Building the Security Function Point Method for Web Application Vulnerability Remediation

Kwansoon Park, Boyoung Kim

Abstract: The web application vulnerability remediation activities are important in terms of actual risk management in corporate security activities. However, traditional software development resource estimation methods do not discuss resource estimation for software vulnerability remediation in terms of security. Moreover, it is difficult to estimate the exact web vulnerability remediation resources using correction factors. In these backgrounds this study aims to establish a resource estimation methodology for web application vulnerability remediation in terms of security from the perspective of dynamic analysis, contributing to foundation building for the systematic management of web application vulnerability remediation among information security organizations and related practitioners. For the new model development, this study used 64 application data of the experimental company to derive the security function point method and 6 web vulnerability assessment project data from the same company to verify the methodology.Hence a web application vulnerability remediation standard was established, and a new security web vulnerability remediation resource estimation technique, "Security Function Point Method (SFPM)," was proposed through data collection based on the standard.It covers the de facto global web application vulnerability framework named OWASP Top 10(2017) and several Korea's standards fromthe practical field. Thus, it is possible tocalculate the web application vulnerability remediation resourcesin a better way.

Keywords :security function point method, vulnerability remediation,vulnerability management, dynamic analysis software test, penetration test, SSDLC

INTRODUCTION

Web application vulnerability remediation is divided into two types: the identification of the static analysis result that detects potential weaknesses in the source code, and the recognition of the dynamic analysis result where actual input values are verified in the external hacker's view [1]–[3]. Static analysis is characterized by the possibility to estimate the remediation by empirical correction factor because the software development and the additional release cycle are performed in the same case, except in the special cases. Dynamic analysis, on the other hand, can be performed at any time, regardless of development and maintenance cycles. Methodology relating to remediation resource estimation is necessary in this analysis type because remediation resource estimation has to be done separately. Organizations may

prefer dynamic analysis over static analysis in implementing web vulnerability remediation because dynamic analysis generates a relatively small number of potential attacks, while static analysis yields a large number of potential vulnerabilities [4]. Therefore, prioritizing the estimation and removal of software remediation resources for dynamic analysis is advantageous in terms of opportunity cost [5].

Major methodologies include Microsoft's security development life cycle (SDL) [2], Sammy Migues' the building security in maturity model 10 (BSIMM10) [3], OWASP's software assurance maturity model (SAMM) [1], and the national institute of standards and technology (NIST) security considerations in the system development lifecycle [6]. Michael classified the above major method as a common five-step security evolution process consisting of security requirements review, design model adaptation, code fixing and patch development, regression testing, and re-deployment [7].

Unfortunately, most SSDLCs focus only onSAST (static analysis software test) and DAST (dynamic analysis software test). Moreover, the methodology presupposes that when the derived vulnerability is delivered to the development team, the remediation will be addressed. However, the difficult part to succeed when building a VM (Vulnerability Management) process is the remediation phase. Vulnerability assessment activities are relatively easy to build, either by creating a new team or through external outsourcing, but the remediation of the generated vulnerability must be handled by the existing development team. Further, remediation projects are likely to fail in the long run unless the amount of additional work is assigned through planning to the developers with previously assigned development project tasks.

As with other projects, the first step in web application vulnerability remediation activities is the resource estimation of web application remediation [8], [9]. However, traditional software development resource estimation methods do not discuss resource estimation for software vulnerability remediation in terms of security [10], [11]. The purpose of this study then is to establish a resource estimation methodology for web application vulnerability remediation in terms of security from the perspective of dynamic analysis,

Manuscript published on November 30, 2019.
* Correspondence Author

Kwansoon Park*, Seoul Business School, Seoul School of Integrated Sciences and Technologies(aSSIST), Seoul, Korea. Email: kwansoon.kr@gmail.com

Boyoung Kim*,Seoul Business School, Seoul School of Integrated Sciences and Technologies(aSSIST), Seoul, Korea. Email: bykim2@assist.ac.kr

contributing to foundation building for the systematic management of web application vulnerability remediation among information security organizations and related practitioners.

