Seasonality and software vulnerabilities in major database management systems

HyunChul Joh

Abstract—Popularity and marketshare are very important index for software users and vendors since more popular systems tend to engage better user experience and environments. Periodical fluctuations in the popularity and marketshare could be vital factors when we estimate the potential risk analysis in target systems. Meanwhile, software vulnerabilities, in major relational database management systems, are detected every now and then. Today, all most every organizations depend on those database systems for store and retrieve their any kinds of informations for the reasons of security, effectiveness, etc. They have to manage and evaluate the level of risks created by the software vulnerabilities so that they could avoid potential losses before the security defects damage their reputations. Here, we examine the seasonal fluctuations with respect to the view of software security risks in the four major database systems, namely MySQL, MariaDB, Oracle Database and Microsoft SQL Server.

Index Terms—Relational database management system, Seasonality, MySQL, Oracle DB, SQL Server

I. INTRODUCTION

Generally speaking database management systems are tools which provide us storages where we can put and get any kinds of data. Even though we are well recognized that invaluable data should be stored somewhere else where others are not able to be accessed, if we need to reach that information from the Internet ourselves, technically there must be a possibility that others also reach to the data, when something goes wrong. While we store and retrieve information from the database systems, they provide a lot of handy concepts in terms of security, effectiveness, multiuser environment, etc. Database management systems such as MySQL, MariaDB, Oracle Database and Microsoft SQL Server are considered as de facto standard in the database industries.

Meantime, a software security vulnerabilities are a weakness factors created by developers mistakenly that could be misused by malicious users, so that they could cause harm or loss in the security system [1]. Sometimes those security bugs are traded in the black hat communities [2]. Security software vulnerabilities which is publicly revealed and not fixed are present security risks for the users and their organizations. A possible security risk could create a lot of damages for stakeholders’ reputations.

When software systems are developed, security vulnerabilities are generated frequently by coding mistakes. Once those security vulnerabilities are created, they might be found by variable reasons, such as many kinds of software tests or from normal software usages. Or some vulnerabilities could not be found until their host software systems are out of the market if we are lucky. When the software vulnerabilities are discovered by white hats or benign users, the security bugs are reported to the software vendors directly. The vendors then should create the corresponding patch and release to the users as soon as possible to minimize the bad impact. [3] in their paper shows that it usually takes about forty five days for delivering patches.

On the other hand, when a black hat or malicious user discover a security vulnerability, then the possibilities that the bug will be exploited will be high. Most of the time it is because of the economic reasons or their reputations in the black hat communities. The vulnerability detecting rate is able to be estimated by vulnerability discovery models in some degree for a target software systems when we have sufficient data sets [4]. Also malicious users and black hats are frequently share the information of vulnerabilities they found, and that could lead fertile security bridges which put a number of software users in danger. As a result, known vulnerabilities without proper patches create disorder in the related industry sectors. Fortunately, when we have a patch with no defects, then we expect the death of the vulnerability. However, occasionally, those vulnerability patches themselves introduce new vulnerabilities too.

In spite of the fact that the high reputations, the database management systems of MySQL, MariaDB, Oracle Database and Microsoft SQL Server frequently introduce security vulnerabilities. Vulnerabilities in those systems are specifically hazardous and sometimes they are extremely risky because those systems provide malicious users rights to access to the computing systems used by many database users. It could be possible that extremely sensitive information might be leaked or the system could be vandalized with no reason. Consequently, a lot of concerns are risen on very feasible security exploitations among the related IT industries and security researchers. In order to control the risk posed by software vulnerabilities related industries need to have some kind of methods to calculate their risk in numbers, so that they could manage their security risks under control. After all, “If You Can't Measure It, You Can't Improve It”.

Here, we are trying to investigate the four major relational database management systems, namely, MySQL, MariaDB, Oracle Database and Microsoft SQL Server. In this paper, first, we examine the seasonal fluctuations with a graph from the google Web search patterns. Then we are trying to represent some of future works in a sense of software
TABLE I

|                    | MySQL  | MariaDB | Oracle DB | SQL Server |
|--------------------|--------|---------|-----------|------------|
| Developer          | Originally MySQL AB, Sun Microsystems, Oracle | MariaDB Corporation | Oracle | Microsoft |
| Rank (Overall)     | 2      | 12      | 1         | 3          |
| Rank (RDBMS)       | 2      | 8       | 1         | 3          |
| Initial release    | 1995   | 2009    | 1980      | 1989       |
| Current release    | 8.0.22 (2020) | 10.5.8 (2020) | 19c (2019) | SQL Server 2019 (2019) |
| Implementation     | C/C++  | C/C++   | C/C++     | C++        |
| OS support         | FreeBSD, Linux, OSX, Solaris, Windows | FreeBSD, Linux, Windows | Solaris, Linux, Windows | Linux, Windows |

