MATEC Web of Conferences **31**, 16004 (2015)
DOI: 10.1051/matecconf/20153116004

Using CVSS to quantitatively analyze risks to software caused by vulnerabilities

Gao Jian-Bo [1,2,a], Zhang Bao-Wen [2], Chen Xiao-Hua [3]

[1] *Department of Electronic and Information Engineeringof Nanchang Hangkong University, Nanchang 330063,China*
[2] *Information Security Department of Shanghai Jiao Tong University, Shanghai 200240,China,*
[3] *China Information Security Certification Center, Beijing 100020,China*

Abstract. Quantitative methods for evaluating and managing software security are becoming reliable with the ever increasing vulnerability datasets. The Common Vulnerability Scoring System (CVSS) provides a way to quantitatively evaluate individual vulnerability. However it cannot be applied to evaluate software risk directly and some metrics of CVSS are hard to assess. To overcome these shortcomings, this paper presents a novel method, which combines the CVSS base score with market share and software patches, to quantitatively evaluate the software risk. It is based on CVSS and includes three indicators: Absolute Severity Value (ASV), Relative Severity Value (RSV) and Severity Value Variation Rate (SVVR). Experimental results indicate that by using these indicators, the method can quantitatively describe the risk level of software systems, and thus strengthen software security.

1 Introduction

Since quite a number of software risks are caused by vulnerabilities, it is necessary to quantitatively analyze the vulnerabilities for an enhanced evaluation of the software risk (or software security).

Depending on how we view systems, software security can be measured by several possible metrics[1,2]. Through examining the number of vulnerabilities and their discovery rates, many researchers have established models to quantitatively analyze software security[3-5]. A limitation of their approaches is that they only consider the relationship between the number of vulnerabilities and time, while in the Common Vulnerability Scoring System (CVSS) [6], the score rather than the number of vulnerabilities can better reflect the risk level of a software system.

Although CVSS defines a number of metrics that can be used to characterize a single vulnerability, it cannot be used directly to evaluate software security[7]. To overcome this, this paper uses three indicators Absolute Severity Value (ASV), Relative Severity Value (RSV) and Severity Value Variation Rate (SVVR) to measure software risk, and adopts the National Vulnerability Database (NVD) as the data source. In addition, the market share and software patches are considered in the method when evaluating the software risk. The experimental results show that the use of these three indicators proposed in this paper is useful in alleviating software risk caused by vulnerabilities and enhancing security of the systems.

[a] Corresponding author: gaojb@nchu.edu.cn

2 The metrics for measuring software risk.

We define three indicators to depict software security. The following are the equations and related explanations.

$$ASV(t) = \sum_{i=1}^{n_t} CVSS_{vul_i}(t) \tag{1}$$

where $ASV(t)$ refers to the sum of the CVSS base score for all vulnerabilities in the software at time t since the release of the software. $CVSS_{vul_i}(t)$ ($i = 1,2,...,n_t$, where n_t is the number of the vulnerabilities before time t) represents the CVSS base score of vulnerabilities in the software at time t. It is worth noting that when using ASV to analyze the software risk, software with different versions is considered to be different software.

RSV of software is showed in Eqs. (2)-(3).

$$RSV(t) = \sum_{i=1}^{n_t} (CVSS_{vul_i}(t) * Patch(t)) * MarketShare(t) \tag{2}$$

$$Patch(t) = \begin{cases} 1, \textit{if there is no patch for the vulnerability at time t} \\ 0, \textit{otherwise} \end{cases} \tag{3}$$

In Eq. (2), MarketShare(t)is the market share of the software at time t which reflects the user's acceptance of the software. In Eq. (3), Patch(t) is the official patch for the vulnerability at time t.
SVVR of software is presented in Eq. (4).

$$SVVR = (ASV(t_1) - ASV(t_2))/(t_1 - t_2) \tag{4}$$

where, t can be years, quarters, months, or weeks, depending on what granularity the vulnerability analysis needs.
Eqs. (5) - (6) are the SVAR which is used to detect the stabilization of the SVVR.

$$SVAR = \sigma / \overline{R} \tag{5}$$

$$\sigma = \sqrt{((R_1 - \overline{R})^2 + (R_2 - \overline{R})^2 + \cdots + (R_n - \overline{R})^2)/n_t} \tag{6}$$

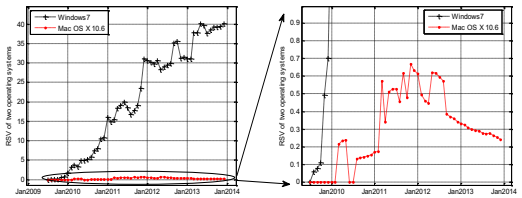
where $\overline{R}$ represents the average SVVR. R_i (i=1,2,…, n_t) is the SVVR at different time intervals.

3 Experiments and results.

The experiments use the vulnerability data reported in the National Vulnerability Database, the MITRE Corporation, and the CVE detail , covering the period 2009-2013. The vulnerabilities is divided into 3 categories according to their patches: vulnerabilities with official patches (e.g. CVE-2011-2383), vulnerabilities without official patches (e.g. CVE-2002-2435), and vulnerabilities with advisories (e.g. CVE-2010-0555). The work focuses on the analysis of software risk caused by vulnerabilities without timely official patches. Therefore, the experiments do not consider vulnerabilities with 0-day patches in the calculation of RSV of software, because the risk caused by vulnerabilities with 0-day patches is greatly reduced. It is worth noting since advisories provide a way to mitigate influences caused by vulnerabilities, they are equally treated with the official patches in the experiments.

