Seven Years of Software Vulnerabilities: The Ebb and Flow

Hossein Homaei

homayi@aut.ac.ir

Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran

Hamid Reza Shahriari

shahriari@aut.ac.ir

Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran

Abstract

This article reviews the amount of software vulnerabilities that have surfaced during the last seven years to answer several questions. Which types of vulnerabilities are more common or more severe? Which ones need more attention and require programmers and developers to learn about their restriction and prevention mechanisms? Which ones should be prevented first to gain maximum benefit? We review vulnerability trends based on the vulnerabilities categorized in the National Vulnerabilities Database (NVD) with the hopes of figuring out which types of vulnerabilities have been restricted in recent years and which ones could still cause severe problems. The results of this study could potentially help security professionals focus on the most common or most severe vulnerabilities. It would be also helpful for programmers to concern themselves about code sanitization. Moreover, researchers could adopt this work to aid their development of prevention and detection methods with respect to common vulnerabilities.

Keywords

Software vulnerabilities, Vulnerability trends, Trend analysis

1- Introduction

A considerable amount of software vulnerabilities are reported every year, with some vulnerabilities being common and others rare. One specific type of vulnerability that is frequently reported in any given year might subside in subsequent years because of prevention and detection methods used by developers. When one vulnerability is addressed and nullified, attackers try to find new types of vulnerabilities and exploit them to penetrate software. On the other hand, security professionals develop more countermeasures to defend systems. In other words, the ebb and flow of vulnerabilities can change both the behavior of attackers and the concerns of defenders.

We have studied the 7-year fluctuation in software vulnerabilities to ascertain relevant vulnerability trends and their severity. Focusing on ascending trends would help security professionals find prevention methods early, warn their customers about more common or more severe vulnerabilities, and train developers in existing restriction methods. It would be also useful for programmers to concern themselves about code sanitization and vulnerability prevention in order to counteract growing pains with respect to software vulnerabilities in a proactive fashion. Moreover, researchers can develop new prevention and detection methods against more common vulnerabilities and improve cyber security.

We should trace vulnerability repositories and databases to study the trends and find severe vulnerabilities. The three most popular repositories are the Open Source Vulnerability Database (OSVDB) [1], the Exploit database [2], and the previously mentioned National Vulnerability Database (NVD) [3]. OSVDB only covers web-application vulnerabilities. The Exploit database is an archive of exploits and vulnerable software; as an exploit collection, it is not a suitable resource for the discovery of vulnerabilities. Therefore, we use NVD as our main resource of raw data.

According to the NVD's website, "NVD is the U.S. government repository of standards based vulnerability management data represented using the Security Content Automation Protocol (SCAP)." [3] NVD contains common vulnerability exposures (CVE) vulnerabilities, US-CERT alerts, US-CERT vulnerability notes, and OVAL queries and is one of the most complete repositories of reported vulnerabilities. Therefore, we use its statistics to reveal trends and uncover the importance of each vulnerability. We consider the statistics since 2008 because of the lack of accurate classification of vulnerabilities at NVD before 2008. For example, 6514 vulnerabilities were reported at NVD in 2007, but only 2731 of them have been classified.

NVD uses a subset of common weakness enumeration (CWE) [4] graph construction to categorize vulnerabilities. In the final revision applied by NVD, this classification encompasses 35 categories. However, the previous 2014 version only had 23 categories. Thus, to find trends and compare categories for the entire time period, we should merge these two classifications.

The following rules were applied as we pruned the two existing classifications to construct ours:

1. Each category of 2014 classification that does not exist in the previous version was merged with its parent node in the tree. If the parent does not yet exist in the previous version, the process is repeated until finding an existing category or reaching the root in the tree. The reason behind the rule is that the details of a new category in 2014 are unknown in the previous classification. Therefore, the category should be generalized to an abstract one. For example, we mapped the "Improper Access Control" category in the 2014 classification with the abstract category "Permissions, Privileges, and Access Controls".
2. The root category of 2014 classification (location) is mapped to the "Not in CWE" category in previous version because it did not exist before 2014.
3. The "Design Errors" category existed in the previous classification, but does not exist in the current version. We mapped this category to "Not in CWE" for the same reason as the previous rule.
4. The "Insufficient Information" and "Other" categories were merged with "Not in CWE" and the whole category was named "useless" because we cannot extract any useful information about vulnerability trends from this category.

The modified version of the classification has 20 categories. In the rest of the article we use these categories as follows: Authentication Issues (Auth); Buffer Errors; Code Injection; Configuration Issues; Credential Management Vulnerabilities; Cross-Site Request Forgery (CSRF); Cross-Site Scripting (XSS); Cryptographic Issues (Crypto); Format String Vulnerabilities; Information Leakage; Insufficient Input Validation; Link Following; Numeric Errors; OS Command Injection (OS); Path Traversal; Permissions, Privileges, and Access Control Issues (Access Control); Race Condition; Resource Management Errors; SQL Injection (SQLi); Useless. The definition of each category can be found in [4].

<u>2- Vulnerability trends at first glance</u>

Figure 1 shows the overall vulnerability trends from 2008 through 2014. The blue and orange lines represent all (blue line) and severe (orange line) vulnerabilities registered on NVD. We demonstrate the ratio of severe vulnerabilities to all reported vulnerabilities for each year on the orange line.

