Journal of Multimedia Information System VOL. 8, NO. 4, December 2021 (pp. 259-266): ISSN 2383-7632 (Online)
http://doi.org/10.33851/JMIS.2021.8.4.259

Empirical Risk Assessment in Major Graphical Design Software Systems

HyunChul Joh[1], JooYoung Lee[2*]

Abstract

Security vulnerabilities have been reported in major design software systems such as Adobe Photoshop and Illustrator, which are recognized as *de facto* standard design tools in most of the design industries. Companies need to evaluate and manage their risk levels posed by those vulnerabilities, so that they could mitigate the potential security bridges in advance. In general, security vulnerabilities are discovered throughout their life cycles repeatedly if software systems are continually used. Hence, in this study, we empirically analyze risk levels for the three major graphical design software systems, namely Photoshop, Illustrator and GIMP with respect to a software vulnerability discovery model. The analysis reveals that the Alhazmi-Malaiya Logistic model tends to describe the vulnerability discovery patterns significantly. This indicates that the vulnerability discovery model makes it possible to predict vulnerability discovery in advance for the software systems. Also, we found that none of the examined vulnerabilities requires even a single authentication step for successful attacks, which suggests that adding an authentication process in software systems dramatically reduce the probability of exploitations. The analysis also discloses that, for all the three software systems, the predictions with evenly distributed and daily based datasets perform better than the estimations with the datasets of vulnerability reporting dates only. The observed outcome from the analysis allows software development managers to prepare proactively for a hostile environment by deploying necessary resources before the expected time of vulnerability discovery. In addition, it can periodically remind designers who use the software systems to be aware of security risk, related to their digital work environments.

Key Words: Software Security, Vulnerability Discovery Model, Adobe Photoshop, Illustrator, GIMP.

I. INTRODUCTION

Software tools can assist designers to express their creative ideas effectively and productively in a digital manner. Software systems such as *Adobe Photoshop* is extremely popular so that most of graphic designers in related industries consider it as one of the virtues they must handle with. Those industries are usually contents driven industries, such as visual, industrial and fashion designs.

Meanwhile, a software vulnerability is a weak point in the structure which could be exploited by malicious system users causing loss or harm [1]. A publicly known software vulnerability which is not patched represents security risk and the potential risk might cause significant damages in both financially and reputably to the related parties. Also, vulnerabilities unknown to the public sometimes are traded for bad purposes [2]. Those software vulnerabilities are frequently created as a result of coding mistakes by software developers. After its creation, a software vulnerability could be discovered by any kinds of software tests, or if it's lucky, it might not be discovered forever.

If it is discovered by benevolent users, then the vulnerability will be reported to the corresponding software vendor, and we will expect a proper patch. Typically, it takes about a month or so for developing a patch from the time when a vulnerability is reported [3]. However, if malicious users find the security bugs, there could be high probabilities that the vulnerabilities will be exploited for their own benefits, mostly economic gains.

The vulnerability discovery rate could be roughly projected by vulnerability discovery models (VDM) for a given software system if it has enough datasets [4]. Malicious users also could secretly release the vulnerability information to black hat communities which will lead to proliferous exploitations. If a flawless patch is available, then the vulnerability is no more considered as a threat. Of course, users must apply the patch into their system, not to be a victim. However, sometimes a security patch itself introduces a new vulnerability.

Manuscript received November 05, 2021; Revised December 08, 2021; Accepted December 14, 2019. (ID No. JMIS-21M-11-038)
Corresponding Author (*): JooYoung Lee, School of Interdisciplinary Studies (Fashion Design Major), Kyungil University, Gyeongsan city, 38428 Korea, +82-53-600-5844, jjoolee@kiu.kr
[1]School of Computer Science, Kyungil University, Gyeongsan city, 38428 Korea, joh@kiu.kr
[2]School of Interdisciplinary Studies (Fashion Design Major), Kyungil University, Gyeongsan city, 38428 Korea, jjoolee@kiu.kr

Despite the high reputations of *Adobe Photoshop*, every now and then, vulnerabilities have been reported in its system. Those vulnerabilities are dangerous, and sometimes even fatal, since they provide privileges controlling the designers' computer systems to hackers. Those privileges could lead to leakages of unrevealed precious design outputs or sensitive personal information. As a result, considerable fears of very possible cyber-attacks are subjects to security researchers' and related industrial workers' attentions.

Overall security risk should stay under certain acceptable degrees in order not to be a target for a possible cyber-attack. To manage their security risk under control, system administrators need to somehow figure that out how much their organizations put up with potential risks. That is because if we want to improve our circumstances, somehow, we should be able to measure the problem in specific numbers.

