes.
7. Design Error (DE): These are caused by improper design of the software structure.
8. Others: Includes vulnerabilities that do not belong to the types listed above, sometimes referred to as nonstandard.

Unfortunately, the eight classes are not completely mutually exclusive. A proportion of vulnerabilities can belong to more than one category. Because a vulnerability can belong to more than one category, the summation of all categories for a single software system may add up to more than the total number of vulnerabilities.

3.2 Modeling Vulnerabilities by severity

The other important taxonomy is to classify vulnerabilities by their severities, here also where many attempts have tried to provide objective classifications, a heuristic-based classifications looks at several attributes of the vulnerability is Common Vulnerabilities scoring system (CVSS).

CVSS is being adapted by the NIST NVD [2] it provides a standard for vulnerabilities severity classification, the system give each vulnerability a value from 1-10, the higher the number the more severe the vulnerability [24]. where the range 1-3.99 corresponds to low severity, 4-6.99 to medium severity and 7-10 to high severity; The National Vulnerability Database of the National Institute of Standards and Technology describes the severity levels, as follows [2]:

1. High Severity: This makes it possible for a remote attacker to violate the security protection of a system (i.e., gain some sort of user, root or application account), or permits a local attack that gains complete control of a system, or if it is important enough to have an associated CERT/CC advisory or US-CERT alert.

2. Medium Severity: This does not meet the definition of either "high" or "low" severity.

3. Low Severity: The vulnerability typically does not yield valuable information or control over a system but rather gives the attacker knowledge provides the attacker with information that may help him find and exploit other vulnerabilities or we feel that the vulnerability is inconsequential for most organizations.

4. Modeling Vulnerabilities by Category and Severity

In this section we examine the applicability of the Alhazmi-Malaiya vulnerability discovery models on the datasets for individual vulnerability type or severity level. If the model is applicable to individual categories it could be expected that estimations for future vulnerabilities of a specific category is possible, giving estimators better details about vulnerabilities estimation. Furthermore, if the model applies to severity levels it can also be possible to predict the severity levels of future vulnerabilities.

Figure 1 and Figure 2 show how the model fits the datasets for Windows XP and Windows 2000 for the major individual vulnerabilities categories; the χ^2 goodness of fit test, results in Tables 2 and 3. In this goodness of fit test, we have examined Access Validation Error (AVE), Exceptional Control Handling Error (ECHE), Buffer Overflow (BOF), and Design Error (DE) vulnerabilities. Other vulnerabilities are not statistically significant; therefore, we can not accurately test the goodness of fit for these vulnerabilities (i.e. Race Condition, Environmental Error, etc.).

Table 2. Windows XP Goodness of fit results

Vulnera-bility Type	Parameters			X^2		
	A	B	C	P-value	X^2	$X^2_{Critical}$
AVE	0.00103	63.595	2.011	1	17.70871	67.50
ECHE	0.00070	92.864	0.674	1	12.10664	
BOF	0.00080	93.578	0.348	1	14.82454	
DE	0.00285	28.984	0.195	1	13.08505	

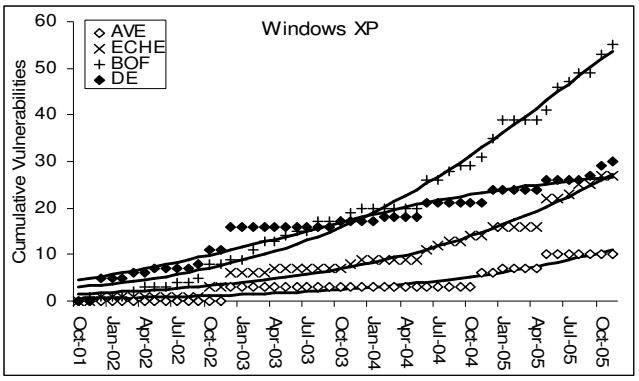

Figure 1. The plots for individual vulnerabilities categories data fitted to the models for Windows XP