For this purpose, a web application vulnerability remediation standard was established, and a new security web vulnerability remediation resource estimation technique, 'Security Function Point Method,' was proposed through data collection based on the standard.

WEB APPLICATION VULNERABILITY REMEDIATIONAND RESOURCE ESTIMATION METHODOLOGY

Web application vulnerability remediation can be divided into the point of completion of the initial development and the point of use after that. In the initial development, vulnerability assessments and remediation are completed before inspections and new releases. Once development is completed, there is no new function development at the point of use, and a procedure to analyze and remove the vulnerability of each function is followed. In this respect, the remediation process is more of a software maintenance type. The four types of software maintenance in ISO / IEC 14764 (software engineering-software life cycle processes-maintenance) are preventive maintenance, corrective maintenance, adaptive maintenance and perfective maintenance and it belongs to preventive maintenance and corrective maintenance [12]. However, the major difference between software maintenance categories and web application vulnerability generation by DAST is that maintenance categories generally have a process that is addressed when problems are found in use, whereas in the case of DAST, web application vulnerability is artificially caused according to the VM plan.

A typical development cost estimating study has evolved from the traditional source line of code (SLOC) concept of Benington in 1956 [13] and Boehm in 1981 [14] to estimate schedule and effort for a given software product [13] –[15]. These advanced methodologies include IFPUG, COCOMO-II, COSMIC-FFP, and Mark II NESMA, with various researches developed based on them being actively conducted.

Function point analysis (FPA) was proposed by Albrecht of IBM in 1979 [16], [17]. It is divided into internal logical file (ILF), external interface file (EIF), external input (EI), external output (EO), and external inquiry (EQ), and it is weighted according to complexity. These methodologies are aimed at estimating the resources and budget of the development project. Once the function point and weight of the application are calculated, the cost can be calculated by assigning the developer's cost to a function point that the developer can handle per day. The quantity of function point is measured differently in various environments, and research on this is being actively conducted. Also, according to Czarnacka, the processing cost per function point may vary depending on the function point size in the application. [18], [19].

A common methodology used to predict software development worldwide is Function Point, which is constantly being researched, improved, and developed. In particular, IFPUG CPM (ISO / IEC 20926:2009), the most commonly used function point measurement method (ISO / IEC 14143-1: 2007) for software development resource estimation, cannot measure the non-functional elements of quality, security, and technology. Therefore, IFPUG presented 14 items of IFPUG SNAP (software non-functional assessment process) to measure non-functional factors [10].

In order to, estimate vulnerability remediation project resources, remediation resources were evaluated,in accordance with OWASP Top 10-2017 [20] for the five most common web vulnerabilities found in this experiment following SNAP, but it could not be calculated as shown in Table 1.

Table-1: OWASP Top10(2017) estimation by SNAP

Items	**2.1 XSS**	**8.1 Important Data TRX Without Encryption**	**9.3 Information Exposure Through Server Error Message**	**6.1 Parameter Manipulation**	**1.1 SQL Injection**
OWASP Top 10	A7	A3	A6	A5	A1
Estimation availability by SNAP	4/4	6/8	0/6	0/8	4/4
	O	Partial	X	X	O

* Results of OWASP Top 10(2017) estimation.

SECURITY FUNCTION POINT METHOD(SFPM)

A. Designing of SFPM

The newly proposed SFPM offers two methods for the convenience of use. The first is the SFPM authenticity method, which calculates the weight in detail based on 1 security function point for each vulnerability and the second is the SFPM simple method, which calculates all of the vulnerabilities as 1 Security Function Point and more easily estimates the software development resources required for vulnerability remediation. For the accurate calculation of the remediation resources of each vulnerability, it is appropriate to use the SFPM authenticity method. For a quick assessment of the approximate total remediation resources, SFPM simple method can be used.

1) SFPM Authenticity Method

$$\sum_{i=1}^{n}(A_i \sum_{j=1}^{m} V_j * n_{a-z}) \qquad (1)$$

A_i = Application

V_j= SFP weight of vulnerability classification

n_{a-z} = # of application's vulnerabilities byclassification

2) SFPM Simple Method

$$V_0 \sum_{i=1}^{n} A_i * n_i \qquad (2)$$

A_i= Application

V_0= 1 SFP = 2.234h

n_i= # of application's vulnerabilities

In the case of using the SFPM simple method, the arithmetic mean time of the entire vulnerability derived from the experiment was set to 2.234h for 1SFP to minimize the deviation of the final result from the SFPM authenticity method. Accordingly, the following Table **2** shows the vulnerability SFPs for each type.