In this paper, we are examining the three major graphical design software systems, namely *Adobe Photoshop*, *Illustrator*, and *GIMP* (GNU Image Manipulation Program) from a software security point of view in a quantitative manner. *Photoshop* and *Illustrator* are initially presented by *Adobe* in 1987 and 1990, respectively, while *GIMP* project, which is free and open-source software, was initially started as a school semester project at the University of California, Berkeley, and it was publicly available in 1996. Because it is free and open-source software, *GIMP* has been instantly adopted and pulled many kinds of contributors from software developers to design artists. The two *Adobe* systems can be installed on Windows and macOS, while *GIMP* has more options when it comes to its host operating systems. All three of them significantly have been expanded since their first releases.

Even though there have been numerous studies related to the software vulnerabilities, graphical software tools have not been one of those subjects of discussions so far. Moreover, most of the research related to the software vulnerabilities has been performed in a qualitative approach, and they have focused on detecting or preventing a specific vulnerability. Since entering the 2000s, security researchers started to examine major software products, such as operating systems, Web browsers and servers in a quantitative manner [5, 6]. That is because, in the early days of the software era, there were not enough vulnerability datasets, so it is hard to study in a quantitative manner. Here, the software vulnerability datasets for the three systems were extracted in June 2020 from the *cvedetail.com*.

In this paper, we have tried to answer the three questions following. First, whether the AML vulnerability discovery model, which was originally proposed for an operating system, can be utilized for the major graphical design tools or not. Second, how to reduce successful attacks by using the information from vulnerabilities. And third, what kind of dataset should be used for the best performance of the vulnerability discovery predictions.

The analysis here provides answers for the three questions above. First, we reveal that the AML model can describe the vulnerability discovery patterns for the graphical software systems. Second, adding an authentication process in software systems dramatically reduce the probability of exploitations. And third, predictions with evenly distributed and daily based datasets perform better than estimations with the datasets of vulnerability reporting dates only.

The remainder of the paper is organized as follows. In Section 2, some of the related studies are presented. In Section 3, vulnerability discovery processes are examined. Also, in the section, vulnerability analysis with the respect to the Common Vulnerability Scoring System is given. In Section 4, the prediction capabilities of the AML vulnerability discovery model will be investigated. Finally, Section 5 concludes this paper.

II. RELATED WORKS

An Internet security company, *Qualys*, once released a software vulnerability related paper based on empirical studies [7]. In the report, they identified the "Laws of Vulnerabilities" of four distinct and quantifiable attributes: half-life, prevalence, persistence, and exploitation. The half-life reveals a time period for reducing incidence of a vulnerability by half, and the study found that average duration of a half-life is approximately a month long, differing by industry. The prevalence measures the turnover rate of vulnerabilities in the topmost prevalent 20 security bugs. The persistence attribute measures the entire life span of vulnerabilities, and it apparently continues technically unlimited. Finally, the last attribute of exploitation shows time interval between an exploit announcement and the first attack. They found that the time gap is getting narrower than before.

The impact of feature types, classifiers, and data balancing techniques on software vulnerability prediction models are investigated by Kaya et al. [8], and they described that the performance of a vulnerability prediction model is affected by the three elements of adopted classification algorithm, adapted features, and data balancing approaches. Then, they demonstrated the effect of those factors on the performance of software vulnerability prediction models. They found that data balancing methods are effective for highly unbalanced datasets. The authors derived the conclusion that *Random*

Forest algorithm and *RusboostTree* provide the best performances in smaller and larger datasets, respectively.

Further, there could be other factors which influence on the vulnerability discovery rate. Software evolution, popularity, size and age are the most important considerations among many reasons. Also, longer lines of code increase the probability of triggering software errors. Relationship between the number of defects and the size of code could be defined with the first order approximation which let us utilize the concept of defect density. For a security vulnerability, Alhazmi and Malaiya [9] defined a comparable measurement called vulnerability density.

Meanwhile, some software vulnerability discovery models have been introduced when the datasets have become big enough to be investigated. Security vulnerabilities, which are falling into subclass of software defects, have their own certain characteristics from the non-security threaten bugs. While non-security defects are relatively patient in time for their fixing, vulnerabilities require us to act as soon as possible due to the security concerns. As a result, a state transition rate in the software lifecycle for the security defect is different from the non-security related bugs [10]. There are several S-shaped vulnerability discovery models using different statistic distributions. Joh and Malaiya [11] examined several S-shaped vulnerability discovery models to investigate any connections between skewness in each model and model performances. The result shows that Gamma and Logistic distribution-based models perform better than other models with positively and negatively skewed datasets respectively.