Although voluminous researches have been conducted on the software security vulnerability, researches focusing on the relational database management systems are not easy to find. Furthermore, much of the software vulnerability studies are concentrated on qualitative methods, and frequently, they have discussed about finding and preventing a particular security vulnerability. After entering the year 2000 when the datasets about software vulnerabilities had been enough to be examined by researchers. As a result, some researchers started to examine the security vulnerabilities in quantitative manners for major software systems such as Web servers and browsers or operating systems [5], [6]. Some researchers match quantitative and qualitative studies to science and art respectively [7]. Methods used in qualitative researches, just like an art, could be very subjective with respect to the experts’ opinions, and they might not be mathematically explained, while a method which is using quantitative methods are considered as science based on the clear numbers. Those actual data driven empirical analyses with statistic tests create objective results for a given issue. Therefore, for the future work related to this research area, which is related to the software vulnerabilities should go along well with a quantitative method more as long as data sets are provided. For the quantitative analysis for a software security vulnerability analysis, we could get the related data sets for the four relational database management systems could be collected on cvedetail.com where we could find all the software vulnerability information from NVD (nvd.nist.gov).

Fig. 1. Popularity comparisons by region.
using the Google Trends [https://trends.google.com] to figure out the popularities among the software systems.

Figure 2 shows us the Web searching pattern for the four database management systems. It is clearly observed that the number of searches about all the four databases has been decreased from the beginning in the plot. For the MySQL which is open source software has been placed number one for all the time followed by Microsoft SQL Server and Oracle database. That is because the open source software is free for use. As a result, the popularity is high in both academia and industry fields. Moreover, across the database systems, they share majority of the syntax. Only small portion of the syntax differ each others. Hence, users who like to learn SQL select MySQL for his first choice.

However, when it comes to the proprietary companies, they frequently, need to solve problems related to their databases every now and then. In that case, companies willing to pay for the service. It could be explained that commercial businesses prefer more reliable and easy to use system to the free software system. In Table I, the first, second and third ranks are Oracle, SQL Server and MySQL respectively. However, in Figure 2, the order of number of users are the opposite. That's probably because small businesses and students tend to use MySQL while big companies tend to use commercial software systems, and the former has bigger number of users than the latter cases.

Now when we check Figure 3 which only shows the past five years of the Web searching trend it clearly tells us that there is a some kind of fluctuation in the graph. The time line in the figure is expending from 2015.12.06 to 2020.11.29. The x-axis and y-axis represents the same significance from the previous figure. It shows that around winter time, the number of searches are lesser than the other time of the year. Furthermore, it is not as clear as the winter time, the middle of year time periods also seems tend to be inactive. Seasonality or periodicity in software systems are sometimes observed [5]. And those attributes could help a lot to characterize and mitigate the risk posed by users behaviors.
### TABLE II

**Top 10 Most and Least Preferring Countries.**

| Rank | Country       | MySQL | MariaDB | Oracle DB | SQL Server |
|------|---------------|-------|---------|-----------|------------|
| 1    | Timor-Leste   | 100%  | –       | –         | –          |
| 2    | South Sudan   | 100%  | –       | –         | –          |
| 3    | China         | 85%   | 1%      | 7%        | 7%         |
| 4    | Turkmenistan  | 69%   | –       | 31%       | –          |
| 5    | Malawi        | 69%   | –       | 20%       | –          |
| 6    | Iraq          | 67%   | –       | 26%       | –          |
| 7    | Estonia       | 67%   | –       | 33%       | –          |
| 8    | Réunion       | 66%   | –       | 24%       | –          |
| 9    | Niger         | 66%   | –       | 25%       | –          |
| 10   | Martinique    | 65%   | –       | 25%       | –          |