Table- 2: Average remediation time, frequency and SFP weight

Section	Vulnerability Classification	Average remediation Time	Frequency	SFP Weight
V1	1.1 SQL Injection	2.071	14	0.93
V2	1.2 Xpath Injection	*2.234	0	1.00
V3	1.3 OS Command Injection	*2.234	0	1.00
V4	2.1 Cross-Site Scripting (XSS)	2.286	84	1.02
V5	2.2 Cross-Site Request Forgery (CSRF)	4.000	1	1.79
V6	2.3 XML External Entity (XXE)	*2.234	0	1.00
V7	3.1 Information Exposure and Forgery Through Cookies	2.250	2	1.01
V8	3.2 Exposure of Data Element to Wrong Session	*2.234	0	1.00
V9	3.3 Insufficient Session Expiration	*2.234	0	1.00
V10	3.4 Concurrent Session Control	4.000	1	1.79
V11	4.1 Weak ID and Password Requirements	2.818	11	1.26
V12	4.2 Weak Password Recovery Mechanism for Forgotten Password	2.833	3	1.27
V13	4.3 Authentication Bypass	2.600	5	1.16
V14	4.4 Improper Authorization	3.469	32	1.55
V15	4.5 Improper Restriction of Excessive Authentication Attempts	2.375	4	1.06
V16	5.1 File and Directory Information Exposure	1.211	19	0.54
V17	5.2 Exposure of Unauthorized Administrator URL	2.192	13	0.98
V18	6.1 Parameter Manipulation	2.194	31	0.98
V19	6.2 URL Redirection to Untrusted Site	1.833	3	0.82
V20	6.3 Unrestricted Download of File	2.083	12	0.93
V21	7.1 Unrestricted Upload of File	3.100	15	1.39
V22	7.2 Improper Service Header	1.417	6	0.63
V23	7.3 Directory Indexing	3.000	2	1.34
V24	7.4 Automated Threats	3.077	13	1.38
V25	8.1 Important Data Transmission Without Encryption	1.756	41	0.79
V26	8.2 Important Data Exposure Without Encryption	2.350	10	1.05
V27	9.1 Information Exposure Through Header	1.194	49	0.53
V28	9.2 Information Exposure Through Inconsistency	1.833	6	0.82
V29	9.3 Information Exposure Through Server Error Message	2.455	33	1.10
V30	9.4 Exposure of Private Information (Privacy Violation)	3.250	2	1.45
V31	10.1 Web-related CVE Vulnerabilities (Open SSL, Apache, Tomcat, WebLogic, etc.)	5.667	3	2.54
Total average remediation time / Total vulnerability		2.234	415	-

B. Web Vulnerability Remediation Standard

The vulnerability remediation standard is the same as the assessment standard and was selected based on a combination of the global and Korean assessment standards used by the financial and public sectors to improve reliability and usability. It was designed based on the common categories of 2017 OWASP Top 10, vulnerability analysis evaluation standard for major information and communication infrastructure (MIST, KOREA 2014), website vulnerability diagnostic remediation guide (KISA, 2013), and standard for analyzing vulnerability of electronic financial infrastructure (Financial Security Agency, 2018) [20]–[22]. Also, 31 categories were finally derived from the classifications, with a high frequency of discovery in the actual operating environment. They are one global standard and three Korean standards, but the three Korean standards are also frameworks created by referring to the CWE(Common Weakness Enumeration) and CWE/SANS Top 25 [23] standards, and so it seems plausible for the practitioner to use the assessment standard using this study's standard. Likewise, it is possible to add and delete assessment categories according to the environment of the company and organization. The assessment categories were selected in abundance as shown in Table 3so that assessment categories would not be missed.