There are some studies debating about open-source versus proprietary software systems in various angles [12]. People prefer open-source solutions to proprietary counterparts when they need better security, free support, and ease of software development [13]. Also, adoption of open-source software is proliferating since it saves a lot of budges. However, others prefer proprietary products when they like to have a commercial support and stability. Boulanger [14] tried to answer which one is more secure and reliable in his paper, and he concluded that both are approximately equivalent in terms of reliability and security. Also, Sridhar et al. [15] showed that when it comes to the releasing patches for reported vulnerabilities, open-source software is marginally quicker than the proprietary systems.

III. VULNERABILITY DISCOVERY TRENDS AND CVSS ANALYSIS

3.1. Vulnerability Discovery Trends

From the early days of software era, there have been various software reliability growth models [16], and they were utilized for characterizing and detecting software defects in ordinary software defects. Those models can show the defect discovery patterns for a given systems in the past, and they also predict when the next bugs will be found in some degree, so that engineers could manage the optimal resource allocations related to patch management. Meanwhile, in the 1990s when many Internet services had been starting to burst out, security bugs became a considerable social concern. From the early 2000s, studies about software reliability growth models only for security related bugs have been introduced [17].

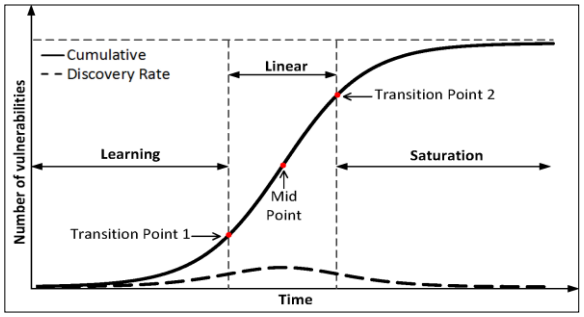
In this section, the vulnerability datasets from the three design software systems are applied into the Alhazmi-Malaiya Logistic (AML) vulnerability discovery model [5] to observe the vulnerability discovery patterns in the systems. Although AML is originally proposed for the operating systems, the model performs very well with other types of software too [6]. Fig. 1(a) represents the AML model. The S-shaped logistic distribution model can represent the observation of the popularity given to a software system. In the figure, the dashed line is represented by Equation (1) which describes the vulnerability discovery rate whereas the solid S-shaped line defined by Equation (2) represents the cumulative number of vulnerabilities along with the timeline.

$$\omega(t) = A\Omega(B - \Omega), \tag{1}$$

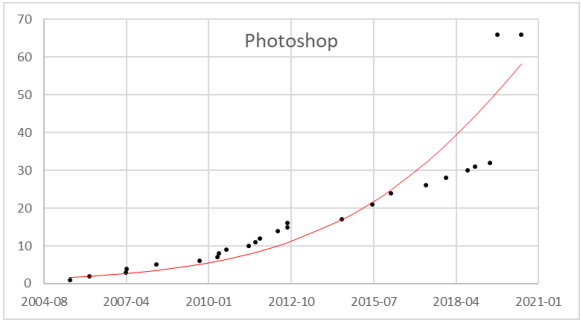
$$\Omega(t) = \frac{B}{BC^{-ABt} + 1}. \tag{2}$$

In Equation (1), the growing number of vulnerabilities is governed by the two components, A and B. The first factor A increases when time goes because of the rising share of software installations. The second factor B represents the number of remaining unknown vulnerabilities which decreases with the timeline. Equation (2) is derived from Equation (1) by the differential equation, and here $\Omega(t)$ indicates the number of cumulative vulnerabilities found at time t. The two parameters could be obtained empirically while parameter C is introduced from solving Equation (1).

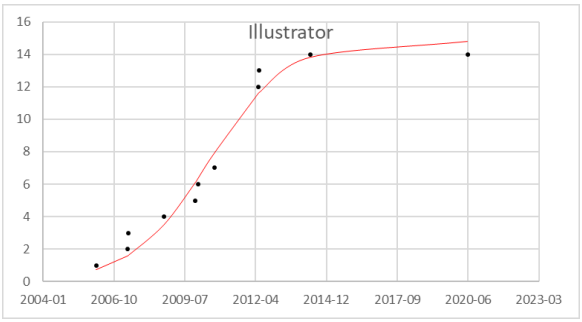
In AML model, there are three phases of learning, linear and saturation as shown in Fig. 1(a). How to calculate for the two Transition Points and the Mid-point from the figure is mathematically well defined in [18]. After a software system created, vulnerability detection rate is going up because of the gaining market share. This period is called a learning phase. Then the linear phase comes where the discovery rate reaches to the height level. The last one is saturation phase where the vulnerability discovery rate goes down because of the losing popularity.