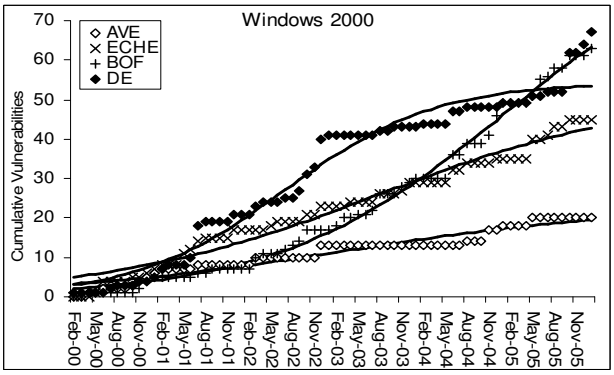

Figure 2. The plots for individual vulnerabilities categories data fitted to the models for Windows 2000

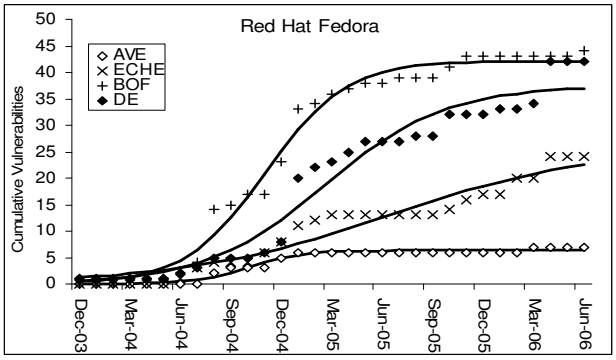

Figure 3. Plots for individual vulnerabilities categories data fitted to the models for Red Hat Fedora

Figure 3 above shows how the AML model fits individual vulnerabilities categories for Red Hat Fedora. Table 4 shows the χ^2 goodness of fit test is applied, it shows a significant fit for all vulnerabilities categories examined. Similarly here, we have only examined vulnerabilities with statistically significant data.

Table 3. Windows 2000 Goodness of fit results

Vulnerability Type	Parameters			X^2		
	A	B	C	P-value	X^2	X^2 Critical
AVE	0.0021	23.181	0.279	1	25.552	92.81
ECHE	0.0010	51.954	0.189	0.999	38.630	
BOF	0.0006	93.862	0.495	1	19.620	
DE	0.0017	54.358	0.343	0.994	45.141	

Table 4. Red Hat Fedora Goodness of fit results

Vulnera-bility Type	Parameters			X^2		
	A	B	C	P-value	X^2	X^2 Critical
AVE	0.1001	6.3065	172.35	1	3.502	44.99
ECHE	0.0064	25.720	0.927	0.905	21.25	
BOF	0.0099	42.190	3.778	0.999	12.17	
DE	0.0075	37.438	2.1773	0.988	15.86	

Figure 4 and Figure 5 show how the AML model fits data separated by severity level, the figures clearly show an s-shaped curves with different parameters values, showing that low severity vulnerabilities are discovered in a faster rate, it also opens the possibility of future prediction of vulnerabilities and their severity level. Table 5 and Table 6 show that the fit was significant at (α=.05) with a P-value>.999; with the χ^2 value significantly than the critical χ^2 in both tables.

