Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 16, no. 7, July 2020, pp. 1008-1018
DOI: 10.23940/ijpe.20.07.p3.10081018

Empirical Characterization of the Likelihood of Vulnerability Discovery

Carl Wilhjelm[a], Taslima Kotadiya[a], and Awad A. Younis[b,*]

[a]*Georgia State University, Atlanta, GA 30303, United States*
[b]*Northern Kentucky University, Highland Heights, KY 41099, United States*

Abstract

Assessing the risk of the likelihood of a vulnerability discovery is very important for decision-makers to prioritize which vulnerability should be investigated and fixed first. Currently, the likelihood of vulnerability discovery is being assessed based on expert opinion which could potentially hinder its accuracy. In this study, we propose using Time to Vulnerability Disclosure (TTVD) as a proxy for assessing the likelihood of vulnerability discovery. We will then empirically explore characterizing TTVD using intrinsic vulnerability attributes including CVSS Base metrics and vulnerabilities types. We examine 799 reported vulnerabilities of Chrome and 156 vulnerabilities of the Apache HTTP server. The results show that TTVD correlated at a statistically significant level to some of the intrinsic attributes, namely, access complexity metric, confidentiality, and integrity metrics, and the vulnerabilities' types. Our results from machine learning analysis also show ranges of TTVD values are associated with specific combined values of the metrics under consideration.

Keywords: likelihood of vulnerability discovery; CWSS; OWASP; software vulnerability; CVSS base score; vulnerability lifecycle; machine learning

(Submitted on June 5, 2020; Revised on July 9, 2020; Accepted on July 21, 2020)

1. Introduction

Assessing the risk presented by a software vulnerability is very important for decision-makers (testers, developers, managers, buyers, security vendors, and researchers) to prioritize their actions. Risk models such as the Common Weakness Scoring System (CWSS) [1] and Open Web Application Security Project (OWASP) [2] assess a vulnerability risk based on the likelihood of vulnerability discovery and exploitation and the impact. OWASP and CWSS subjectively estimate the likelihood of vulnerability discovery which could potentially hinder its accuracy. While more empirical research has been conducted on characterizing and modeling the likelihood of vulnerability exploitation [3-11] little work has been done to empirically characterize the likelihood of vulnerability discovery. The likelihood of vulnerability discovery is important when the vulnerability is only known to developers at the time of discovery and the developers may need to prioritize which vulnerability should be investigated and fixed first. A likelihood of vulnerability discovery is the likelihood that an attacker can discover the vulnerability [1].

To quantitatively assess the likelihood of vulnerability discovery, we propose using the time taken to discover a vulnerability as a proxy for the likelihood of vulnerability discovery as the time taken to discover a vulnerability depends on the likelihood of a vulnerability being discovered. The less time it takes to discover a vulnerability the higher the risk. Computing the time taken to discover a vulnerability can be accomplished by subtracting the vulnerability birth date from its discovery date. The actual discovery date, however, is not reported to the public. According to [6], few sources such as the Open Source Vulnerability Database (OSVDB) and the security bulletins of commercial vulnerability markets provide information, such as time of vendor notification and time of purchase, that can be used to derive the discovery date. However, those sources only contain the time a vulnerability is reported to the vendor or purchased and not the actual discovery date [12]. Besides, our source of the studied data, National Vulnerabilities Database [13], does not provide information about the vulnerability discovery date. This requires finding alternative approaches to find the date of discovery.

Frei et al. [5] stated that the vulnerability disclosure date implies its discovery. According to the Organization for Internet

* Corresponding author.
E-mail Address: mussaa1@nku.edu

Safety [14], the disclosure time can take 30 calendar days after the notification date. Besides, CERT (Computer Emergency Readiness Team) [15] has stated that vulnerabilities reported to them will be disclosed to the public 45 days after the initial report (regardless of the existence or availability of patches or workarounds from affected vendors). CERT also has stated that to remediate difficult vulnerabilities extension may be granted. In this research, we propose using the time a vulnerability is disclosed to the public as a proxy for the date of discovery. Thus, based on the vulnerability lifecycle proposed by Arbaugh et al. [16], we will use the time taken from when the software version containing the vulnerability is first released (birthdate) until the time a vulnerability is disclosed (disclosure date) to the public (Time To Vulnerability Disclosure (TTVD)) to represent the likelihood of vulnerability discovery. TTVD is measured by subtracting the first affected software release date (birthdate) from the vulnerability disclosure date, which is the date on which the vulnerability is first released to the public [6].