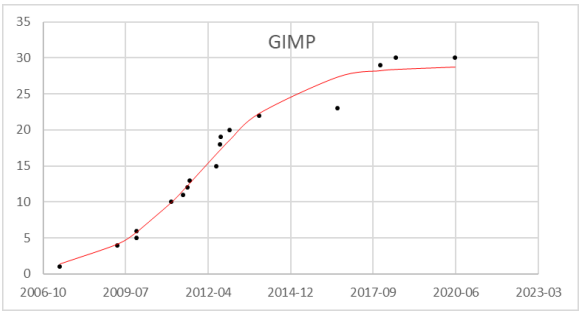
(a) AML vulnerability discovery model.



(b) Vulnerability discovery trend in *Photoshop.*



(c) Vulnerability discovery trend in *Illustrator.*



(d) Vulnerability discovery trend in *GIMP.*

Fig. 1. AML vulnerability discovery model and the model fittings.

Table 1. AML model fitting parameters from Fig. 1(b), 1(c), and 1(d)

Tools	A	B	C	R^2	T1	MP	T2
Photoshop	0.0000054	140	0.63	0.8569	2016-11-25	2021-09-02	2026-06-10
Illustrator	0.0001242	14.8077	1.2641	0.9730	2008-06-28	2010-06-14	2012-05-30
GIMP	0.0000607	28.9136	0.7078	0.9754	2009-12-24	2012-01-14	2014-02-02

Fig. 1(b), 1(c) and 1(d) show the AML model fittings for the three systems. The x-axis is the calendar time while the y-axis represents the cumulative number of vulnerabilities. The dots and solid lines on the figures represent real data points and AML model fitting lines, respectively. Table 1 shows the model fitting parameters from the three figures. The table also contains the two transition points (T1, T2) and the Mid-point (MP) with R^2 values which tells us the Pearson product moment correlation coefficient for the model fittings. The Pearson correlation is considered as one of the most common method to use for numerical variables and its output is ranges between -1 to 1, where -1 is total negative correlation, 0 means no correlation, and 1 is total positive correlation. The R^2 values higher than 0.7 indicate significant correlations between the real datasets and model generated lines [19].

The transition points shown in Table 1 indicate that vulnerability discovery trends in *Illustrator* and *GIMP* are currently in the saturation phase. It signifies that the two software systems are in stable state based on the vulnerability discovery rate. Meanwhile, the *Photoshop* is currently in the linear phase, and its peak discovery rate appears in September 2021, according to the model fitting. However, we need to keep in mind that the AML model does not consider a software evolution explicitly which introduces new chunk of source codes into systems. As a result, the vulnerability discovery rate presented in the figures might be changed in the future as the software systems are updated.

3.2. CVSS Analysis

The Common Vulnerability Scoring System (CVSS, https://www.first.org/cvss) is used as unifying and standardizing software vulnerabilities in various security bulletins in many fields in both academia and industries [20]. The scoring system has three metric groups of base, temporal, and environmental metrics. Scores in each metric group ranges from 0.0 (no risk) to 10.0 (critical). The base metric is required for the final CVSS score while the others are optional. The majority of the publicly available CVSS scores reflect the base metric information only.

The base metric is based on the exploitability and impact sub-scores. The exploitability sub-score measures how vulnerable to exploitation for a given vulnerability, and it contains attributes of Access Vector (AV), Access Complexity (AC), the number of required Authentications (Au), etc. AV reflects how a vulnerability is abused in terms of Network (N), Adjacent network (A) or Local (L). AC

Table 2. CVSS in Photoshop.

Exploitability sub-score		
AV	**AC**	**Au**
A:0	H:0	M:0
L:3	M:34	S:0
N:63	L:32	N:66
Impact sub-score		
C	**I**	**A**
C: 41	C: 41	C: 41
P: 25	P: 19	P: 19
N: 0	N: 6	N: 6

Table 3. CVSS in Illustrator.