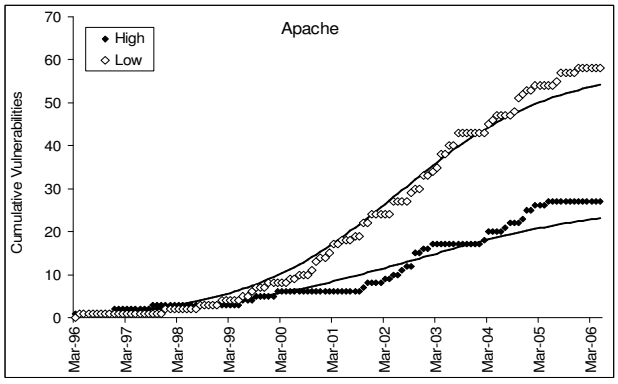

Figure 4. Apache by severity level vulnerabilities fitted to the AML

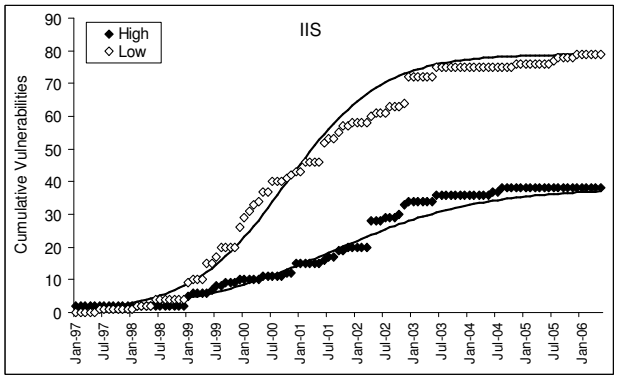

Figure 5. IIS by severity level vulnerabilities fitted to the AML

Table 5. IIS Goodness of fit results for data classified by severity

Vulnerability Type	Parameters			X^2		
	A	B	C	P-value	X^2	$X^2_{Critical}$
High	.00176	38	.999	1	28.2	133.2
Low	.00127	77.9	1.21	.999	53.5	

Table 6. Apache Goodness of fit results for data classified by severity

Vulnerability Type	Parameters			X^2		
	A	B	C	Pvalue	X^2	$X^2_{Critical}$
High	.00156	27.00	1.00	1	42.1	144.3
Low	.00248	18.00	1.76	1	15.7	

5. Vulnerabilities Type and Severity

The goal of this analysis is to determine which vulnerabilities are more severe so testers can target vulnerabilities of high severity. We can identify the most severe vulnerabilities categories, so testers can design tests targeting a specific category of vulnerabilities.

The analysis will include a diverse group of software systems which are: Windows 98, Windows XP, Windows 2000, Red Hat Fedora, Apache and IIS web Servers. The vulnerabilities data are from the National Vulnerabilities Database maintained by NIST. The market share data from Netcraft [25] was used.

The data shows that high severity vulnerabilities are half the number of vulnerabilities making them the most common vulnerabilities, a significant number of high severity vulnerabilities are buffer overflow then design error in windows 2000 and boundary condition. Boundary condition errors (IVE-BCE) has shown to be of high severity only in Windows systems (see Tables 8 and 9) while it is not so in Fedora, IIS and Apache see Tables 10-12, this possibly indicate that there is a common flow in windows regarding boundary testing.

Table 8 to Table 11 show a common observation that Exceptional Control Handling Error ECHE is associated with low severity vulnerabilities this observation is common in all examined datasets. The tables show that Race Condition, Environmental Error and Configuration Error are very few which make it hard to link it to a particular severity.

Table 10 below shows that only 35 apache vulnerabilities are of high severity, about half of them are Input validation errors with some buffer overflows and general input validation error. Design errors count for 23 vulnerabilities in total with only 5 high severity vulnerabilities.

Table 9 shows Red Hat Fedora vulnerabilities, which shows that about 41.3% of vulnerabilities, are of high severity, the majority is buffer overflow, and here we can observe similarity in both Windows systems in the dominance of Buffer overflow and other input validation error, this can be due to using C programming language, which does not have memory management features. Furthermore, a significant number of high severity vulnerabilities belong to design error which is 15 vulnerabilities. Exceptional condition handling errors are mostly low severity vulnerabilities, similar to Windows systems.

Table 7. Vulnerabilities Type vs. Severity for Windows 2000

	AVE	DE	ECHE	IVE-BOF	IVE-BCE	IVE-other	RC	EE	CE	other	Total
High	10	29	12	52	18	7	1	2	4	1	135
Medium	6	15	2	5	0	2	0	0	2	1	33
Low	6	26	32	9	3	16	1	2	3	1	100
Total	22	70	46	66	21	25	2	4	9	3	268

Table 8. Vulnerabilities Type vs. Severity for Windows XP

	AVE	DE	ECHE	IVE-BOF	IVE-BCE	IVE-other	RC	EE	CE	other	Total
High	8	11	7	47	16	5	1	1	0	2	98
Medium	0	4	2	5	0	3	0	0	0	2	16
Low	3	15	19	4	1	12	1	1	2	0	58
Total	11	30	28	56	17	20	2	2	2	4	172