In this research, our objective is to investigate whether TTVD is related to intrinsic attributes of a vulnerability namely the CVSS Base score [17] and vulnerability types. The Base score captures the intrinsic and fundamental characteristics of a vulnerability. Here, we use the overall Base score values and the values of the individual metrics of the Base score. The Base score individual metrics include the exploitability and impact metrics. The exploitability metrics are access vector, access complexity, and authentication whereas the impact metrics are confidentiality (CI), integrity (II), and availability (AI) metrics. We will also identify vulnerability types for every vulnerability using the Common Weakness Enumeration (CWE) [18] provided by the NVD. Hafiz and Fang [19] showed that some types of vulnerabilities are harder to detect and require more effort than others. We postulate the following hypotheses on how TTVD is related to the CVSS Base score and vulnerability types. If the hypothesized relationships can be empirically validated, this information can provide useful inputs to security risk assessment and modeling the likelihood of vulnerability discovery studies.

- Hypothesis 1: A vulnerability that has an overall high CVSS Base severity has a short TTVD.
- Hypothesis 2: A vulnerability that has an overall high exploitability or impact values has a short TTVD.
- Hypothesis 3: A vulnerability with a low access complexity or with a high value of the confidentiality, integrity, or availability impact has a significantly short TTVD.
- Hypothesis 4: There is a combination of metrics that correlate with TTVD.

To validate the hypotheses, we conduct a case study on vulnerability data collected from Chrome (799 reported vulnerabilities) and the Apache HTTP server (156 vulnerabilities) [13]. In the case study, we first compute TTVD which requires determining the first affected software release date (birth time) and the vulnerability disclosure date (disclosure time). To determine the date of the affected version, we used the release date information found in [20-21] for Chrome and in [22], [23] for Apache HTTP server, whereas the disclosure date was obtained for Chrome and Apache HTTP server from [13]. Then, we analyze how the individual and the combined CVSS Base metrics and vulnerabilities types correlate with TTVD using the Spearman correlation and machine learning models such as decision trees, neural networks, support vector regressions, relevance vector machines, and Gaussian processes. The results show that the likelihood of a vulnerability discovery can be characterized by some of the intrinsic attributes, namely, access complexity metric, confidentiality, and integrity metrics as well as to the vulnerabilities' types. Our results from machine learning analysis also show that ranges of TTVD values are associated with specific combined values of the metrics under consideration.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 reports the results of the case study and discusses the implications of the results. Finally, Section 4 presents the discussion whereas section 5 concludes the paper and outlines some avenues for future work.

2. Related Work

Schneier [24] has identified the window of exposure that vulnerability presents when it exists in software. Arbaugh et al. [16], have suggested a vulnerability lifecycle model and using three vulnerabilities, they measured the number of intrusions during the vulnerability lifecycle. Frei et al. [5-6], extended Arbaugh et al.'s model and using more than 80,000 vulnerabilities, they identified and measured three types of risk exposures (black, gray, and white). Muegge and Murshed in [25] attempted to measure the time to discover and fix vulnerabilities in open source software projects using release dates and versions affected. However, none of these previous works consider investigating or measuring TTVD and relating this metric to the likelihood of vulnerability discovery.

McQueen et al. [12] have defined 0-Day vulnerability and analyzed its lifespans. They have defined the 0-Day lifespan as the time from the vulnerability discovery date to the public disclosure date. They were only able to identify the actual vulnerability discovery dates for 15 vulnerabilities that were provided to them by a researcher. They also compared the CVSS Base score to the mean lifespan. However, in this study, we consider TTVD starting from the vulnerability birth date and we do not only correlate TTVD with CVSS Base score, but also with the individual CVSS Base metrics and the types of vulnerabilities. We also study the effect of the vulnerability reward programs on the TTVD. Joh and Malaiya [26] formally defined a risk measure and utilized the vulnerability lifecycle and applied the Markov stochastic model to measure the likelihood of vulnerability exploitability. They also used the impact-related metrics of CVSS to estimate the exploitability impact. However, Joh and Malaiya did not take into consideration the TTVD.