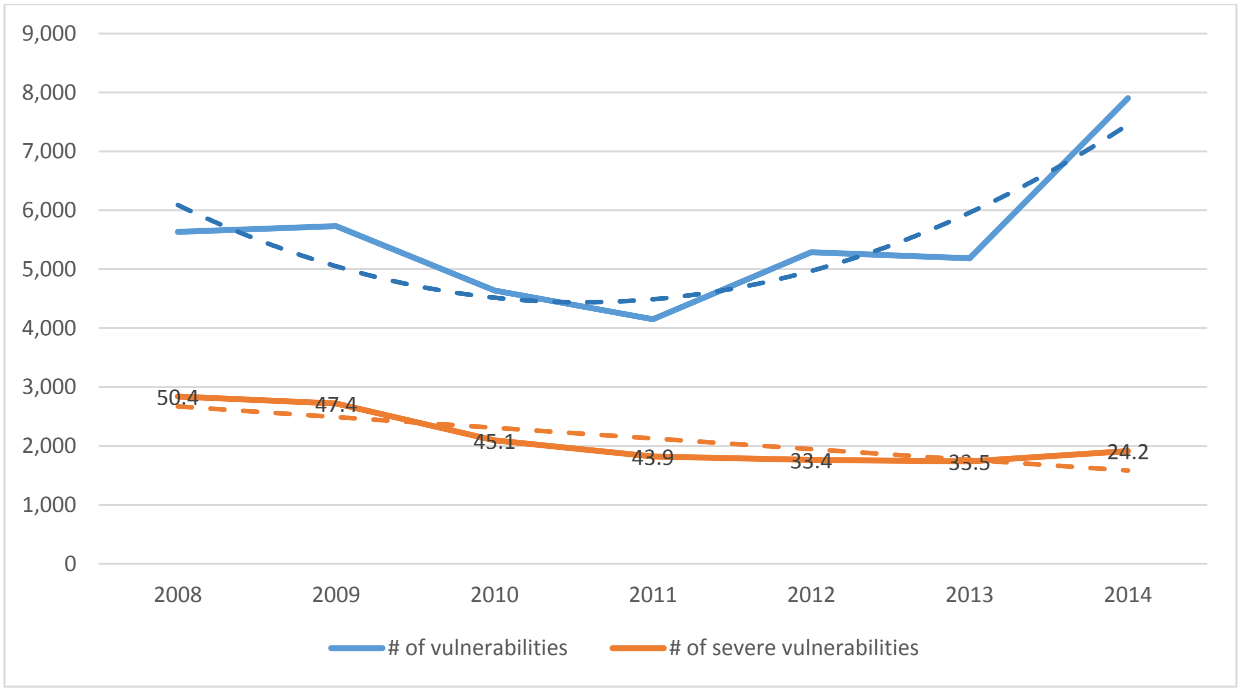

Figure 1. Overall vulnerability trends

Although the number of vulnerabilities decreased in 2010 and 2011, the overall trend is ascendant. However, the number of severe vulnerabilities has decreased. Even in 2014, when an exponential growth occurred in the amount of vulnerabilities, only 24% of all vulnerabilities were severe. In other words, although it is becoming easier to find vulnerabilities, only a limited number of those vulnerabilities lead to significant consequences when exploited.

3- Increases and decreases in vulnerabilities

Although overall trends could be useful to analyze the ceaseless competition between attackers and defenders, we should have more detailed information to discuss future opportunities and threats. Whereas the amount of some types of vulnerabilities has decreased, some types have instead increased. Along these lines, each category is presented separately in Figure 2. Since the scales of vulnerability occurrences vary from category to category, we split the main chart into two charts. Charts B and C represent vulnerability trends, but we separate them based on their scales to be more distinguishable. If a vulnerability occurs less than 250 times in every year, it appears in Chart C.