Table- 3: Integrated web application vulnerability classification

Vulnerability Classification		KI SA	OW ASP	MI ST	FS I
1. Injection	1.1 SQL Injection	O	A1	SI	001
	1.2 Xpath Injection	-	A1	XI	-
	1.3 OS Command Injection	O	A1	OC	014
2. Unrestricted Script execution	2.1 Cross-Site Scripting (XSS)	O	A7	XS	041
	2.2 Cross-Site Request Forgery (CSRF)	O	A7	CF	028
	2.3 XML External Entity (XXE)	-	A4	-	015
3. Cookie/ Session Management	3.1 Information Exposure and Forgery Through Cookies	O	A2	CC	013
	3.2 Exposure of Data Element to Wrong Session	-	A2	SE	012
	3.3 Insufficient Session Expiration	-	A2	SC	033
	3.4 Concurrent Session Control	O	A2	SF	004
4. User Authenticatio n/Authorizatio n Management	4.1 Weak ID and Password Requirements	O	A2	BF	006
	4.2 Weak Password Recovery Mechanism for Forgotten Password	O	-	PR	012
	4.3 Authentication Bypass	-	A2	IA	019

	4.4 Improper Authorization	-	A2	IN	003
	4.5 Improper Restriction of Excessive Authentication Attempts	-	A2	-	032
5. Broken Access control	5.1 File and Directory Information Exposure	O	A6	PL	040
	5.2 Exposure of Unauthorized Administrator URL	O	A5	AE	039
6. URL / Parameter Manipulation	6.1 Parameter Manipulation	O	A5	-	-
	6.2 URL Redirection to Untrusted Site	-	-	-	016
	6.3 Unrestricted Download of File	O	A6	FD	010
7. Security Misconfiguration	7.1 Unrestricted Upload of File	O	A6	FU	021
	7.2 Improper Service Header	-	A5	-	038
	7.3 Directory Indexing	O	A6	DI	029
	7.4 Automated Threats	O	-	AU	021
8. Improper Encryption	8.1 Important Data Transmission Without Encryption	O	A3	SN	027
	8.2 Important Data Exposure Without Encryption	O	A3	-	024
9. Information Exposure	9.1 Information Exposure Through Header	-	A3	-	-
	9.2 Information Exposure Through Inconsistency	-	-	-	-
	9.3 Information Exposure Through Server Error Message	O	A6	IL	031
	9.4 Exposure of Private Information (Privacy Violation)	-	A3	-	011
10. Web CVE	10.1 Web-related CVE Vulnerabilities (Open SSL, Apache, Tomcat, WebLogic, etc.)	-	A9	-	-

C. Web Vulnerability Data Collection Environment

The data used in this study were collected by company K for 1 year (from March 2018 to February 2019) through vulnerability assessment and remediation. Among the 54 candidate developers, short-term outsourcing development project developers were excluded, whereas 27 developers who could track vulnerability remediation activity in ITSM(IT service management) system were the objects for data collection. This study uses Jira Software's issue management system to implement and automate the web application vulnerability remediation flow as follows Fig.1.
The automated assessment tools used in the dynamic analysis were IBM Appscan and Qualys WAS. The false-positive analysis was performed by internal ethical hackers. and manual penetration tests were executed. The 415 vulnerabilities found in 64 applications, which were built from 2008 to 2018 by various 40 software provider's programmers, were sequentially delivered to developers through ITSM.When the vulnerability was delivered, it was pre-distributed to each project manager, and the vulnerability was finally assigned to avoid excessive workload considering the currently assigned work. A total of 27 developers were selected, their careers ranged from one to 25 years, and they completed a 6-hour vulnerability remediation training which was a 2-hour training taking place three times before theprogram's commencement.