OWASP [2] and CWSS [1] assess the likelihood of vulnerability discovery subjectively based on expert opinion. In this research, we propose characterizing the likelihood of vulnerability discovery empirically using TTVD, CVSS Base Score, and vulnerability types. Sommestad et al. [27], and Holm et al. [28], studied the effort required to discover a software vulnerability. They used an expert estimate to assess the vulnerability discovery effort using Cook's classical method. Finifter et al. [29] examined the characteristics of Google Chrome and Mozilla Firefox's vulnerability reward programs (VRPs). The authors found that VRPs improved the likelihood of finding latent vulnerabilities. Younis et al. [30], used VRPs as ground truth to assess the CVSS Base metrics. In this study, however, we study VRP data of Google Chrome to have an insight into the discoverers' effort and skills and study their relationship with TTVD.

3. Case Study

The purpose of the case study presented in this section is to test the experimental hypotheses we postulated about the relationships between TTVD and CVSS Base Score and vulnerability types. We have conducted a case study on Google Chrome and the Apache HTTP Server. The two software systems have been selected because of their rich history of publicly available vulnerabilities, the existence of an integrated repository (which enables us to find the release date information and the disclosure date), the use (Chrome) and the lack of use (Apache HTTP Server) of the vulnerability reward program (VRP), and their diversity in size, functionalities, and domain. The VRP data for Google Chrome has been used so an insight into the effects of efforts and skills on the TTVD may be provided. Besides, to examine the effects when the VRP is not used on TTVD, we consider the Apache HTTP server data.

3.1. Data Collection

In this study, 799 reported vulnerabilities of Google Chrome during the period 2007 to February 2016 and 156 reported vulnerabilities of the Apache HTTP server during the period 1999 to February 2016 were collected from NVD [13] as shown in Table 1. It should be noted that we were not able to find the release date for 369 vulnerabilities of Chrome. Besides, we could not fund rewards information for 72 vulnerabilities of Chrome because of the unauthorized access permission, "you are not authorized to access this data."We could not find the release date and version information for 27 vulnerabilities of the Apache HTTP server. Furthermore, as there are differences between some of the metrics inthe CVSS Base score version 2 and 3, we only collected CVSS data for version 2, which were used until February 2016 and after that, CVSS Base score version 3 has been put in use.

Table 1. Chrome and apache HTTP server vulnerabilities

Software	Total # of Vulnerabilities	Period	Examined Vulnerabilities
Google	1240	2007 - Feb 2016	799
Apache	183	1999 - Feb 2016	156

3.2. Computing Time-to-Vulnerability-Disclosure (TTVD)

TTVD is the time taken from when the software has first been released until the time a vulnerability is disclosed to the public. Our approach to measuring TTVD is based on the vulnerability lifecycle. A vulnerability is created as a result of a coding or specification mistake. A vulnerability lifecycle is a model that describes the states that vulnerability can enter during its lifetime [16]. Figure 1 shows vulnerability lifecycle states: birth, discovery, and discovery may be followed by any of the following: internal disclosure, patch, exploit/script, or public disclosure. More details about the vulnerability lifecycle can be found in [6, 16]. The birth of the vulnerability is defined to start after the software is released or deployed. TTVD can be measured as shown below:

$$TTVD = \textit{Vulnerability Disclosure Date} - \textit{Release Date of the First Affected Version}$$

Therefore, measuring TTVD requires determining the first affected software release date and the vulnerability disclosure date.

Software Release Date: Frei et al. [6] indicated that the time of a vulnerability birth can be determined by looking back at the history of the vulnerability after its discovery or disclosure. However, the authors did not consider that. In this work, we determine the vulnerability birth date and use it to measure TTVD. The vulnerability birth date is determined by finding the release date of the first affected software version. To find the release date, we first find the affected software versions. After that, we determine the release date of the first affected version. The National vulnerability DataBase (NVD) [13] records the version information of the software that is affected by a vulnerability. We have noticed, however, that some vulnerabilities do not have version information. Table 2 shows a list of Chrome versions affected by the CVE-2014-1702 vulnerability provided by NVD. Due to the page limitation, we only show 3 affected versions.