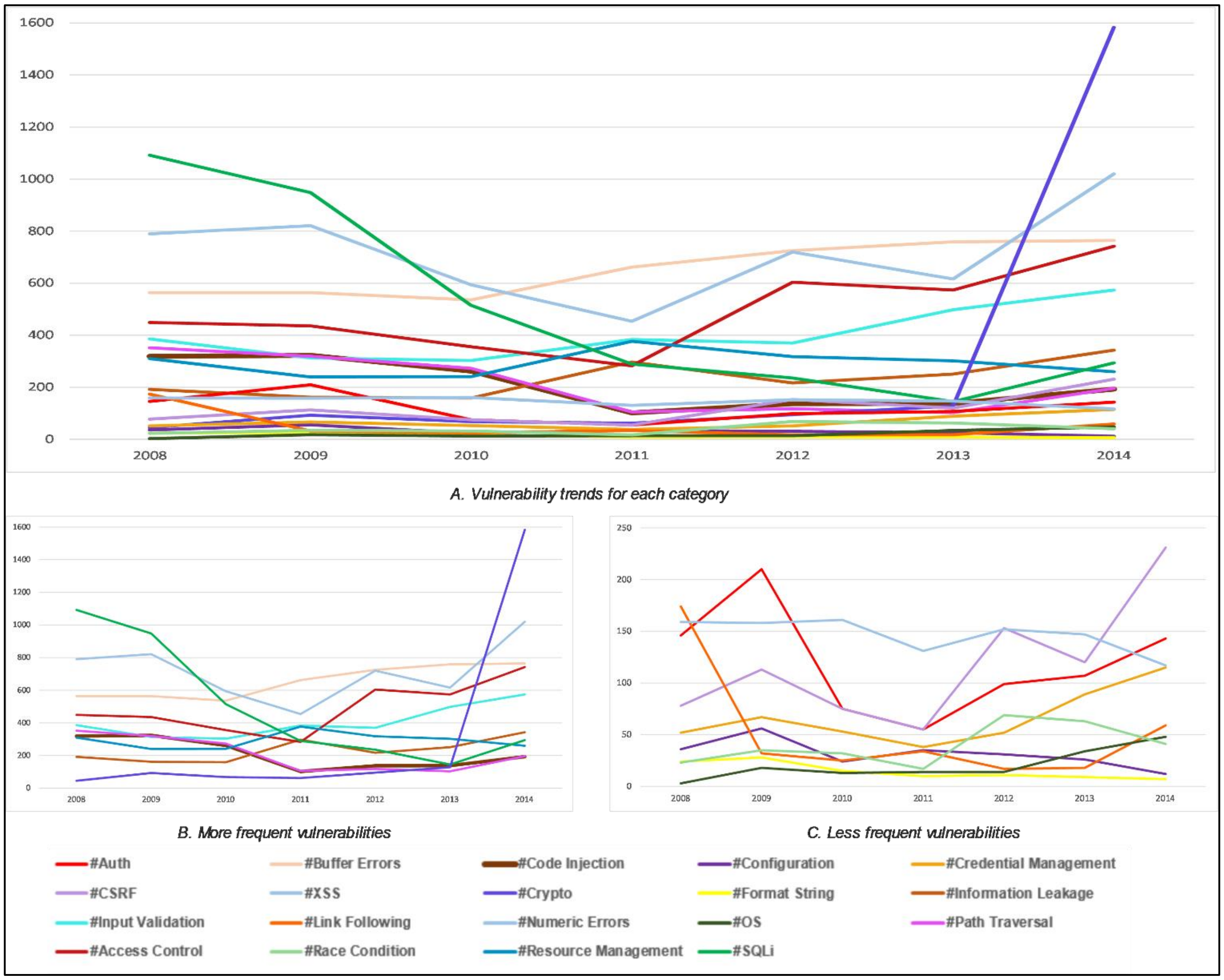

Figure 2. Vulnerability trends for each category

These charts illustrate that most categories have limited ebb and flow in a bounded range. The most interesting thing in these diagrams is the fluctuation in the categories of Cryptographic Issues and SQL Injection.

The number of cryptographic issues suddenly grew last year. This type of vulnerability skyrocketed to number one in 2014 after a more-than 1100% increase over the previous year. Additional study of the amount of cryptographic issues in 2014 demonstrates that approximately 88% of these vulnerabilities occurred in September and October. The main cause of this surge was a specific type of vulnerability in Android applications. There were lots of Android applications in which X.509 certification was not properly verified, making it possible to perform man-in-the-middle attacks. Since this type of vulnerability is categorized as a Cryptographic Issue, the amount of reported cryptographic vulnerabilities suddenly increased during these two months. However, it has decreased in recent months; in a sense, it can be described as a proverbial “flash in the pan”. In spite of this, more research on security testing of the implementation of cryptographic features should be performed to restrict possible similar future vulnerabilities, especially for Android applications.

Although SQL Injection does not have any sudden change in incidence, the number of these types of vulnerabilities has greatly decreased (gradually) over the studied time period. The magnitude of the standard deviation (373), average absolute deviation (300), and MAX-MIN difference (947) in comparison with the average value (503) proves the wide variability of SQL injection.

Figure 2-C demonstrates that, in the group of less frequent vulnerabilities, CSRF and Link Following vulnerabilities have the greatest growth and decline, respectively. The trends also indicate that there is generally an upward trend in three categories: CSRF, Credential Management, and Authentication vulnerabilities. Thus, it is likely that more will be heard about these vulnerabilities in the future..

4- Which types of vulnerabilities are more severe?

Although some vulnerabilities are more common and reported more than others, they do not have serious consequences on the system. Thus, it is also worthwhile to indicate dangerous vulnerabilities. We use Common Vulnerability Scoring System (CVSS) base score as a metric to extract the most serious vulnerabilities. The vulnerabilities with high severity (7 to 10) are assumed dangerous.

Figure 3 illustrates the trends of severe vulnerabilities. As with the previous figure, we split the main chart into two parts. Charts B and C represent severe vulnerability trends, but we separate them based on their scales to be more distinguishable. For this depiction, Chart C shows vulnerabilities with less than 150 occurrences per year.