Table 9. Vulnerabilities Type vs. Severity for Red Hat Fedora

	AVE	DE	ECHE	IVE-BOF	IVE-BCE	IVE-other	RC	EE	CE	other	Total
High	2	15	4	33	6	9	3	0	1	1	74
Medium	1	4	0	3	2	2	0	0	0	1	13
Low	3	30	19	9	9	17	0	2	1	2	92
Total	6	49	23	45	17	28	3	2	2	4	179

Table 11 below shows that 85 of IIS vulnerabilities are of low severity while 41 are of high severity, most high severity vulnerabilities belongs to Input Validation Error vulnerabilities and most of them is Buffer overflow, Design error vulnerabilities account for 26 vulnerabilities only ten of them highly severe, showing some errors with the design of IIS. However, IIS has matured over the years and has not shown new vulnerabilities in some time now.

Table 10. Vulnerabilities Type vs. Severity for Apache

	AVE	DE	ECHE	IVE-BOF	IVE-BCE	IVE-other	RC	EE	CE	other	Total
High	3	5	3	7	1	10	0	3	2	1	35
Medium	0	1	1	3	0	0	1	0	0	0	6
Low	3	17	12	3	1	14	1	1	12	4	68
Total	6	23	16	13	2	24	2	4	14	5	109

Table 11. Vulnerabilities Type vs. Severity for IIS

	AVE	DE	ECHE	IVE-BOF	IVE-BCE	IVE-other	RC	EE	CE	other	Total
High	3	10	2	13	1	8	0	0	3	1	41
Medium	0	1	0	1	0	0	0	0	1	0	3
Low	14	15	13	4	8	22	1	3	2	3	85
Total	17	26	15	18	9	30	1	3	6	4	129

From Table 10 and Table 11 the distributions of the severities of the Apache and IIS vulnerabilities show similarity. About 60% of total vulnerabilities have low severity, followed by about 30% with high severity, with very few vulnerabilities of medium severity.

Table 10 and Table 11 illustrate how severity level correlates with error classification. It is noticeable that Exceptional control handling error constituted the majority among low severity vulnerabilities for both Apache and IIS. In IIS, Buffer overflow vulnerabilities are associated with high severity, a relatively smaller fraction of exceptional condition errors are of high severity. In IIS as well, the exceptional condition errors tend to be from among the vulnerabilities with low severity. For IIS, most configuration errors are medium severity. Tables 7-11 show that Input Validation Errors other than Buffer overflow and Boundary Condition Error are likely to be of low severity.

7. Conclusions

In the past the security vulnerabilities have been often either studied as individual vulnerabilities or as an aggregate number. This paper examines categories of vulnerabilities. The results show that individual vulnerabilities categories and severity level complies with the AML vulnerabilities discovery models. This work can be extended by examining the prediction capabilities of the models for individual categories and severity level. This prediction capability can be included in risk assessment applications, similar to the application suggested by Sahinoglu in [26] where many factors including estimations of future vulnerabilities factors are used. For some categories and severity levels, the number of vulnerabilities of those categories are statistically insignificant and thus not suitable for statistical modeling.

The vulnerabilities categories-severities analysis was able to link Buffer Overflow to high severity vulnerabilities and Exceptional Control Handling Error is linked to low severity vulnerabilities. Input Validation errors other than Buffer overflow and Boundary Condition error are also found to be linked to low severity vulnerabilities.

Design Error vulnerabilities were found to be a significant category of which a significant proportion is usually of high severity indicating that there is a need to take past mistakes of those vulnerabilities in consideration when designing new software systems, integrating security design into the lifecycle of software system became more important, as suggested by Seacord in [27].

Tracking a vulnerability profile can be a good idea to learn from past mistakes and highlight the weaknesses of the software, which can point to mistakes and practices that were responsible for the introduction of some vulnerabilities. Further research can add more vulnerabilities attributes to identify possible patterns.

References

[1] Schultz, E. E., Brown, D. S., and Longstaff, L. T. A. *Responding to computer security incidents*. Lawrence Livemore National Laboratory (July 1990).

[2] National Vulnerability Database . http://nvd.nist.gov/.

