Classifying Common Security Vulnerabilities by Software Type

Onyeka Ezenwoye[1], Yi Liu[2], and William Patten[1]
[1] Augusta University, Augusta, GA, USA, oezenwoye,wpatten@augusta.edu
[2] University of Massachusetts Dartmouth, Dartmouth, MA, USA, yliu11@umassd.edu

***Abstract*—The National Vulnerability Database does not identify a type for the software that is impacted by a specified weakness. To gain some insight into the security vulnerability landscape, we classify by software type a total of 51,110 vulnerability entries from 2015 to 2019. The software types are operating system, browser, middleware, utility, web application, framework, and server. This classification shows the pattern of prevalence of software weaknesses and the persistence of weaknesses as they pertain to each software type.**

Keywords: Software, Security, Vulnerability, Weakness, Taxonomy.

I. Introduction

Since 2002, the National Vulnerability Database (NVD) [2] has maintained a list of exploitable security vulnerabilities that exist in software. Each vulnerability entry in the list has multiple attributes, one of which is the software weakness (E.g., buffer error). An aggregate of common weaknesses over time is available [3]. The vulnerability entries span many thousands of products which range from the recognizable to the very obscure. The database however does not specify a software type for these products. Identifying the software type is important in understanding the vulnerability landscape as it pertains to each software type [9].

With the ubiquity of software comes the associated developer-driven software weaknesses. Knowing the weakness characteristics of the software type is important in devising targeted fault avoidance and detection mechanisms which include threat modeling, architecture review, risk analysis, training, and policies for evolution and maintenance [5], [12]. To this end, we are analysing vulnerability database entries. Part of this effort includes classifying the vulnerabilities by software type. Here, we report our findings thus far of classifying the most recent five years of vulnerability data. With this, we seek to provide answers to two questions:

1) Is the occurrence of the most common weaknesses consistent across software types?
2) How well do weaknesses persist over years for the same product?

With these answers, we provide some additional insight into vulnerability weaknesses that isn't available in existing literature. The rest of this paper is structured as follows, our approach is described in Section II. We present results in Section III. Related work can be found in Section IV with Conclusion in Section V.

DOI reference number: 10.18293/SEKE2020-047

II. Approach

The NVD catalogs security vulnerabilities in a list known as Common Vulnerabilities and Exposures (CVE). Each entry in the list contains attributes such as a unique identifier, the product's vendor, product name, description, severity score, etc. Each CVE entry also contains a Common Weakness Enumeration (CWE) name [1]. The CWE name identifies the specific vulnerability type (E.g., Improper Authentication, and Buffer Error). From here on, we refer to each CVE entry as vulnerability and each vulnerability type as weakness.

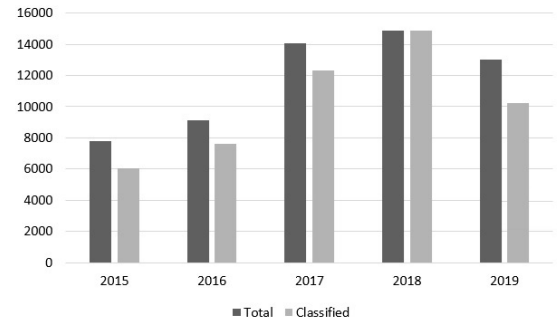

Fig. 1. Total number of vulnerabilities reported Vs number of vulnerabilities classified by software type

Each vulnerability contains the product's vendor and name (E.g., Microsoft and Windows 10) but not the product type (E.g., Operating System). To answer the questions from Section I, we decided to review all vulnerabilities from 2015 to 2019 and classify each vulnerability by a product type. We classified each vulnerability into one of seven product types. We use a database (of vendor, product name, and product type groupings) to map each vulnerability to a product type. We continue to update this database (with vendor, product name, and product type groupings) in order to achieve 100% classification for all years. Currently only 2018 is complete (Figure 1). The product types are Web Application, Utility, Server, Operating System, Browser (Web), Framework, and Middleware. The types are loosely based on the taxonomy proposed by Forward et al. [9]. We briefly describe each type:

- Web Application: web-based (or cloud-based) software such as content management systems, information management systems, transaction processing systems, etc.