The training time was excluded from the vulnerability remediation time. To prevent developers from delaying or abandoning vulnerability remediation due to their lack of vulnerability remediation knowledge, the help desk has been operated by supporting security source code experts so that developers could receive guidance whenever they want. Every

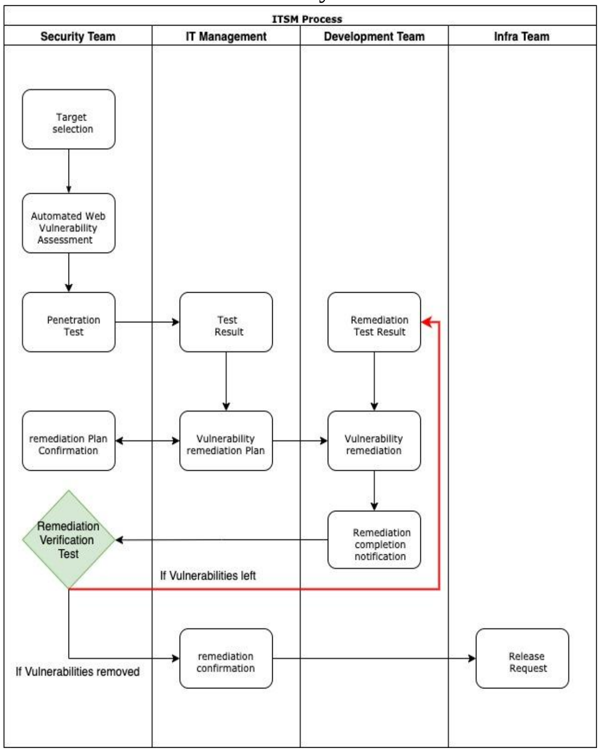
ITSM Process
Security Team
IT Management
Development Team
Infra Team
Target selection
Automated Web Vulnerability Assessment
Penetration Test
Test Result
Remediation Test Result
remediation Plan Confirmation
Vulnerability remediation Plan
Vulnerability remediation
Remediation Verification Test
Remediation completion notification
If Vulnerabilities left
If Vulnerabilities removed
remediation confirmation
Release Request

Fig 1. Automated the web application vulnerability remediation flow

two weeks, two people were assigned to address the challenges of web vulnerability remediation activities through on-site visits to development departments that were physically separated.

D. Setting Frequency and Average Remediation Time Metrics by Vulnerability Type

The concept of the new SFPM is not very different from the approach of Albrecht from IBM in 1979 [17]. Since the existing method was not effective in defining web vulnerability categories, the appropriate standards were selected, and the initial standards were calculated by measuring the actual data. This is in the same context as the approach that led to the first method of functional point analysis (FPA) [17].

Specifically, the moment of time for web vulnerability remediation is not when the information security manager assigns the vulnerability, but when the developer recognizes the vulnerability assignment from the Jira system and starts remediation. The developer removes the vulnerability to when ITSM requests the remediation verification testing. If the logged record does not fit common sense, individual reviews of the category are made directly with the developer, while misunderstandings and errors on the recording are corrected weekly.

In the middle of the development of remediation, logging is done, excluding time hindered by environmental factors and time to do another work with autonomy. In addition, researching the remediation method is a natural process of remediation because the developer does not understand the vulnerability.

Additionally, researching is included in the remediation time because it is included in the hours actually worked by the ILO. Table 2 showed the frequency and mean remediation time for each of the 415 vulnerabilities found in 64 applications. The SFP weight is the weight of remediation time in each vulnerability when the average remediation time is calculated as Table 2.

The purpose of the experiment is to derive 1 'Security Function Point (SFP)' as a reference through measured web vulnerability remediation time and to present each vulnerability weight. According to the results of the measurement, the average time taken for the remediation of each vulnerability is identified in 64 target applications. However, the V2, V3, V6, V8, and V9 items marked with * out of 415 vulnerability types of 64 applications are not found during the 1-year assessment and remediation period. The item has been replaced by 2.234, which is the arithmetic mean.

TESTING AND EVALUATION

Data from six additional web application vulnerability assessment and remediation projects are collected for SFPM validation. A total of six applications consist of two internal services and four customer services, developed in PHP, JAVA, and ASP languages, while a total of 47 web vulnerabilities are found. Six developers are assigned remediation for the validation project; however, six remediation developers to verify the experimental results are not included in the first 27 developers to calculate SFPM. The assessment results are shown in Table 4.