| Rank | Country       | MySQL | MariaDB | Oracle DB | SQL Server |
|------|---------------|-------|---------|-----------|------------|
| 1    | Finland       | 44%   | 6%      | 12%       | 38%        |
| 2    | South Korea   | 42%   | 6%      | 29%       | 23%        |
| 3    | Estonia       | 60%   | 4%      | 13%       | 23%        |
| 4    | France        | 46%   | 4%      | 23%       | 27%        |
| 5    | Slovakia      | 45%   | 4%      | 21%       | 30%        |
| 6    | Germany       | 43%   | 4%      | 23%       | 30%        |
| 7    | Madagascar    | 62%   | 3%      | 15%       | 20%        |
| 8    | Lithuania     | 55%   | 3%      | 11%       | 31%        |
| 9    | Bulgaria      | 54%   | 3%      | 10%       | 33%        |
| 10   | Laos          | 53%   | 3%      | 13%       | 31%        |

| Rank | Country       | MySQL | MariaDB | Oracle DB | SQL Server |
|------|---------------|-------|---------|-----------|------------|
| 1    | Åland Islands | –     | –       | 100%      | –          |
| 2    | St. Helena    | 24%   | 1%      | 45%       | 30%        |
| 3    | Qatar         | 22%   | <1%     | 41%       | 37%        |
| 4    | Oman          | 24%   | –       | 41%       | 35%        |
| 5    | Japan         | 32%   | 1%      | 39%       | 28%        |
| 6    | Liechtenstein | 21%   | –       | 39%       | 40%        |
| 7    | Bahrain       | 25%   | –       | 37%       | 38%        |
| 8    | Chad          | 64%   | –       | 36%       | –          |
| 9    | Kuwait        | 21%   | –       | 36%       | 43%        |
| 10   | Djibouti      | 35%   | –       | 35%       | 30%        |

| Rank | Country       | MySQL | MariaDB | Oracle DB | SQL Server |
|------|---------------|-------|---------|-----------|------------|
| 1    | Guernsey      | –     | –       | 4%        | 96%        |
| 2    | Bermuda       | –     | –       | 5%        | 95%        |
| 3    | Nicaragua     | 29%   | 1%      | 10%       | 60%        |
| 4    | Cayman Islands| 30%   | –       | 10%       | 60%        |
| 5    | Jersey        | 36%   | –       | 5%        | 59%        |
| 6    | Isle of Man   | 29%   | –       | 13%       | 58%        |
| 7    | Bahamas       | 33%   | –       | 9%        | 58%        |
| 8    | Monaco        | 43%   | –       | 57%       | –          |
| 9    | Aruba         | 44%   | –       | 56%       | –          |
| 10   | Libya         | 34%   | –       | 11%       | 55%        |

| Rank | Country       | MySQL | MariaDB | Oracle DB | SQL Server |
|------|---------------|-------|---------|-----------|------------|
| 1    | Guernsey      | –     | –       | 4%        | 96%        |
| 2    | Bermuda       | –     | –       | 5%        | 95%        |
| 3    | Nicaragua     | 29%   | 1%      | 10%       | 60%        |
| 4    | Cayman Islands| 30%   | –       | 10%       | 60%        |
| 5    | Jersey        | 36%   | –       | 5%        | 59%        |
| 6    | Isle of Man   | 29%   | –       | 13%       | 58%        |
| 7    | Bahamas       | 33%   | –       | 9%        | 58%        |
| 8    | Monaco        | 43%   | –       | 57%       | –          |
| 9    | Aruba         | 44%   | –       | 56%       | –          |
| 10   | Libya         | 34%   | –       | 11%       | 55%        |
In fact, those attributes are very well known research fields such as in the stock market [8] or birth defects [9].

In order to check whether there is a seasonal fluctuation in a given data set, researchers use the seasonal index. It represents the average value of a specific period of time about the exacts. The index could be achieved by Equation (1), where \( d \) is a grand average value, \( d_i \) is the average value for the \( i^{th} \) timepoint and \( s_i \) represents the seasonal index on the \( i^{th} \) time point [10].

\[
s_i = \frac{d_i}{d}
\]

Based on the seasonal index given above, for the future works could be conducted by applying the technique to the software vulnerabilities from the four database systems. From that kind of research we could expect that the optimal resource allocations from the database patch management to the related software marketing.

III. DISCUSSION

Table II represents top ten the most and the list popular countries in each database systems. When the sampling values are not significant, then the corresponding value is represented with a dashed line. For the top 10 most popular list of MySQL, there are four Asians, five Africans and a Caribbean island. They are all emerging countries, which does not have extra economic conditions. As a result, their first choice is naturally MySQL. It is betrayal of interest when we compare to the graphic design softwares such as Adobe products since Adobe cost policy virtually let users in those countries use their project for free [11], [12]. Some of countries in the list appear in the top 10 least popular lists from the two commercial systems which makes sense. On the other hand, in the top 10 least popular list of MySQL, there are seven Arabians, two Europeans and an African island (St. Helena). Among the Arabian countries appears again for the Oracle database friendly list, but interestingly no country is shown in the SQL Server list.