- Utility: standalone applications such as productivity, creativity, antivirus, scripts, non-web clients, etc.
- Server: system servers (including cloud-based) such as database, email, proxy, web, FTP, DNS, load balancers, network monitors, etc.
- Operating System: firmware, device drivers, virtual machines, and all types of operating systems.
- Browser: all types of web browsers.
- Framework: software components such as libraries, plugins, and extensions.
- Middleware: enterprise transaction platforms such as message queuing, object storage, and identity management systems.

Of the 58,781 unique vulnerabilities over that period, we have classified 51,110 (87%), including 100% of all vulnerabilities from 2018. Figure 1 shows a comparison of the number of vulnerabilities from each year and the number of those that have been successfully classified by product type, so far. We are confident that at this point, the results from the number that have been classified should be fairly representative of the total. To support this argument, Figure 2 shows a comparison distribution by software product between 2018, which has been 100% classified, and all other years. The chart shows a similarity in distribution across product type. It also shows that operating systems and utilities account for about half of all vulnerabilities.

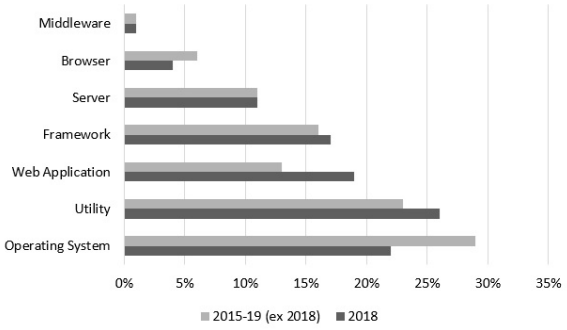

Fig. 2. Relative vulnerability count by software product type

From our classification, we found that there are a total of 169 distinct weaknesses across all product types. Figure 3 shows the most common of these weaknesses. By most common we mean that each weakness has a count that is at least 3% of all vulnerabilities classified (51,110). Collectively, these 7 weaknesses account for just over 53% of all vulnerabilities, the other 162 weaknesses account for the rest. The chart shows that buffer errors and cross-site scripting are the two most individually occurring security weaknesses in general.

III. Results

In this section we offer some answers to the questions discussed in Section I.

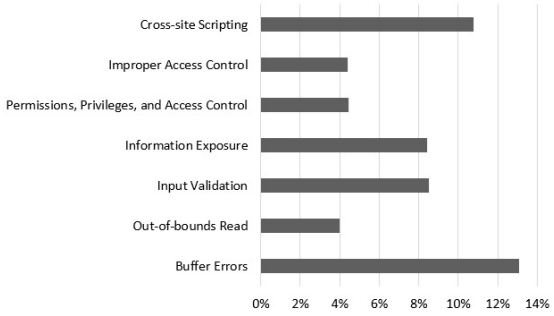

Fig. 3. Most common weaknesses across all vulnerabilities (2015-2019)

A. Is the occurrence of the most common weaknesses consistent across software types?

To help answer this question, we present a breakdown of the most common weaknesses by each product type. Figure 4 shows the most common weaknesses for the Browser type. For this type, there were 74 different weaknesses across a total of 2,897 vulnerabilities. These 8 weaknesses account for 75.5% of all vulnerabilities. Buffer error is the most occurring weakness, which is consistent with Figure 3.

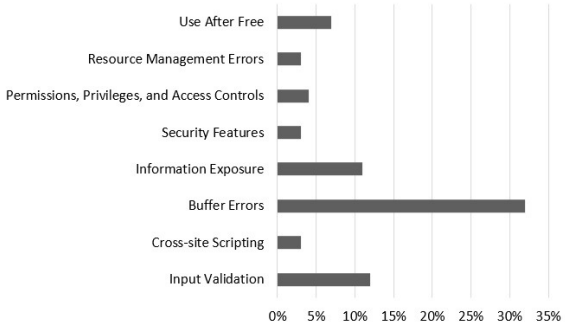

Fig. 4. Most common weaknesses for the Browser type

Figure 5 shows the most common weaknesses for the type classified as Framework. There are a total of 129 weaknesses across 8,444 vulnerabilities. These 9 weaknesses account for 59.6% of all vulnerabilities. The two most common weaknesses are buffer errors and cross-site scripting. This could be explained by the fact that many of the products that fall into this type are frameworks for web applications and other downloadable program libraries. The two most common here is somewhat consistent with Figure 3.