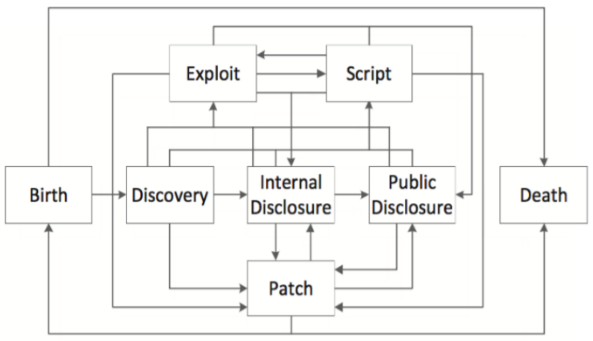


Figure 1. Vulnerability Lifecycle

Table 2. Products affected by CVE-2014-1702

#	Vendor	Product	Version
1	Google	Chrome	33.0.1750.0
2	Google	Chrome	33.0.1750.38
3	Google	Chrome	33.0.1750.146

After sorting this list, we have found out that the Chrome version 33.0.1750.0 is the first affected one. However, until the date of this paper, NVD does not provide information about the date of the affected software versions. To determine the date of the affected version, we used the release date information found in [20-21] for Chrome, and in [22-23] for the Apache HTTP server. Table 3 shows some samples of how the date of the affected software has been determined.

Table 3. First affected software release date

Vulnerability	Software version	Release date
CVE-2011-2170	11.0.696.65	5/6/2011
CVE-2012-2819	20.0.1132.0	6/26/2012
CVE-2013-0926	26.0.1410.0	3/26/2013
CVE-2014-1731	34.0.1847.130	4/24/2014
CVE-2015-1233	41.0.2272.102	3/24/2015

Vulnerability Disclosure Date: Vulnerability disclosure date (VDD) is the date on which the vulnerability is first released to the public [16]. According to the Organization for Internet Safety [10], the disclosure time can take 30 calendar days after the notification date. Besides, CERT (Computer Emergency Readiness Team) [3] has stated that vulnerabilities reported to them will be disclosed to the public 45 days after the initial report (regardless of the existence or availability of patches or workarounds from affected vendors). CERT also has stated that for difficult vulnerabilities to remediate, an extension may be granted. Besides, Google [12] suggests that 60 days is a reasonable upper bound for genuinely critical issues in widely deployed software. In most of the cases, the disclosure date will be around 60 days after the vendor notification date.

Computing TTVD: based on the data collected in a and b, Table 4 shows how some of the TTVD has been computed for both Chrome and Apache HTTP server using the YEARFRAC function in [31]. Due to the page limitation we only show four cases. As can be seen, there is a noticeable difference between the TTVD for Apache HTTP and Chrome. TTVD can be influenced by extrinsic factors such as discoverers' skills and efforts and/or intrinsic attributes of vulnerability and types. Zhao et al. [26], found a significant positive correlation between the monetary incentive and the number of vulnerabilities reported. To capture the effects of the discoverers' skills and effort, we use a vulnerabilities rewards program (VRP). VRPs are programs adopted by software vendors to pay security researchers, ethical hackers, and enthusiasts for the exchange of discovering vulnerabilities in their software [15].

Figure 2 shows the TTVD for Apache and Chrome. It should be noted that Chrome has been using a vulnerability reward program (VRP) since 2010. The effect of the VRP is noticeable on TTVD. The mean for Apache TTVD is 4.512 and the maximum TTVD is 17.570, whereas the mean TTVD for Chrome is 0.2438 (around three months, 0.2438 * 12(months)) and the maximum TTVD is 3.6580. Due to the difference in the release dates (Apache in 1998 and Chrome in 2008) and the date the Chrome starts using the VRP, we also compared the TTVD of the two datasets only from 2010 onward. The mean for Apache TTVD is 7.157 and the maximum TTVD is 13.230, whereas for Chrome the mean and the maximum TTVD are 0.2173 and 3.6580 respectively.