Exploitability sub-score		
AV	**AC**	**Au**
A:0	H:0	M:0
L:1	M:5	S:0
N:13	L:9	N:14
Impact sub-score		
C	**I**	**A**
C:13	C:13	C:13
P: 1	P: 1	P: 1
N: 0	N: 0	N: 0

Table 4. CVSS in GIMP.

Exploitability sub-score		
AV	**AC**	**Au**
A: 0	H: 1	M: 0
L: 0	M: 24	S: 0
N: 30	L: 5	N: 30
Impact sub-score		
C	**I**	**A**
C: 5	C: 5	C: 5
P: 23	P: 23	P: 24
N: 2	N: 2	N: 1

measures how complex it is for an exploitation after attackers have gained access privileges to victim systems in terms of High (H), Medium (M) or Low (L). Au metric, which counts the number of times that an attacker needs to authenticate for a successful attack, has three values of Multiple (M), Single (S) and None (N). Meanwhile, the impact sub-scores measure how much a system is compromised by attackers, after a successful exploitation, with respect to the confidentiality(C), integrity(I), and availability(A). They all are measured in terms of Complete (C), Partial (P), or None(N).

Table 2, 3 and 4 represent the number of incidents in the exploitability and the impact sub-scores from the CVSS base metric group. In the tables, the shaded cells indicate that they have the highest counts in each attribute. It is observed that the network access has the most incidents in AV while the other two values of adjacent network and Local access have few or none. For AC, there is no High incident for the two *Adobe* products and *GIMP* have only one case. The majority of the incidents are Medium or Low. It suggests that complicated systems have a lot less risks to be compromised. For Au metric, there is only None values available. This signifies that if the software systems have at least one authentication process, it should be a lot safer than no authentication at all.

For the impact sub-scores, C appears the most in the two *Adobe* systems while P occupies most of the time in *GIMP* for all three categories of confidentiality, integrity and availability. This could be translated that when the systems are compromised, the open-source software has less impact from adverse events. It is possible that the *Adobe* systems share source codes, which means they share potential vulnerabilities too. It could be a future research work to uncover how sharing code effects on the vulnerability discovery trends in the systems.

IV. PREDICTION CAPABILITIES OF AML

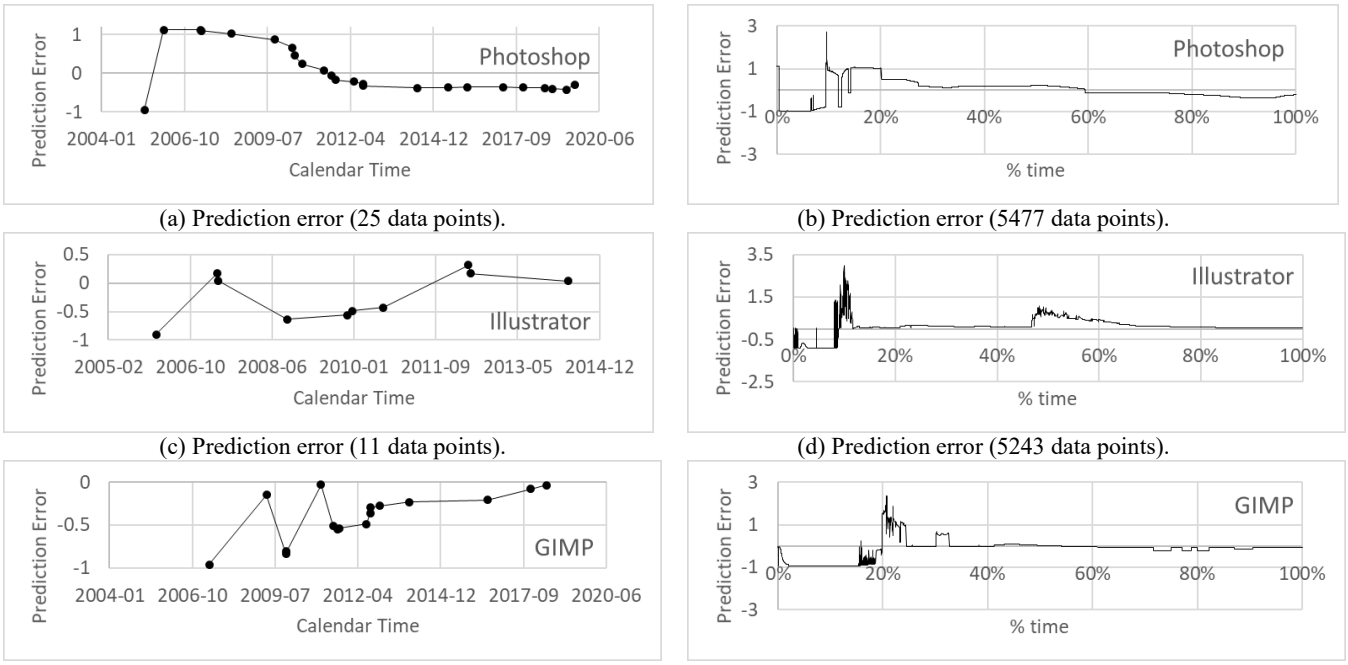
The goodness of fit tests, shown in the previous section, can provide the past vulnerability discovery trends. However, when the future trends are not consistent with models, then their performance should not be good enough for predictions. A key mission of vulnerability discovery models is estimating the number of vulnerabilities which might be encountered by a software user. This is the main issue when it comes to the estimating software stability after a particular period of test time.