Figure 6 shows the most common weaknesses for the Middleware product type. There are a total of 59 weaknesses across 441 vulnerabilities. These 9 weaknesses account for 54.6% of all vulnerabilities. Although present, buffer errors, a typically common weakness (Figure 3), don't feature prominently here. Figure 7 shows the most common weaknesses for Operating systems. There are a total of 148 weaknesses across 13,670 vulnerabilities for this product type. These 7 weaknesses account for 54.1% of all vulnerabilities. All of these weaknesses are the same that appear as the most common

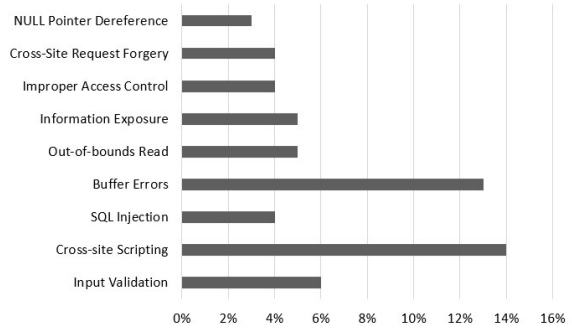

Fig. 5. Most common weaknesses for the Framework type

in Figure 3. This is partly due to prominence of the Operating System type in the classification (Figure 2).

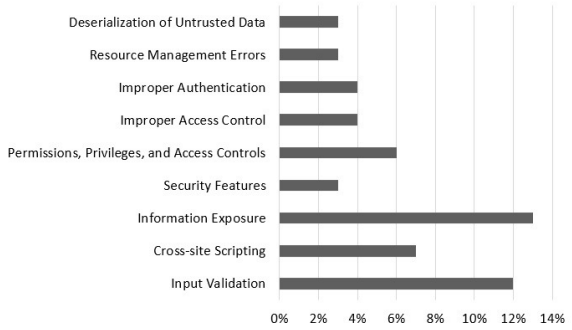

Fig. 6. Most common weaknesses for the Middleware type

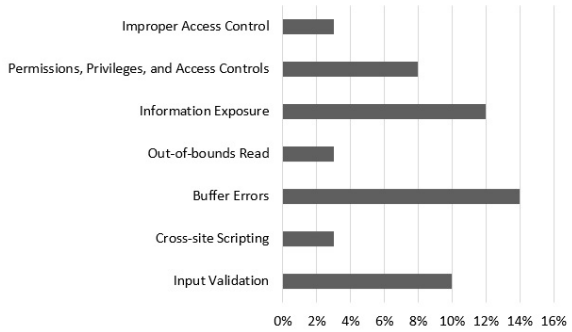

Fig. 7. Most common weaknesses for the Operating System type

Figure 8 shows the most common weaknesses for the Server product type. There are a total of 118 weaknesses across 5,840 vulnerabilities for this type. These 9 weaknesses account for 59.8% of all vulnerabilities. All of the most common weaknesses (Figure 3) feature prominently here, in addition to the SQL Injection and Path Traversal weaknesses. Figure 9 shows the most common weaknesses for the product type classified as Utility. There are a total of 124 weaknesses across 8,878 vulnerabilities for utilities. These 9 weaknesses account for 61% of all vulnerabilities.

Figure 10 shows the most common weaknesses for Web applications. For this product type, there are 105 weaknesses