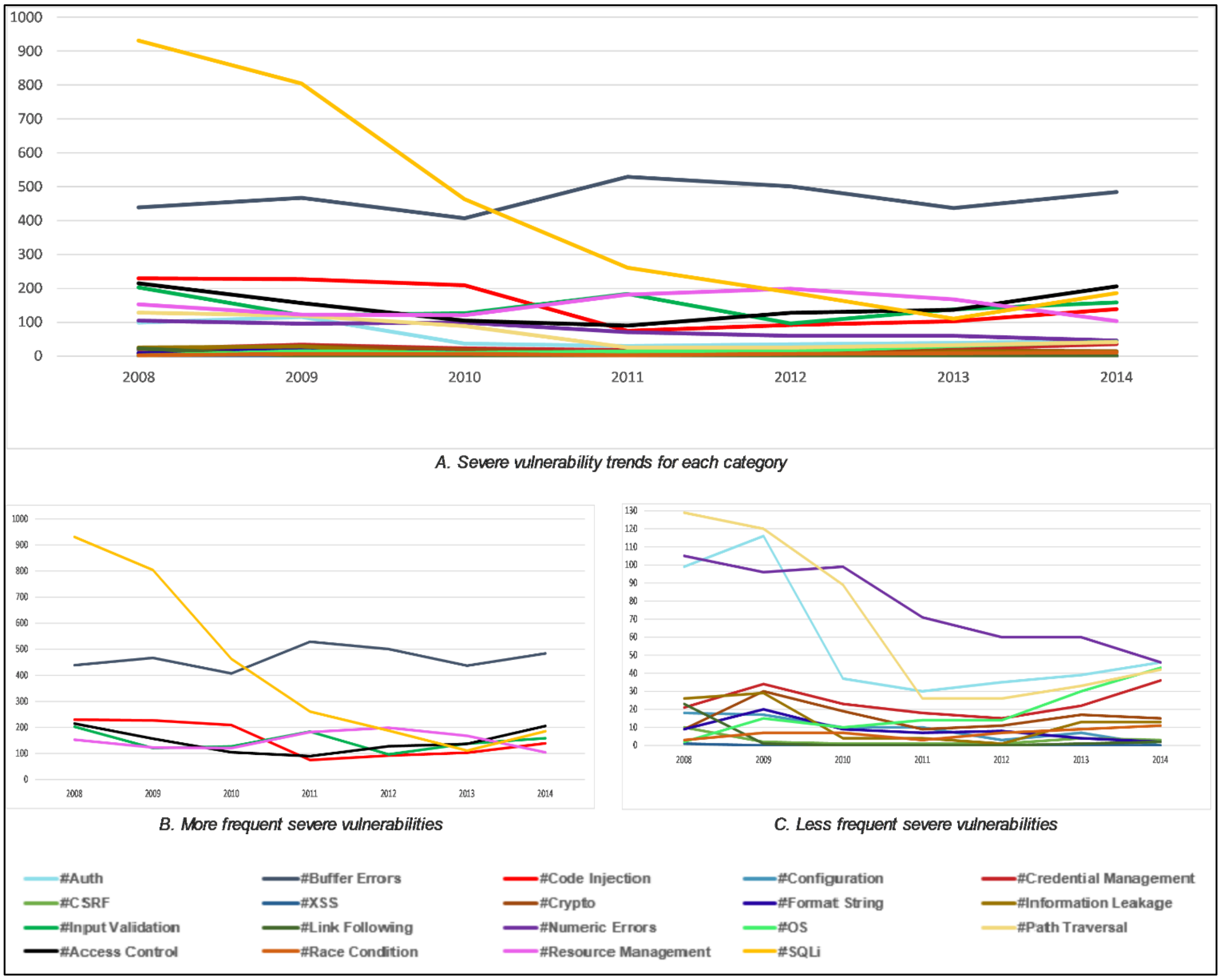

Figure 3. Severe vulnerability trends for each category

Figure 3-B shows the results from our tracking of the six most-reported severe vulnerability categories. Other categories have rarely occurred and are therefore shown in Figure 3-C. Comparing diagrams illustrates that:

- Buffer errors are almost always at the top of the severe vulnerability list with an ordinary fluctuation during the time. The Average absolute deviation equals 33. This variation is acceptable when compared with the magnitude of the average value (466).
- Although SQL injections are important severe vulnerabilities, the overall trend of this type is descending.

- The number of severe access control vulnerabilities is in an upward trend. Accordingly, developing more automatic tools to detect access control vulnerabilities will be of high value.
- For severe vulnerabilities, there has been normal fluctuations in some types of vulnerabilities. Other vulnerabilities have been either reported less or have not been severe. (SQL injections are one exception, in that they fluctuated widely.)

It is worth mentioning that many severe vulnerabilities have been classified in the "Insufficient Information" category by NVD. We have excluded these vulnerabilities from the figure since details of them are unknown or unspecified. Therefore, useful information about these types of vulnerabilities cannot be extracted.

We have now presented the number of severe vulnerability types and discussed the more frequent ones and their trends. But which category is more severe than the others? We use two factors to determine the severity of each category: the ratio of the number of occurrences of a specific type of severe vulnerability to all reported severe vulnerabilities; and to all reported vulnerabilities in the corresponding category. This process reflects the importance of knowing not just the number of occurrences for any given severe vulnerability in each category, but also the probability that the reported vulnerability will cause severe damage.

Figure 4 shows these two factors simultaneously. The quantities are calculated based on the average values over seven years. We also add two error bars for better visualization of the variation in the results. The black error bar shows the 95% confidence interval of standard deviation. The red error bar illustrates the range of changes from minimum to maximum values during the years 2008 through 2014.