[3] MITRE Corp, Common Vulnerabilities and Exposures, http://www.cve.mitre.org/.

[4] Securityfocus, http://www.securityfocus.com/

[5] Anderson, R. Security in open versus closed systems—the dance of boltzmann, coase and moore. *In Conf. on Open Source Software: Economics, Law and Policy* (2002), 1–15.

[6] Brocklehurst, S., Littlewood, B., Olovsson T. and Jonsson, E. On measurement of operational security. *Proc. 9th Annual IEEE Conference on Computer Assurance* (1994), 257–266.

[7] Hallberg, J., Hanstad, A., and Peterson, M. A framework for system security assessment. *Proc. 2001 IEEE Symposium on Security and Privacy* (May 2001), 214–229.

[8] Browne, H. K., Arbaugh, W. A., McHugh, J., and Fithen, W. L. A trend analysis of exploitations. *In IEEE Symposium on Security and Privacy* (2001), 214–229.

[9] Madan, B. B., Goseva-Popstojanova, K., Vaidyanathan, K., and Trivedi, K. S. A method for modeling and quantifying the security attributes of intrusion tolerant systems. *Perform. Eval*. 56, 1-4 (2004), 167–186.

[10] Rescorla, E. Is finding security holes a good idea? *IEEE Security and Privacy* 03, 1 (2005), 14–19.

[11] Alhazmi, O. H., and Malaiya, Y. K. Quantitative vulnerability assessment of system software. *Proc. Annual Reliability & Maintainability Symp.* (Jan. 2005), 615–620.

[12] Alhazmi, O. H., Malaiya, Y. K., and Ray, I. Security vulnerabilities in software systems: A quantitative perspective. *Proc. Ann. IFIP WG11.3 Working Conference on Data and Information Security* (Aug. 2005), 281–294.

[13] Alhazmi, O. H., and Malaiya, Y. K. Modeling the vulnerability discovery process. *Proc. 16th Intl Symp. on Software Reliability Engineering* (Nov. 2005), 129–138.

[14] Qualys, The Laws of vulnerabilities, http://www.qualys.com/docs/laws_of_vulnerabilities.pdf.

[15] Moore, D., Shannon, C., and Claffy, K. C. Code-red: a case study on the spread and victims of an internet worm. *In Internet Measurement Workshop* (2002), pp. 273–284.

[16] Aslam, T., and Spafford E. H. *A taxonomy of security faults*. Technical report, Carnegie Mellon, 1996.

[17] Seacord, C. R. and Householder, A. D. A structured approach to classifying vulnerabilities. Technical Report CMU/SEI-2005-TN-003, Carnegie Mellon, 2005.

[18] Musa, J. *Software Reliability Engineering*. McGraw-Hill, 1999.

[19] Lyu, M. R. Handbook of Software Reliability. McGraw-Hill, 1995.

[20] Alhazmi, O. H., and Malaiya, Y. K. Prediction capability of vulnerability discovery process. *Proc. Reliability and Maintainability Symposium* (Jan. 2006).

[21] Woo Sung-Whan, Omar H. Alhazmi, and Yashwant K. Malaiya, "Assessing Vulnerabilities in Apache and IIS HTTP Servers", Proc.2nd IEEE Int. Sym. on Dependable Autonomic and Secure Computing (DASC'06), Indianapolis, USA, September 29-October 1, 2000.

[22] Bishop, M. Vulnerability analysis: An extended abstract. *Proc. Second International Symp. on Recent Advances in Intrusion Detection* (Sept. 1999), 125-136.

[23] Landwehr, C. E., Bull, A. R., McDermott, J. P., and Choi, W. S. A taxonomy of computer program security flaws. *ACM Comput. Surv.* 26, 3 (1994), 211–254.

[24] Common Vulnerabilities Scoring System, http://www.first.org/cvss.

[25] Netcraft,. http://news.netcraft.com/.

[26] Sahinoglu, M. Quantitative risk assessment for dependent vulnerabilities. *Proc. Reliability and Maintainability Symposium* (Jan. 2006), 82-85.

[27] Seacord, R. *Secure Coding in C and C++*. Addison Wisely, 2005.