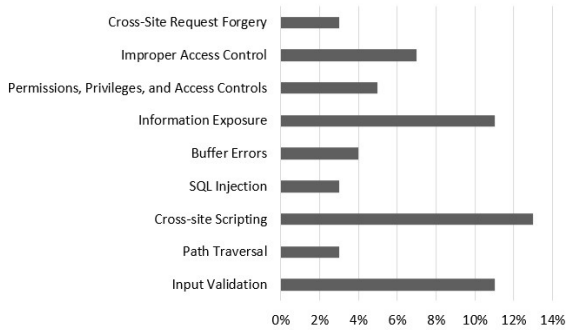

Fig. 8. Most common weaknesses for the Server type

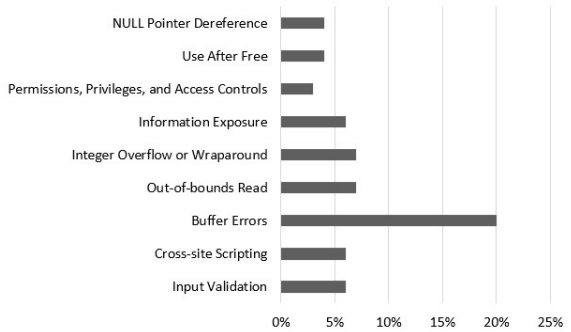

Fig. 9. Most common weaknesses for the Utility type

across 7,595 vulnerabilities. These 8 weaknesses account for 72% of all vulnerabilities for Web applications. We note that the most common weaknesses for the Web application type are similar to those of the Server type (Figure 8), allbeit at different degrees. The results show that the most common weaknesses (Figure 3) do not appear consistently across all software product types. Of the 7 weaknesses, only Input Validation, Cross-site Scripting, and Information Exposure occur at a high enough rate (3%) for every product type. The Buffer Error weakness met this threshold all types but Middleware. Buffer Error was recorded at 1.6% of all vulnerabilities for the Middleware type. Also, the rate at which each weakness occurs does vary greatly across product types.

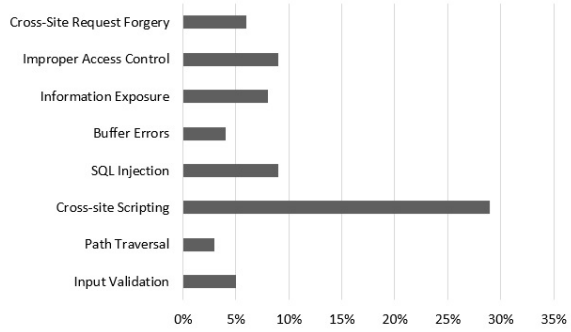

Fig. 10. Most common weaknesses for the Web Application type

B. How well do weaknesses persist over years for the same product?

We reviewed the data to determine whether the same weakness (E.g., Buffer Error) for a given product (E.g., Microsoft Windows 10) occurs again in subsequent years for the same product. Our results show that only a small percentage (less than 5%) of weaknesses repeat for the same product and the rate at which they reoccur does decrease over time (Figure 11). A reasonable assumption here is that a majority of the weaknesses get repaired. Also, over time, some products get discontinued or evolve into a different product. Products that have evolved into a different product (name) would not show up in the data as repeating. Figure 12 shows that Web applications account for the most occurrences of repeating weakness. It is important to note that the Operating System type which accounts for a high number of vulnerabilities (Figure 2) does not have as high a rate of repeating weaknesses, relatively. The results here highlight the importance of updating existing software installations.