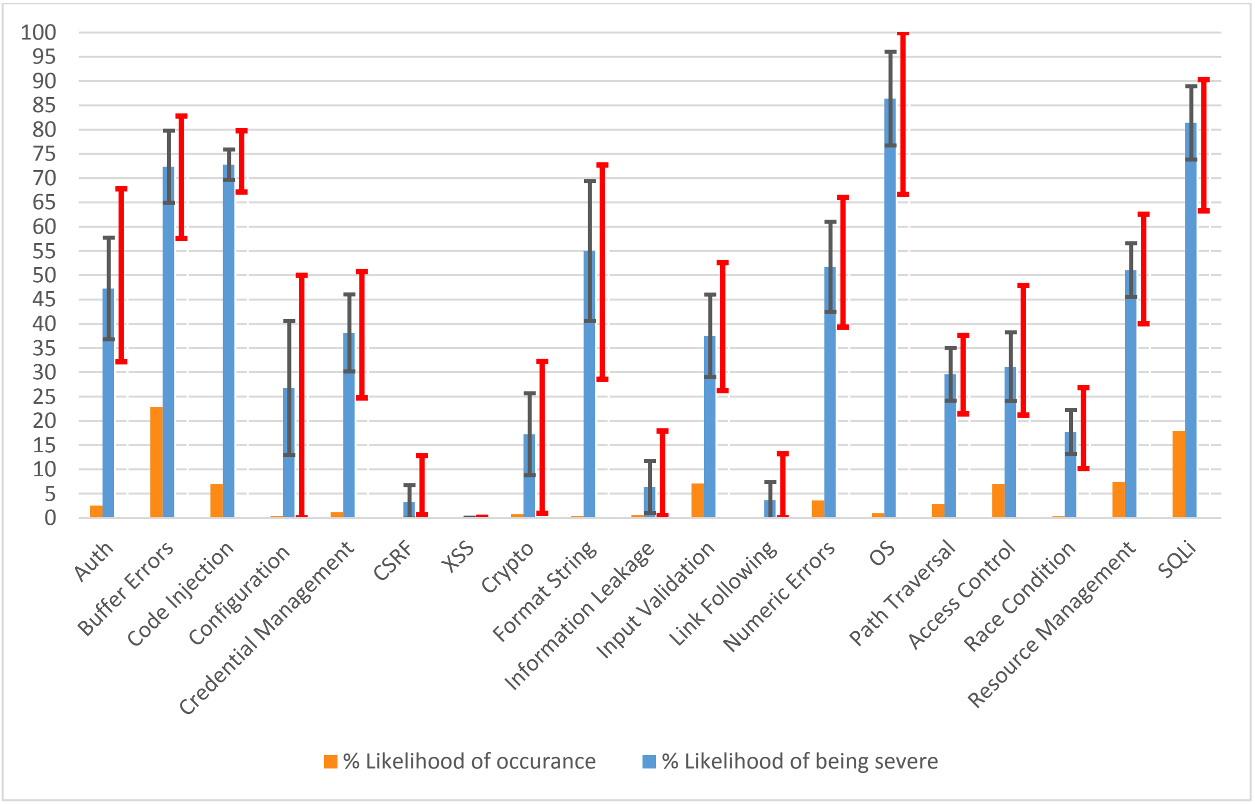

Figure 4. Severity of vulnerability types

The diagram illustrates that the Buffer Errors, SQL Injections, Code Injections, and Resource Management Errors are the most severe vulnerabilities, as they are more common than other vulnerabilities (the probability of occurring each of them is more than 5%) and also often lead to severe impact (the probability of causing dangerous impact is more than 50%). To give one example, for the time period studied nearly 23% of all vulnerabilities were buffer errors and 72% of these errors were severe. It is also worth mentioning that the OS command injection has rarely happened. However, if this vulnerability occurs, its exploitation has a high probability, nearly 85%, of causing software a severe problem.

5- Great benefit with relatively little effort

Lots of software vulnerabilities are reported every year. Some types of vulnerabilities are very common and some are rare. Types that count as common or rare could change year over year; a vulnerability type could be very typical in one year and not popular the following year. In this section we discuss the most common vulnerability types to see if they are identical every year. If they are repeated every year, how many vulnerabilities could be prevented by avoiding these common types? In other words, can we find any rule similar to the “Pareto principle“ in the domain of vulnerability prevention? Which vulnerability types should be prevented first to gain maximum benefit?

Figure 5 shows the five most often reported vulnerabilities in each of the last seven years. The collection of these vulnerabilities in each year encompasses approximately 50 percent of all reported vulnerabilities in the year. Therefore, this figure could provide an abstract view of the most prevalent vulnerabilities in each year.

Two of the most often reported web application vulnerabilities are SQL Injection and XSS. SQL Injection was the most reported vulnerability in 2008 and 2009. But the number of SQL injections declined in 2010 and finally disappeared from top-5 list in 2011. Cross-Site Scripting (XSS) almost always has been the second most-popular vulnerability, except in 2010 when it was first.

Although specific web application vulnerabilities were noteworthy until 2010, buffer errors gradually got more attention. As illustrated in the figure, the most often reported vulnerability during 2011 to 2013 was buffer errors.

The figure also illustrates that just 9 categories of the classification encompass all of the vulnerabilities that appeared in the top five in terms of being reported most often through seven years. We can also conclude that five vulnerability categories are more interesting from the attacker point of view. Thus, it is worth paying particular attention to these vulnerability categories and cautioning developers to avoid them. These vulnerabilities include Buffer Errors, Cross-Site Scripting, Access Control Vulnerabilities, SQL Injection, and Insufficient Input Validation.