Table 4. Computed TTVD

CVE	Vulnerability release date	Software version	Software release date	TTVD
CVE-1999-1293	12/31/99	Apache 1.2.5	03/07/1995	4.817
CVE-2011-3348	09/20/2011	Apache 0.8.11	12/28/1995	15.728
CVE-2008-7246	09/18/2009	Chrome 0.2.149.27	09/02/2008	1.044
CVE-2010-1665	05/03/2010	Chrome 0.2.149.27	09/02/2008	1.669

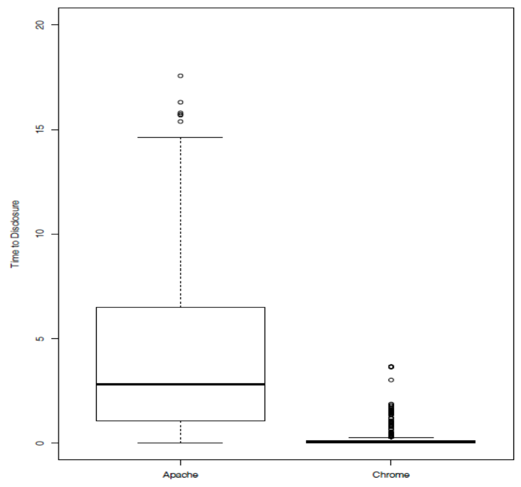


Figure 2. Comparing TTVD for Apache HTTP server and Chrome

3.3. Descriptive Statistics

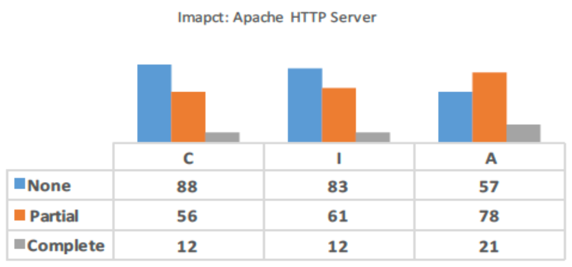
Table 5 shows a part of the attributes of the selected vulnerabilities (the first two vulnerabilities are for Apache and the second two are for Chrome). CVSS Base score is the severity score of a vulnerability. The exploitability metrics are access vector (AV), access complexity (AC), and authentication (AU). The impact metrics are confidentiality (C), integrity (I), and availability (AV). The CWE assigns a number for the vulnerabilities' type. Looking at the CVSS Base score data and the access complexity, we have found that the access vector and authentication values are almost constant for Chrome and Apache and thus, we did not show their descriptions. For Chrome, only four vulnerabilities out of 799 require local access and the rest requires network access whereas one vulnerability requires a single authentication. For Apache, 23 vulnerabilities out of 156 require local access and the remaining requires network access whereas only three vulnerabilities require a single authentication.

Table 5. The attributes of the examined vulnerabilities

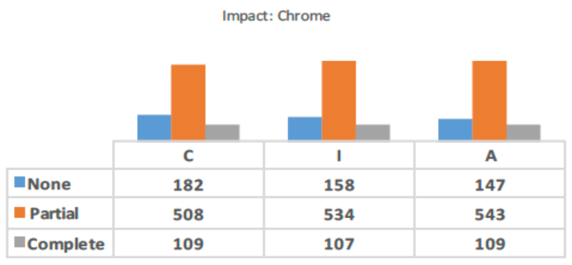
CVE	CVSS-Bases Score	Exploitability	Impact	AV	AC	AU	C	I	A	CWE	Reward
CVE-1999-1293	10	10	10	NW	L	NR	Complete	Complete	Complete	399	N/A
CVE-2011-3348	4.3	8.6	2.9	NW	M	NR	None	None	Partial	Not Given	N/A
CVE-2008-7246	5	10	2.9	NW	L	NR	None	None	Partial	399	NRW
CVE-2010-1665	7.5	10	6.4	NW	L	NR	Partial	Partial	Partial	199	$500
NW: Network, L: Low, NR: Not Required, NRW: Not Rewarded, N/A: Not Applicable											

We have also observed that there are more medium severity vulnerabilities for Apache and almost the same number of medium and high severity vulnerabilities for Chrome. Besides, for both softwares, there is more ease to access vulnerabilities (low access complexity) and few hard to access vulnerabilities (high access complexity). We have noticed that when the access complexity is low, the low and high impact percentage for both softwares is almost the same whereas the medium impact percentage for Chrome is 64.63% compared to 22.22% for Apache. In addition, when the access complexity is medium, the low impact percentage for Apache is about 68.75% whereas in Chrome, the low and medium impact percentage is almost the same and the high impact percentage is about 18.14% compared to 4.6% in Apache.