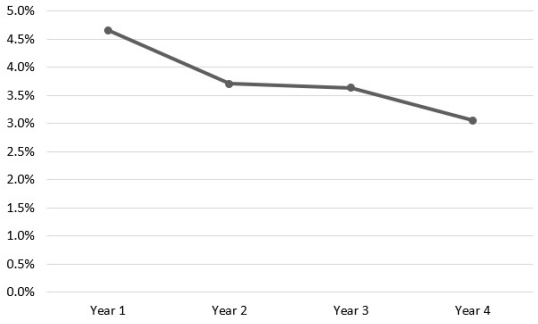

Fig. 11. Number of weaknesses that repeat for the same product over time

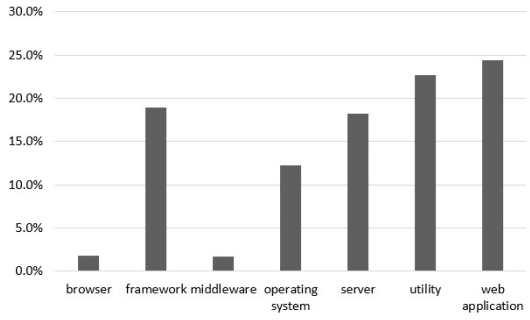

Fig. 12. Distribution of repeating weaknesses by software type

IV. Related Work

Some existing works have looked at analyzing vulnerability data from the NVD. None that we know of classify software weaknesses by product type. Santos et al. [8] devised a Common Architectural Weakness Enumeration as a means to catalog the common types of architectural weaknesses that generally exist in software. Na et al. [13] discuss a technique for analyzing vulnerability entries that do not have identified weaknesses. Their techniques attempts to identify the weakness using existing information in the vulnerability entry. Neuhaus et al. [14] analyzed vulnerability entries to identify trends in topics such as PHP and Format Strings that appear in the entries. Chang et al. [7] analyzes vulnerability entries from trends in the frequency and severity of vulnerability types. Others [4], [6], [10], [11], [15], [16] take a similar approach by analyzing existing texts for vulnerability trends.

V. Conclusion

To gain further insight into reported software security vulnerabilities, we analyzed 51,110 vulnerabilities from 2015 to 2019. We classify these vulnerabilities by software type. Our analysis shows that the occurrence of weaknesses across software types does vary greatly both in the type of weakness and rate of occurrence for each software type. We also show that the rate at which identified weaknesses repeat for the same product is significantly reduced over time. As we continue to analyze the data, we believe that our findings here will help inform approaches to the avoidance and detection of software weaknesses as well as inform strategies for software evolution and maintenance.

References

[1] The common weakness enumeration. https://nvd.nist.gov/vuln/categories, Retrieved February, 2020.

[2] National vulnerability database. https://nvd.nist.gov/, Retrieved Febraury, 2020.

[3] Relative vulnerability type totals by year. https://nvd.nist.gov/vuln/visualizations/cwe-over-time, Retrieved February, 2020.

[4] S. Alqahtani and J. Rilling. Semantic modeling approach for software vulnerabilities data sources. In *Proceedings of the 17th International Conference on Privacy, Security and Trust*. IEEE, 2019.

[5] E. Amoroso. Recent progress in software security. *IEEE Software*, 35(2), 2018.

[6] F. Bulut, H. Altunel, PMP, and A. Tosun. Predicting software vulnerabilities using topic modeling with issues. In *Proceedings of the 4th International Conference on Computer Science and Engineering*, 2019.

[7] Y.-Y. Chang, P. Zavarsky, R. Ruhl, and D. Lindskog. Trend analysis of the cve for software vulnerability management. In *Proceedings of the IEEE Third International Conference on Social Computing*, Oct 2011.

[8] J. C. da Silva Santos, K. Tarrit, and M. Mirakhorli. A catalog of security architecture weaknesses. In *Proceedings of the 2017 IEEE International Conference on Software Architecture Workshops*, 2017.

[9] A. Forward and T. Lethbridge. A taxonomy of software types to facilitate search and evidence-based software engineering. In *Proceedings of the 2008 Conference of the Center for Advanced Studies*, 2008.

[10] D. Gonzalez, H. Hastings, and M. Mirakhorli. Automated characterization of software vulnerabilities. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution*, 2019.

[11] T. H. M. Le, B. Sabir, and M. A. Babar. Automated software vulnerability assessment with concept drift. In *Proceedings of the IEEE/ACM 16th International Conference on Mining Software Repositories*, 2019.

[12] G. McGraw. Silver bullet talks with Ksenia Dmitrieva-Peguero. *IEEE Computing Edge*, February 2019.

[13] S. Na, T. Kim, and H. Kim. A study on the classification of common vulnerabilities and exposures using naïve bayes. In *Proceedings of Advances on Broad-Band Wireless Computing, Communication and Applications*. Springer, 2016.

[14] S. Neuhaus and T. Zimmermann. Security trend analysis with cve topic models. In *Proceedings of the 21st IEEE International Symposium on Software Reliability Engineering*. IEEE, November 2010.

[15] M. Williams, R. Camacho Barranco, S. M. Naim, S. Dey, M. Hossain, and M. Akbar. A vulnerability analysis and prediction framework. *Computers Security*, February 2020.

[16] X. Wu, W. Zheng, X. Chen, F. Wang, and D. Mu. CVE-assisted large-scale security bug report dataset construction method. *Journal of Systems and Software*, 160, 11 2019.