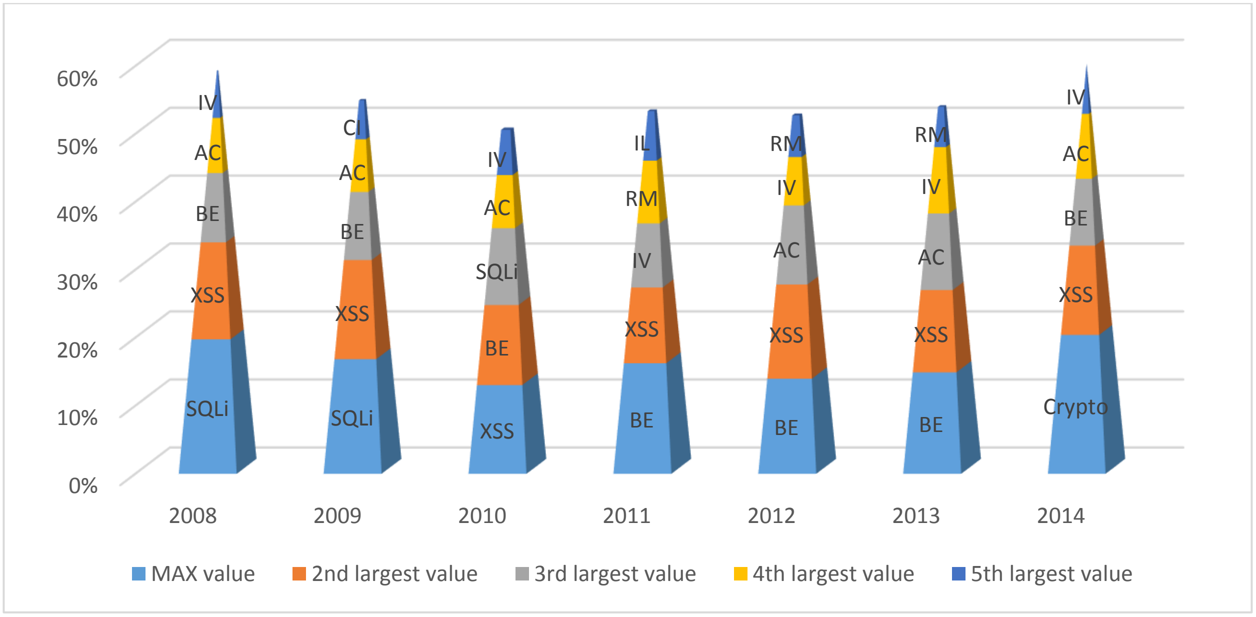

Figure 5. Top-5 yearly vulnerabilities
(abbreviations: AC=Access Control, BE=Buffer Errors, CI=Code Injection, IL=Information Leakage, IV=Input Validation, RM=Resource Management)

Figure 6 shows the first 4 severe vulnerability categories in each year. Like the previous chart, the collection of these vulnerabilities in each year approximately encompasses 50 percent of all reported severe vulnerabilities. It is also amazing that just six categories encompass the top-4 reported severe vulnerabilities: Buffer Errors, SQL Injection, Access Control Vulnerabilities, Insufficient Input Validation, Resource Management Errors and Code Injection.

We have reviewed how SQL Injections and Cross-Site Scripting vulnerabilities change over time (as shown in Figure 5). But there are some interesting points to be noted when considering the severity. Surprisingly, Cross-Site Scripting, which has been the second most often reported vulnerability, has never been the most often reported severe vulnerability. In other words, although many cross-site scripting vulnerabilities have been reported every year, they have seldom caused serious problems. Moreover, in spite of the reduction in the amount of reported SQL injections, this category is still one of the most dangerous vulnerabilities.

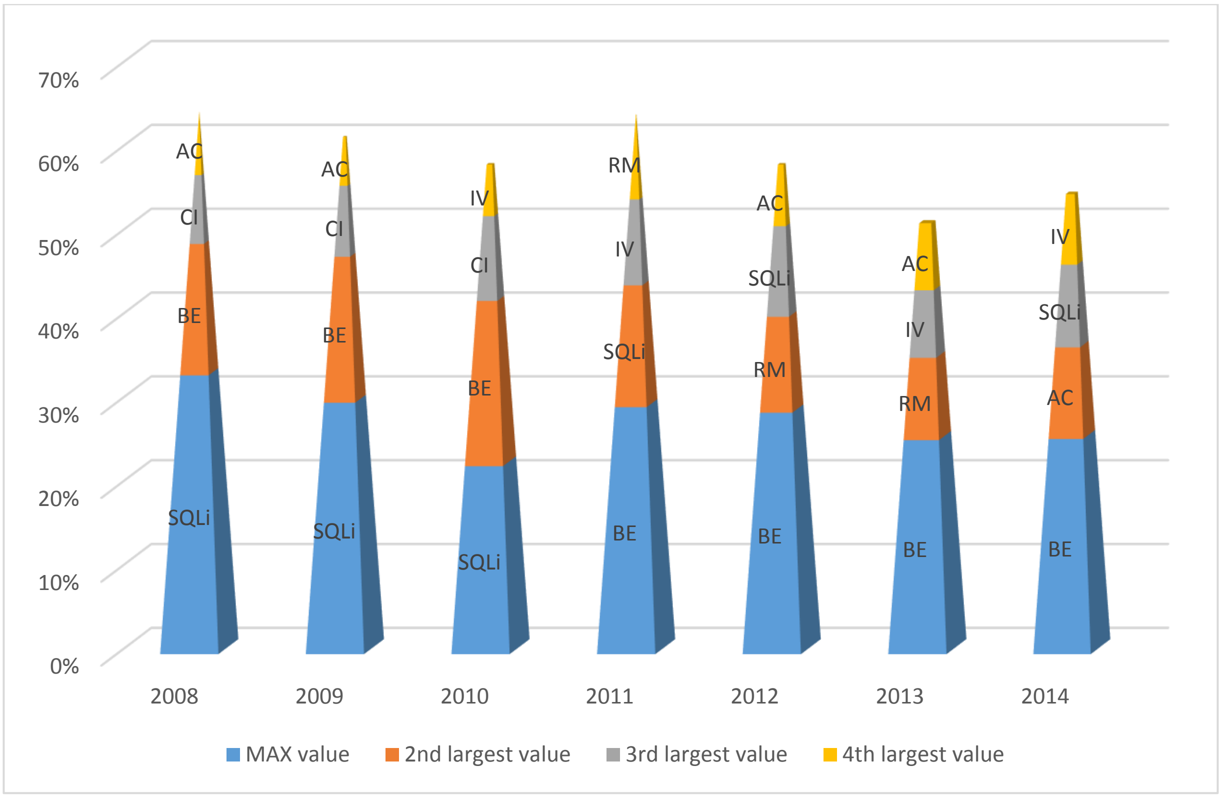

Figure 6. Top-4 yearly severe vulnerabilities
(abbreviations: AC=Access Control, BE=Buffer Errors, CI=Code Injection, IV=Input Validation, RM=Resource Management)

In addition to the study of vulnerabilities in each year, it could be useful to have an abstract view of most reported vulnerabilities over the entire time period. This could be useful when deciding which vulnerabilities were most important for the entire time period, and eventually drive the answer to the question of which vulnerability types should be prevented first to maximize benefit and how many vulnerabilities could be prevented by avoiding these vulnerability types.