If a specific vulnerability discovery model has a better estimation power than others, it could predict the number of vulnerabilities more precisely using only the data currently available. That kind of prediction abilities are necessary to evaluate the resources required for risk estimation and maintenance in advance. In the software engineering field, some of reliability growth models had been examined for their prediction capabilities [21]. When we can predict the future trend accurately, it is possible to allocate the related resources optimally in a timely manner.

The six figures from Fig. 2(a) to 2(f) show the prediction capabilities. In Fig. 2(a), 2(c) and 2(e), data points on the x-axes represent calendar time while the y-axes show prediction errors in percent. In each figure, the x-axis ranges from the date when the first vulnerability reported to the date of 2020.06.10 when the vulnerability datasets were minded. Meanwhile in Fig. 2(b), 2(d) and 2(f), the x-axes represent the percentage-time covering the same periods with the previous corresponding figures of Fig. 2(a), 2(c) and 2(e) respectively. Also, the y-axes in Fig. 2(b), 2(d) and 2(f) have the same significance with the previous figures. The prediction error *PE* can be calculated with Equation (3), where Ω is the number of actual number of vulnerabilities and Ω_t is the estimated number of vulnerabilities by AML at time *t*.

$$PE(t) = \frac{\Omega_t - \Omega}{\Omega}. \tag{3}$$

At each time point *t*, the vulnerability discovery model is initiated, and elapsed from the start point to the end, the actual datasets are fitted by the regression analysis. Then, parameters from the generated AML model are applied to estimate the number of cumulative vulnerabilities. In the entire figures, at the beginning, the prediction errors



(a) Prediction error (25 data points).

(b) Prediction error (5477 data points).

(c) Prediction error (11 data points).

(d) Prediction error (5243 data points).

(e) Prediction error (17 data points).

(f) Prediction error (4791 data points).

Fig. 2. Comparing Prediction Errors between Group1 (a, c, e) and Group2 (b, d, f).

fluctuate, then gradually converge to smaller error rates. That is because there are not enough datasets for the model regressions in the early prediction stages.

To find out how the number of time points on the x-axis affects to the model fittings, we categorize the model predictions into two groups. The first group (group 1) is composed of Fig. 2(a), 2(c) and 2(e) while the second group (group 2) is composed of Fig. 2(b), 2(d) and 2(f). Data points in the first group are unequally distributed and the number of data points are relatively small sizes because, in those figures, prediction errors are only calculated on the vulnerability reporting dates. On the other hand, in the second group, the data points in the x-axis are daily basis, so that the data points are not only equally distributed but also big sizes. The figures in the second group cover the same time periods with the first group.

Although it is true that the future projections are not meaningful when there are not enough datasets for regression analysis, a prediction result could be enhanced depending on different models, and perhaps even by using the same model, we might expect different results if we utilize the model differently.

In this paper, we calculate the model prediction performances from the two groups. It took about five minutes to calculate the prediction errors in the first group for each software system whereas the calculations from the second group took more than half a day for each software system. The calculations were performed on the desktop computer one at a time with the CPU of Core i5-4690 @ 3.50GHz, and main memory of 8.0 GB.