One of the black hat conference paper, called The laws of vulnerabilities [13], represents several vulnerability datasets showing the seasonal patterns or periodic fluctuations. Actually, in the paper, the authors do not emphasize the seasonal datasets, but four quantifiable and distinct characteristics which can be found in software vulnerabilities: Half-life, Prevalence, Persistence and Exploitation. However, among the datasets what day presents turns out that there are indeed periodic patterns [2]. Half-life represents “time interval for reducing occurrence of a vulnerability by half. Average duration of half-life continues to be about 30 days, varying by industry sector,” prevalence “measures the turnover rate of vulnerabilities in the top twenty list during a year. Prevalence has increased, with 60% remaining in the list in 2008 compared to 50% in 2004, persistence means “total life span of vulnerabilities. Persistence remains virtually unlimited,” and exploitation represents “time interval between an exploit announcement and the first attack. Exploitation is faster, often happening in less than 10 days compared to 60 days in 2004.”

It might be an interesting research, if we apply the seasonal data sets from the database systems into the autocorrelation function analysis and ANOVA (analysis of variance) test. While autocorrelation test is able to tell us that the period of seasonality, ANOVA test can pin point specific time point when the peak values appear. When we conduct the future work, then we could optimally allocate the proper resources at the right time, so that we could mitigate the potential risk posed by vulnerabilities located in the database management systems.

Furthermore, if we combine the idea of seasonal attributes into a one of vulnerability discovery model, then the model performance estimating the number of vulnerability detection process will be dramatically enhanced.

REFERENCES

[1] Pfleeger, C. P. & Pfleeger, S. L. (2003). Security in computing (3rd ed.). Prentice Hall PTR
[2] Allodi, L. (2008, May). Economic Factors of Vulnerability Trade and Exploitation. in proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, TX, USA, 1483–1499.
[3] Scarfone, K. & Mell, P. (2009). An Analysis of CVSS Version 2 Vulnerability Scoring. in proceedings of the 2009 International Symposium on Empirical Software Engineering and Measurement, 516–525.
[4] Wai, F. K., Yong, L. W., Divakaran D. M. & Thing, V. L. (2018). 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, Singapore, 220–225.
[5] Joh, H. (2015). Assessing Web Browser Security Vulnerabilities with respect to CVSS. Journal of Korea Multimedia Society, 18(2), 199–206.
[6] Alhazmi, O.H. & Malaiya, Y.K. (2008). Application of Vulnerability Discovery Models to Major Operating Systems. IEEE Transactions on Reliability, 57(1), 14–22.
[7] Joh, H. & Malaiya, Y. K. (2017). Periodicity in software vulnerability discovery, patching and exploitation. International Journal of Information Security, Vol.16, pp.673-680.
[8] Heston, S. L. & Sadka, R. (2008) Seasonality in the cross-section of stock returns. Journal of Financial Economics, Vol.87 No.2, pp.418-445.
[9] Carrion-Baralt, J. R., Smith, C. J., Rossy-Fullana, E., Lewis- Fernandez, R., Davis, K. L. & Silverman, J.M. (2006) Seasonality effects on schizophrenia births in multiplex families in a tropical island. Psychiatry Research, Vol.142 No.1, pp.93-97.
[10] Arsham, H (1994-2015) Time-Critical Decision Making for Business Administration. Accessed on 2020.12.04: http://home.ubalt.edu/ntsbarsh/business-stat/stat-data/forecast.htm
[11] Americanassembly.org (2011), Adobe Logic. Retrieved from https:// piracy.americanassembly.org/adobe-logic/
[12] Mondok, m. (2007). Microsoft executive: Pirating software? Choose Microsoft! Retrieved from http://arstechnica.com/microsoft/news/2007/ 03/microsoft-executive-pirating-software-choose-microsoft.ar
[13] Qualys, L: The laws of vulnerabilities 2.0. In Black Hat 2009, Presented by Wolfgang Kandek (CTO) (July 28, 2009)

HyunChul Joh is an associate professor at the School of Computer Science (A.I. Software major) in Kyungil University, Korea, since March 2014. He was serving as an executive director at computing information center in the same 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 security vulnerabilities and risk metrics. Recently he had started research on wearable computing, big data and artificial intelligence. 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.