Figure 7 illustrates the average amount of most reported vulnerabilities from 2008 through 2014. We choose only those categories which consisted of more than 5% of the average number of reported vulnerabilities. Other vulnerability types are aggregated and represented as a single category named "other categories". In the outer doughnut, the categories are arranged clockwise from most-reported to least-reported severe vulnerability. But the inner doughnut, which represents all reported vulnerabilities, arranged in the same order as the outer doughnut.

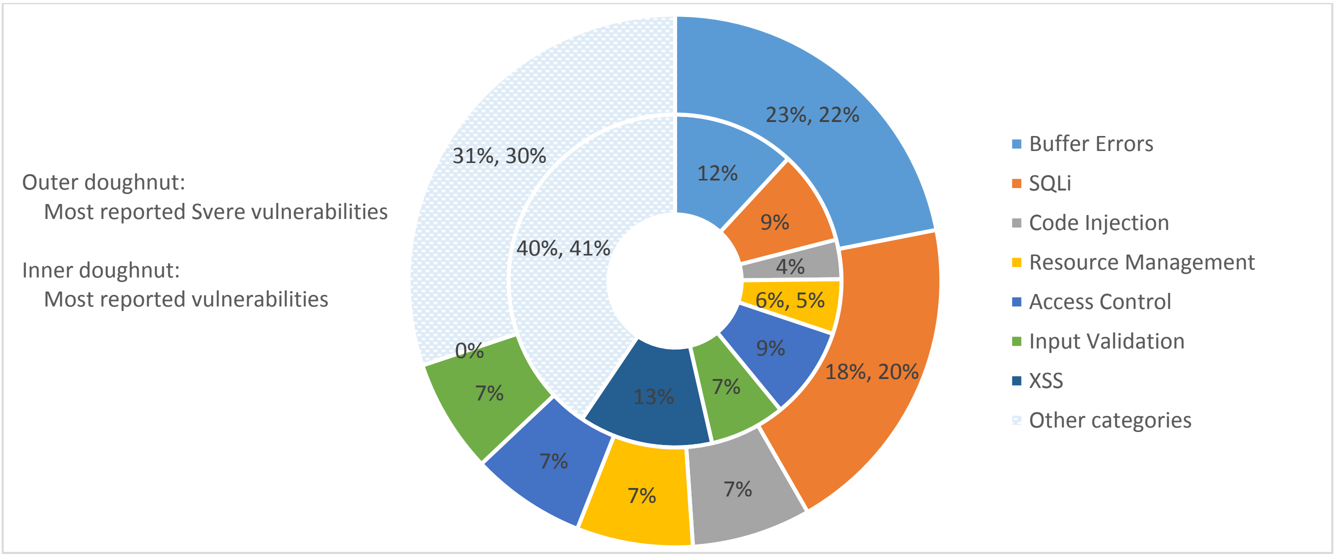

Figure 7. The percentage of each vulnerability type occurred from 2008 through 2014

The numbers on the chart indicate the average number of vulnerabilities in each category. For example, the Code Injection category on average consists of 7% of all reported severe vulnerabilities. But only 4% of all vulnerabilities are code injections. Some of the slices of the doughnut chart contain two numbers because we calculate the average quantity in two ways. The left side number is calculated by averaging the percentage of vulnerabilities in each category in each year. The right side number is computed by dividing the total number of reported vulnerabilities in the category by the entire amount of reported vulnerabilities. Obviously, for the outer doughnut, the number of vulnerabilities should be replaced with the number of severe vulnerabilities in the formula. We show only one value for a slice if the two formulas yield equal results.

Astonishingly, the average number of reported severe vulnerabilities just for six categories is above 5%. Approximately, more than 70% of severe vulnerabilities and 46% of all vulnerabilities could be prevented by avoiding these six vulnerability categories: Buffer Errors, SQL Injection, Code Injection, Resource Management Errors, Access Control Issues, and Insufficient Input Validation. Moreover, if programmers beware of XSS in addition to these six categories, nearly 60% of all vulnerabilities will be probably prevented.

6- Conclusion

In this section, we summarize previous charts, give an abstract view of vulnerability distributions, and provide conclusions about our work. Figure 8 shows the distribution of vulnerabilities in each category from 2008 through 2014. Each category could be compared with others in the same year or in other years.

Notice that the sum of the portions is not equal to 100 percent in each year because we exclude the “useless” category from the chart. Although this category comprises some special types that cannot be integrated into our merged classification, the predominant part of it includes vulnerabilities which are in the NVD’s “Insufficient Information” class. For example, more than 18% of vulnerabilities reported in 2013 do not have sufficient information to be classified into the existing categories.