Average error (*AE*) and average bias (*AB*) are evaluated as presented in Equation (4) and (5), to compare the prediction capabilities between the two groups more objectively. In the equations, Ω is the number of actual vulnerabilities while Ω_t is the projected number of vulnerabilities at time *t*. *n* represents the total number of time points. *AE* measures how a model estimates an actual number precisely throughout the test phase while *AB* assesses model tendency of under or over estimations, and its values could be positive (over estimation) or negative (under estimation). They can be considered as normalized predictability measures [18].

$$AE(t) = \sum_{t=1}^{n} \left| \frac{\Omega_t - \Omega}{\Omega} \right|, \tag{4}$$

$$AB(t) = \sum_{t=1}^{n} \frac{\Omega_t - \Omega}{\Omega}. \tag{5}$$

Fig. 3 shows the comparisons between the two groups in average error and average bias. A quick glance tells us that the values from the second group are smaller than their counterpart values for both measurements, which indicates that the estimations based on the daily predictions are more accurate than the estimations based on the unequally distributed datasets. Table 5 shows how much the predictions from the second group perform better than the first group. For all the predictions in both average error and bias, it clearly tells us that the second group outperforms to the first.

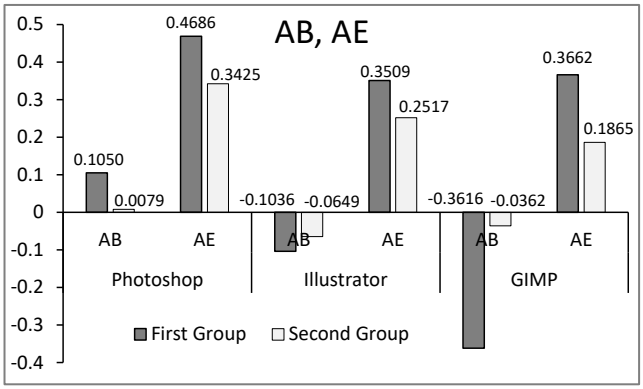


Fig. 3. Average Error and Average Bias.

Table 5. Relative Error Ratios.

	Photoshop	***Illustrator***	***GIMP***
AE2/AE1	0.7309	0.7173	0.5094
AB2/AB1	0.0756	0.6268	0.1000

AE1: AE from group1, AE2: AE from group2
AB1: AB from group1, AB2: AB from group2

V. CONCLUSION

In this paper, we quantitatively examine the security concerns in the three major graphical software systems, namely, *Photoshop*, *Illustrator* and *GIMP*. We practice the model fittings on the systems to examine if the vulnerability discovery processes on those systems are nicely represented by the AML model. Although the AML model was originally proposed for operating systems, it also can be applicable to the major design software tools too. According to the model fittings, we are carefully expecting that *Illustrator* and *GIMP* have entered their stable period while *Photoshop* is continually expecting new vulnerabilities for a while.

Next, the type of vulnerabilities in the systems were analyzed based on the CVSS. Here, we found that even a single authentication process could prevent a lot of penetrations. After a security bridge is compromised, the two *Adobe* systems bring more damages than the open-source software system according to the impact sub-scores examined. Previously, similar research about the CVSS analysis was performed by other researchers [20].

Finally, we conducted prediction capabilities with two different approaches. For the first method, estimations were only calculated on the dates only when a new vulnerability was reported while in the second method, predictions were made daily from the beginning to the end. Although the two groups have the same information of the entire and actual vulnerability discovery dates, for the model prediction, the second method performs better than the first one.

The insight from this paper can be used by software development managers to allocate software developers optimally for security patches. We could also expect that normal graphical software users start to be aware of security risk, related to their digital work environments.

Acknowledgement

This research was supported by the intramural research program in Kyungil University.

REFERENCES

[1] C.P. Pfleeger and S. L. Pfleeger, *Security in Computing*, 3rd ed., Prentice Hall PTR, 2003.

[2] L. Allodi, "Economic Factors of Vulnerability Trade and Exploitation," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, TX, USA, pp. 1483-1499, 2017.

[3] K. Scarfone and P. Mell, "An Analysis of CVSS Version 2 Vulnerability Scoring," in *Proceedings of the 2009 International Symposium on Empirical Software Engineering and Measurement*, pp. 516-525, 2009.

[4] F.K. Wai, L.W. Yong, D.M. Divakaran, and V.L.L. Thing, "Predicting vulnerability discovery rate using past versions of a software," in *Proceedings of the 2018 IEEE International Conference on Service Operations and Logistics, and Informatics*, pp. 220-225, 2018.

[5] O.H. Alhazmi and Y.K. Malaiya, "Application of Vulnerability Discovery Models to Major Operating Systems," *IEEE Transactions on Reliability*, vol. 57, no. 1, pp. 14-22, 2008.

[6] H. Joh, "Assessing Web Browser Security Vulnerabilities with respect to CVSS," *Journal of Korea Multimedia Society*, vol. 18, no. 2, pp. 199-206, 2015.