Table-4: Web application vulnerability assessment results from 6 web applications

No	Vulnerability Classification	Application						Total
		1	2	3	3	4	5	
1	1.1 SQL Injection	-	-	-	-	1	-	1
2	2.1 Cross-Site Scripting (XSS)	2	3	1	-	2	1	9
3	3.1 Information Exposure and Forgery Through Cookies	-	-	-	-	1	-	1
4	4.4 Improper Authorization	1	1	-	-	1	-	3
5	5.1 File and Directory Information Exposure	-	-	1	1	-	2	4
6	5.2 Exposure of Unauthorized Administrator URL	-	1	1	1	-	-	3
7	6.1 Parameter Manipulation	1	3	-	-	1	-	5
8	6.2 URL Redirection to Untrusted Site	-	-	1	-	-	-	1
9	6.3 Unrestricted Download of File	-	1	-	-	-	-	1
10	7.4 Automated Threats	-	-	-	-	-	1	1
11	8.1 Important Data Transmission Without Encryption	1	-	2	3	1	-	7
12	9.1 Information Exposure Through Header	-	-	-	1	1	1	3
13	9.2 Information Exposure Through Inconsistency	-	-	-	1	1	-	2
14	9.3 Information Exposure Through Server Error Message	1	-	-	-	1	2	4
15	9.4 Exposure of Private Information (Privacy Violation)	-	-	-	-	-	1	1
16	10.1 Web-related CVE Vulnerabilities (Open SSL, Apache, Tomcat, WebLogic, etc.)	-	-	-	-	1	-	1
Total		6	9	6	7	11	8	47

Depending on the purpose and environment, SFPM authenticity and SFPM simple methods can be used. Since the results of the two methods are not significantly different between the SFPM authenticity 46.05 SFP and the SFPM simple Method 47 SFP, there is considerable significance in their usefulness even for beginners using the simple method. However, the accuracy of the SFPM authenticity method in the vulnerability assessment is considered to be relatively high when the target is checked for a specific vulnerability and when the characteristics of the target application are focused on the specific vulnerability.

Besides, the average developer experience in the experiment is 12.26 years. This average career is calculated based on the number of 415 vulnerability allocations, not the simple average of 27 participants in the experiment. In Korea, the average annual wage of developers is announced through the Software Developer Wage Status Survey (Statistics Act: Statistical approval No. 37501) [24]. The average wage of developers over 12 years in the experimental section is 50,793 won per hour. Average wages include the base salary, sundry allowances, bonuses, retirement allowances, and corporate contributions, and do not include direct expenses and profits from other companies. Therefore, it can be expressed by the following formula [25].

$$\text{Software Development Cost} = \text{Development Cost} + \text{Direct Cost} + \text{Profit} \quad (3)$$

Profit is supposed to be estimated within 25% of development costs in the case of Korean public projects and varies depending on the environment [25]. However, the above formula can be adjusted according to the characteristics of the business and the project, and it is not absolute.

The survey on wages for SW technicians is based in 2018 and is the result of 914 companies out of a total of 1,500 software providers. In summary, the 46.05 SFP calculated by the SFPM authenticity method can be calculated as follows.
46.05 SFP to cost: 1SFP = 2.234 h / developer with an average of 12.26 years of experience.
The hourly cost of a developer with 12.26 years of experience is 50,793 won or 113,472 won based on 1SFP. Thus, 46.05 SFP x 113,472 = 5,225,365 won.

Here, if the direct costs and profits are added, 47 web vulnerability remediation costs can be derived. Finally, for the result of 47 web vulnerabilities measured (May 2019-June 2019) in 6 applications to verify SFPM predictive power, for the result of 415 web vulnerabilities (March 2018-February 2019) measured through a 1-year experiment to derive the SFPM, and to see if the results showed after the independent sample t-test for each vulnerability type by the SFP were significant, Table6 shows as follows.