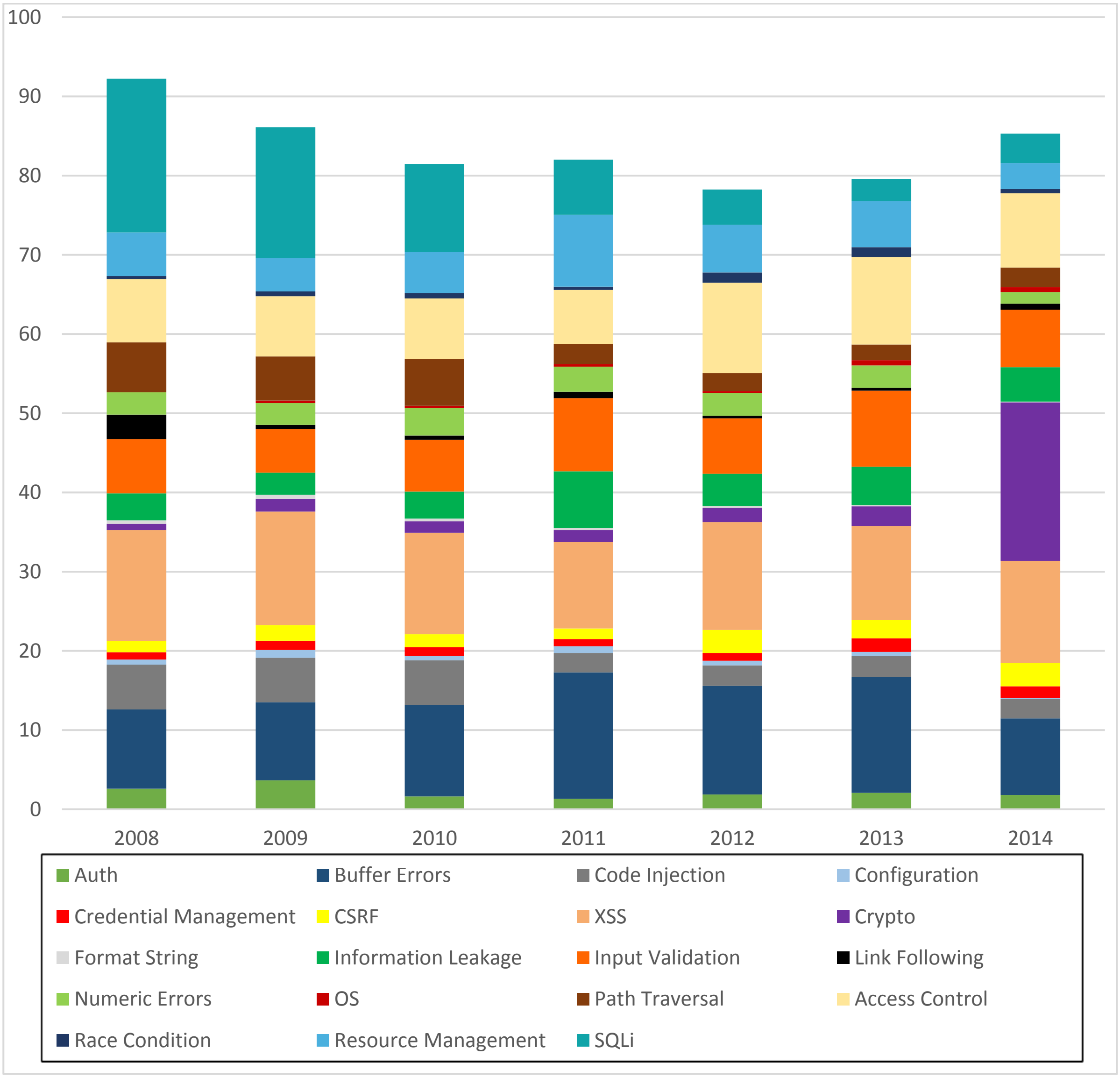

Figure 8. Vulnerability distribution

Analyzing the 7-year NVD's statistics for software vulnerabilities demonstrates the following results:

- The total number of reported vulnerabilities is ascending. But the portion of them that might cause severe problems is descending.
- Buffer Errors, XSS, and Access Control problems have been in the list of most reported vulnerabilities almost every year. Furthermore, buffer errors are almost always at the top of the severe vulnerability list.
- There is an upward fluctuation in "Insufficient Input Validation", "CSRF", and "Credential Management". But the greatest growth in the amount of reported vulnerabilities occurred in the "Cryptographic Issues" category in September and October 2014. However, this type of vulnerability decreased in the following months. Thus, we guess that a valley will be formed near this peak.
- The OS command injections would cause severe consequences if exploited, but fortunately, this exploitation rarely happened during this time period. However, Buffer Errors, SQL Injections, Code Injections, and Resource Management Errors are the most severe vulnerabilities. Because these vulnerabilities are not only more common than many others, but also many of their occurrences lead to severe impacts.
- More than 70% of severe vulnerabilities could be prevented by avoiding these categories: Buffer Errors, SQL Injection, Access Control Vulnerabilities, Insufficient Input Validation, Resource Management Errors and Code Injection. Moreover, 60% of all vulnerabilities would have not occurred if programmers had prevented Cross-Site Scripting in addition to these 6 categories.

To conclude, we recommend programmers learn more about Input Validation, XSS, and CSRF. Furthermore, we encourage researchers to develop more methods and tools to detect cryptographic problems and access control vulnerabilities. Developers should also be cautioned with respect to the severe consequences of buffer errors, SQL injections, and code injections.

References

[1] "OSVDB: Open Sourced Vulnerability Database," January 2015; http://osvdb.org.

[2] "Exploits Database by Offensive Security," January 2015; http://www.exploit-db.com.

[3] "National Vulnerability Database," January 2015; https://nvd.nist.gov.

[4] “CWE- Common Weakness Enumeration,” January 2015; http://cwe.mitre.org.