[7] Qualys Inc., "The Laws of Vulnerabilities 2.0" *in Black hat 2009*, 28 July 2009; https://www.qualys.com/docs/laws-of-vulnerabilities-2.0.pdf

[8] A. Kaya, A.S. Keceli, C. Catal, and B. Tekinerdogan, "The impact of feature types, classifiers, and data balancing techniques on software vulnerability prediction models," *Journal of Software Evolution and Process*, vol. 31, no. 9, 2019.

[9] O.H. Alhazmi, Y.K. Malaiya, and I. Ray, "Security Vulnerabilities in Software Systems: A Quantitative Perspective," in *Proceedings of the Working Conference on Data and Information Security*, pp. 281-294, 2005.

[10] H. Okamura, M. Tokuzane, and T. Dohi, "Quantitative Security Evaluation for Software System from Vulnerability Database," *Journal of Software*

Engineering and Applications, vol. 6, no. 4A, pp. 15-23, 2013.

[11] H. Joh and Y. K. Malaiya, "Modeling Skewness in Vulnerability Discovery," *Quality and Reliability Engineering International*, vol. 30, no. 8, pp. 1445-1459, 2014.

[12] A. Singh, R. K. Bansal, and N. Jha, "Open Source Software vs Proprietary Software," *International Journal of Computer Applications*, vol. 114 no. 18, pp. 26-31, 2015.

[13] S. Dhir and S. Dhir, "Adoption of open-source software versus proprietary software: An exploratory study," *Strategic Change*, vol. 26, no. 4, pp. 363–371, 2017.

[14] A. Boulanger, "Open-source versus proprietary software: Is one more reliable and secure than the other?," *IBM Systems Journal*, vol. 44, no. 2, pp. 239-248, 2005.

[15] S. Sridhar, K. Altinkemer, and J. Rees, "Software Vulnerabilities: Open Source versus Proprietary Software Security," in *Preceedings of Americas Conference on Information Systems*, Omaha, Nebraska, USA, Aug. 2005.

[16] N. Ullah, M. Morisio, and A. Vetro, "A Comparative Analysis of Software Reliability Growth Models using Defects Data of Closed and Open Source Software," in *Proceedings of the 35th Annual IEEE Software Engineering Workshop*, Greece, pp. 187-192, Oct. 2012.

[17] H.K. Browne, W. A. Arbaugh, J. McHugh, and W.L. Fithen, "A trend analysis of exploitation', in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 214–229, May 2001.

[18] O. H. Alhazmi and Y. K. Malaiya, "Prediction capabilities of vulnerability discovery models," in *Proceedings of annual reliability and maintainability symposium*, pp. 86–91, 2006.

[19] D. Nettleton, *Commercial Data Mining*, Chapter 6 - Selection of Variables and Factor Derivation, M. Kaufmann and et al. (Eds.), Boston, pp. 79-104, 2014.

[20] S. H. Houmb, V. N. Franqueira, and E. A. Engum, "Quantifying Security Risk Level from CVSS Estimates of Frequency and Impact," *Journal of Systems and Software*, vol. 83, no. 9, pp. 1622-1634, 2010.

[21] Y. K. Malaiya, N. Karunanithi, and P. Verma, "Predicta-bility of software reliability models," *IEEE Transa-ctions on Reliability*, vol. 41, no. 4, pp. 539-546, 1992.

Authors

HyunChul Joh is an associate professor at the School of Computer Science in Kyungil University, Korea, since March 2014. He was serving as an executive director at computing information center in Kyungil university from 2018 to 2020. From 2012 to 2014, he was a GIST college laboratory instructor in division of liberal arts and sciences at Gwangju Institute of Science and Technology (GIST) in Korea. His research focuses on modeling the discovery process for software security vulnerabilities and risk metrics. Recently he had started research on AI and big data analysis. He received his Ph. D. and M. S. in computer science from Colorado State University, CO USA, in 2011 and 2007, respectively. He also received a B. E. in information and communications engineering from Hankuk University of Foreign Studies in Korea, 2005.

JooYoung Lee is an associate professor at the School of Interdisciplinary Studies in Kyungil University, Korea, since March 2012. From 2010 to 2011, she was a part-time instructor in Department of Fashion Design at Chung-Ang University in Korea. Her research focuses on fashion design and technology. Recently she had started research on wearable computing of fashion design. She completed her doctoral course works at department of clothing in Chung-Ang University, Korea. She received her B. A. and M. A. in Fashion Design and Technology from University of Arts London (LCF) UK, in 2004 and 2009 respectively.