Table-6: The results of the independent sample t-test for vulnerability types

#	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference
1	.	.	0	12	1	0	0.7026
4	2.367	0.127	-0.047	91	0.962	-0.0143	0.3017
7	.	.	0	2	1	0	0.5774
14	0.978	0.33	-0.254	33	0.801	-0.1979	0.7805
16	3.672	0.069	0.869	21	0.395	0.3355	0.3861
17	1.46	0.247	-0.954	14	0.356	-0.4744	0.4972
18	0.175	0.678	-1.772	34	0.085	-0.6065	0.3422
19	.	.	1	2	0.423	0.3333	0.3333
20	.	.	-0.472	11	0.646	-0.4167	0.883
24	.	.	1.411	12	0.184	1.5769	1.1178
25	0.732	0.397	-0.566	46	0.574	-0.5296	0.9361
27	3.556	0.065	0.393	50	0.696	0.1939	0.4934
28	0.222	0.654	0.732	6	0.492	0.4333	0.5919
29	4.05	0.052	0.857	35	0.397	0.8295	0.9675
30	.	.	0.577	1	0.667	0.25	0.433
31	.	.	0.574	2	0.624	1.6667	2.9059

The cases of vulnerability 1, 7, 19, 20, 24, 30, and 31 were excluded because they had only one sample, and all cases for the remaining nine types of vulnerability had p>.05 so that the non-existence of significant differences among the 415 data for expecting web vulnerability remediation time and the 47 data values used for verification can be concluded. It means measured data from 64 applications, which are built by 40 other companies, had not differenced with evaluated data from 6 applications by 6 companies.

CONCLUSIONS

Wefinally proposed a ‘Security Function Point Method’ that can be applied in business practice through experiments. This will enable a more objective approach to the challenges of estimating web vulnerability assessment resources in the past, as predicted by the intuition of business executives, information security managers, security consultants, and development professionals who want to implement web vulnerability remediation.

To use SFPM, SFPM authenticity and SFPM simple method are presented together, so even beginners can easily calculate web vulnerability resources, which can be a practical resource estimation analysis. The existing methodologies were approached from a traditional software development and maintenance perspective, so there was a limit to web application vulnerability remediation resource estimation.Moreover, the software non-functional assessment process proposed to reinforce this limitation was difficult to apply in detail from the security perspective. However, SFPM has the advantage of intuitively calculating remediation resources for detected web application vulnerability, so it is easily utilized in various organizations such as the information security organization, development organization, and operation organization.Also, this technique contains almost all ofthe major web vulnerabilities that are currently used to derive the entire major web vulnerability remediation time. Thus, this technique can be applied to the organization with minimal customization. In addition, the cost of Korea's 1 SFP can be used to infer other countries' costs.

Nevertheless, generalization may be limited because this study used 64 application data of the experimental company to derive SFPM and 6 web vulnerability assessment project data from the same company to verify the methodology. To define SFPM as a more general method, therefore, it was necessary to secure reliability by conducting data analysis and simulation for various organizations in different countries. Lastly, this study analyzed dynamic analysis mainly on web applications, but it should be extended to cover the entire software vulnerability including static analysis and secure coding.

REFERENCES

1. P. Chandra, Software Assurance Maturity Model: A guide to building security into software development Version 1.0. Open Web Application Security Project 2009.
2. M. Howard and S. Lipner, The security development lifecycle; Microsoft Press Redmond: 2006; Vol. 8.
3. J.S. Sammy Migues and M. Ware, BUILDING SECURITY IN MATURITY MODEL (BSIMM) – VERSION 10 2019.
4. A. Aggarwal and P. Jalote, Integrating Static and Dynamic Analysis for Detecting Vulnerabilities. In Proceedings of 30th Annual International Computer Software and Applications Conference (COMPSAC'06), 17-21 Sept. 2006; pp. 343-350.
5. R. Scandariato, J. Walden and W. Joosen, Static analysis versus penetration testing: A controlled experiment. In Proceedings of 2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE), 4-7 Nov. 2013; pp. 451-460.
6. R.L. Kissel, K.M. Stine, M.A. Scholl, H. Rossman, J. Fahlsing and J. Gulick, Security considerations in the system development life cycle; 2008.