Table 1 SVVR of Windows 7 and Mac OS X 10.8 from June 2013 to December 2013

Software	2013 Jun	2013 Jul	2013 Aug	2013 Sep	2013 Oct	2013 Nov	2013 Dec	Average $SVVR_A$	SVAR
Windows 7	20.6	66.4	39.4	62.3	57.4	35.6	36.2	45.41	0.3434
Mac OS X 10.8	34.4	0	0	53.1	142.4	0	2.1	33.14	1.469



(a) (b)

Fig.1 (a) RSV of Windows 7 and Mac OS X 10.6, (b) partial enlarged view.

This section compares the risks associated with two popular desktop operating systems: Windows 7 and Mac OS X 10.6 (Mac OS X 10.8). Table 1 shows the average SVVR of Windows 7 and Mac OS X 10.8 from June 2013 to December 2013, and the SVARs are listed in the last column of the table. The average SVVR of Windows 7 and Mac OS X 10.8 are 45.41 and 33.14, respectively. From Table 4, the average SVVR of Windows 7 is higher than that of Mac OS X 10.8, and the SVAR of the former is lower than that of the latter, which means the risks associated with Windows 7 caused by vulnerabilities change faster than that of Mac OS X10.8, and the change rate of the former is more stable than that of the latter. These may be explained by the fact that Windows 7 was under continued and sustained attacks during a certain period of time, while Mac OS X 10.8 was under short term attacks during the same period.
Fig. 1 shows the RSV of both Windows 7 and Mac OS X 10.6 from July 2009 to December 2013. From Fig. 1 (a), it can be observed that the RSV of Windows 7 is much higher than that of Mac OS X 10.6. The reasons for this include Windows 7 having more vulnerabilities without timely patches than the Mac OS 10.6 after its release and the market share of Windows 7 is much greater than the MAC OS 10.6. From Fig. 1 (b) is the partial enlarged view of Fig. 1 (a), the RSVs of Mac OS 10.6 increase first, then peak, and then decrease with time.

4 Discussions

A quantitative analysis of vulnerabilities makes it possible for software users to assess the risks in software systems more clearly; the more risk caused by vulnerabilities in a software product, the less secure the software is. Because the CVSS is only suitable for analyzing single vulnerability[6], while software systems always contain multiple vulnerabilities.
The proposed method extends usability of the CVSS by combing the CVSS base score with market share and software patches. The reason for considering market share and software patches is that they are the main factors in deciding the software risk, since a higher market share draws more attention from attackers (red or black), which leads to more newly discovered vulnerabilities; and software patches form the major mitigation strategies for vulnerabilities. Furthermore, the method uses the CVSS base score instead of the number of vulnerabilities to describe the severity of software risk, which increases the accuracy of evaluation of software

risk. The experimental results verify that the use of one indicator or a combination of indicators gives an accurate evaluation of software risk caused by vulnerabilities. Based on the proposed method, the following suggestions can be made on how to reduce software risk: A) to strengthen the protection of software with high ASV and RSV; B) if the software has a high SVVR and SVAR for some time, which means the software is currently suffering from frequent attacks, and then it requires more immediate attention; C) to increase the 0-day patch rate for software with a high market share.

5 Conclusions

In this paper, a new CVSS-based metric with three indicators (ASV, RSV, SVVR), is presented to evaluate software risk. The indicator ASV is the total of the CVSS base score since the release of the software. RSV is based on ASV, and it considers the market share and software patches. SVVR is a variation of the CVSS base score at a time interval. Experimental results show that the use of one indicator or a combination of indicators included in the metrics can accurately evaluate software risk caused by vulnerabilities. By calculating the three indicators of the representative software systems such as Windows 7 and Mac OS X 10.8 (10.6), we can obtain the growth rate of software risk and compare the risk of different software, thus providing practical security policies to strengthen the software security. Future work will focus on the refinement of the metric, making it more suitable for software risk assessment. It is necessary to analyze more software in order to test the effectiveness of the metrics. Visualization of results and automatic risk assessment tools are also necessary in the future.

6 Acknowledgments

This research is supported in part by Specialized Research Fund for the Doctoral Program of Higher Education: “Research on security mechanism of internet of things based on wireless identification and wireless sensor” (No. 13Z102090101); Shanghai Key Scientific and Technological Project (No. 11511504302).

References

1. Manadhata, P.K., Wing, J.M. An attack surface metric. IEEE Transactions on Software Engineering. 37 (3) (2011), p.371-386.
2. Frei, S., 2009. Security Econometrics: The Dynamics of (In) Security. PhD thesis, ETH Zurich.
3. Rescorla, E., 2004. Is Finding Security Holes a Good Idea? In: Proceedings Third Annual Workshop on Economics and Information Security (WEIS04), pp. 1-18.
4. Anderson, R. Security in Open versus Closed Systems – The Dance of Boltzmann, Coase and Moore. Conf. on Open Source Software: Economics, Law and Policy, 2002, p. 1-15.
5. Alhazmi, O.H., Malaiya, Y.K., Ray, I. Measuring, analyzing and predicting security vulnerabilities in software systems. Computer & Security 26 (3) (2007), p. 219-228.
6. Mell, P., Scarfone, K., Romanosky, S. Cvss: A complete guide to the common vulnerability scoring system version 2.0. 2007
7. Wang, J.A., Wang, H., Guo, M., Xia, M. Security Metrics for Software Systems. In the Proc. of ACMSE '09, 2009, p. 19-21.