7. M. Felderer and B. Katt, A process for mastering security evolution in the development lifecycle. International Journal on Software Tools for Technology Transfer 2015, 17, 245-250.
8. D. Cornell, How-to-Guide for Software SecurityVulnerability Remediation. 2010.
9. D. Cornell, Remediation statistics: what does fixing application vulnerabilities cost. Proceedings of the RSAConference, San Fransisco, CA, USA 2012.
10. Committee, IFPUG Non-Functional Sizing Standards. Software Non-functional Assessment Process (SNAP)-Assessment Practices Manual. International Function Point Users Group (IFPUG), Princeton Junction, NJ 2013.
11. V. Marthaler and Clarkston; Michigan. Function Point Counting Practices Manual International Function Point Users Group (IFPUG), Princeton Junction, NJ 2005.
12. ISO/IEC. International Standard-ISO/IEC 14764 IEEE Std 14764-2006 Software Engineering; Software Life Cycle Processes &; Maintenance. 2006.
13. H.D. Benington, Production of Large Computer Programs. Annals of the History of Computing 1983, 5, 350-361.
14. B.W. Boehm, Software Engineering Economics. IEEE Transactions on Software Engineering 1984, SE-10, 4-21.
15. J.W. Bailey and V.R. Basili, A meta-model for software development resource expenditures. In Proceedings of Proceedings of the 5th international conference on Software engineering, San Diego, California, USA; pp. 107-116.
16. C.E. Walston and C.P. Felix, A method of programming measurement and estimation. IBM Systems Journal 1977, 16, 54-73.
17. A.J. Albrecht, Measuring application development productivity. In Proceedings of Proc. Joint Share, Guide, and IBM Application Development Symposium, 1979.
18. B. Czarnacka-Chrobot, What Is the Cost of One IFPUG Method Function Point?- Case Study. In Proceedings of Proceedings of the International Conference on Software Engineering Research and Practice (SERP), 2012
19. P. Radford and R. Lawrie, The role of function points in software development contracts. White Paper, Charismatek 2000.
20. A. Van der Stock, B. Glas and T. Gigler, OWASP Top 10 2017. The Ten Most Critical Web Application Security Risks 2017.
21. Korea Internet & Security Agency. Website Vulnerability Diagnostic Remediation Guide. (2013).Accessed: October. 11, 2019. [Online]. Available:https://www.kisa.or.kr/public/laws/laws3_View.jsp?mode=view&p_No=259&b_No=259&d_No=49&ST=T&SV=
22. Korea Internet & Security Agency. Vulnerability Analysis Evaluation Standard for Major Information and Communication Infrastructure. (2018). Accessed: October. 11, 2019. [Online]. Available:https://www.kisa.or.kr/public/laws/laws3_View.jsp?cPage=6&mode=view&p_No=259&b_No=259&d_No=106&ST=T&SV=
23. CWE & SANS 25. Accessed: October. 11, 2019. [Online]. Available: https://cwe.mitre.org/top25/archive/2019/2019_cwe_top25.html
24. Korea Software Industry Association. 2018' Software Engineer Salary Research 2018. Accessed: October. 11, 2019. [Online]. Available: https://www.sw.or.kr/site/sw/ex/board/View.do?cbIdx=304&bcIdx=41033&searchExt1=
25. Korea Software Industry Association. SW project cost estimation guide. 2019. Accessed: October. 11, 2019. [Online]. Available: https://www.sw.or.kr/site/sw/ex/board/View.do?cbIdx=276&bcIdx=43381&searchExt1=

AUTHORS PROFILE

Kwansoon Park.is a Ph.D.studentat Seoul School of Integrated Sciences and Technologies(aSSIST).Heis an information security professional with more than 17 years of experience. He is a CISA, CRISC, CEH, CCFP, CWSP and certified ITIL, COBIT expert, also he has been working for the Korea government as an ISMS-PAuditor and Information Security Product Auditor.Recently He makes efforts to integrate information security tothe business area with his white-hacker team.

Bo-Young Kim is anassociate professor of Seoul School of Integrated Sciences and Technologies(aSSIST). She graduated with a Ph.D. degree in Engineering and Design School at Brunel University and holds a BA in Visual Information Design and MA of Design Management Strategy, both from the University of Ewha, South Korea. Her filed career is in business consulting as a director of IDS & Associates Consulting Company. She has worked for IPS (The Institute of Industrial Policy Study) as a senior researcher. Recent her research startup business, service innovation, consumer behavior in new communication channels.