As seen in Figure 3, there are more partial impact values for Chrome vulnerabilities than none values, whereas there is almost the same number of partial and none values for Apache vulnerabilities. However, we have also noticed that the majority of the exploitability values for both datasets are high (Apache:27 are medium or low, Chrome: 10 are medium or low).



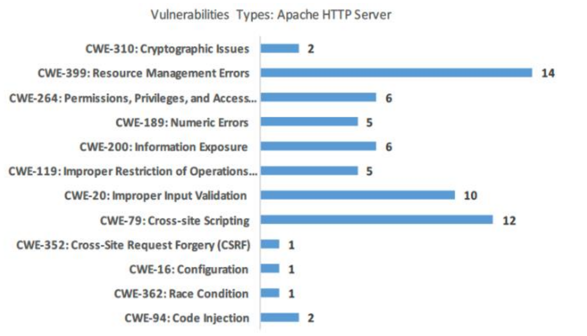
	C	I	A
🟦 None	88	83	57
🟧 Partial	56	61	78
⬜ Complete	12	12	21



	C	I	A
🟦 None	182	158	147
🟧 Partial	508	534	543
⬜ Complete	109	107	109

Figure 3. Impact metrics values for C, I, and A

Figure 4 shows the vulnerability types for both datasets. Only 41.66% (65 out of 156) of the Apache vulnerabilities have been assigned CWE type by NVD, whereas only 75.46% (603 out of 799) of the Chrome vulnerabilities have a CWE number assigned. We have noticed that all vulnerability types of Apache datasets are similar to the vulnerability types of Chrome. On the other hand, there are 19 different types of vulnerabilities in Chrome. It should be noted that the types in Figure. 4 are ordered from bottom to top. The following vulnerability types are just found in Chrome: CWE-17: Code, CWE-19: Data handling, CWE-22: Path traversal, CWE-59: Link following, CWE-254: Security feature, CWE-2 87: Improper authentication, and CWE-289: Improper access control. However, we have noticed that the vulnerability types CWE-399, CWE-20, CWE-264, CWE-119, CWE-189, and CWE-79 are the commonly found types in both datasets.



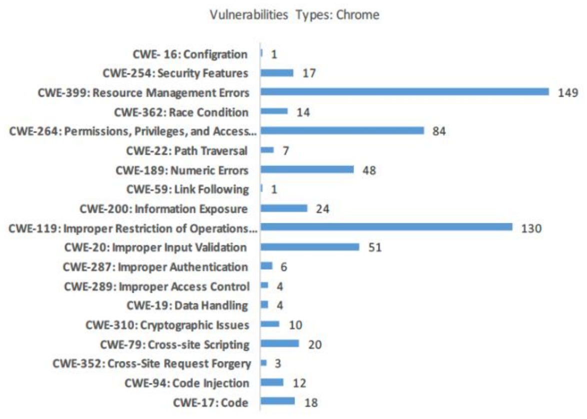


Figure 4. Vulnerabilities types and their CWE numbers

Figure 5 shows the relationship between every vulnerability type and TTVD. The vulnerabilities types of Chrome are numbered from 1 to 19 based on the order in Figure. 4 from bottom to top. Looking at the Chrome Boxplot, we can see that vulnerability types (1, 3, 7, 8, 10, 11, and 13 to 19) are found in a very short time. On the other hand, the other 6 types (2, 4, 5, 6, 9, and 12) have more noticeable TTVD variability. Figure 5 also shows the relationship between every vulnerability type and TTVD of Apache HTTP server dataset. The vulnerability types of Apache are also numbered based on the order in Figure 4 from bottom to top. Vulnerabilities of types 1 to 4 have been found in a relatively short time compared to the others, whereas vulnerabilities of type 7 and 12 have the most TTVD variability. We have noticed that the same vulnerability types take more TTVD in Apache take less TTVD in Chrome.

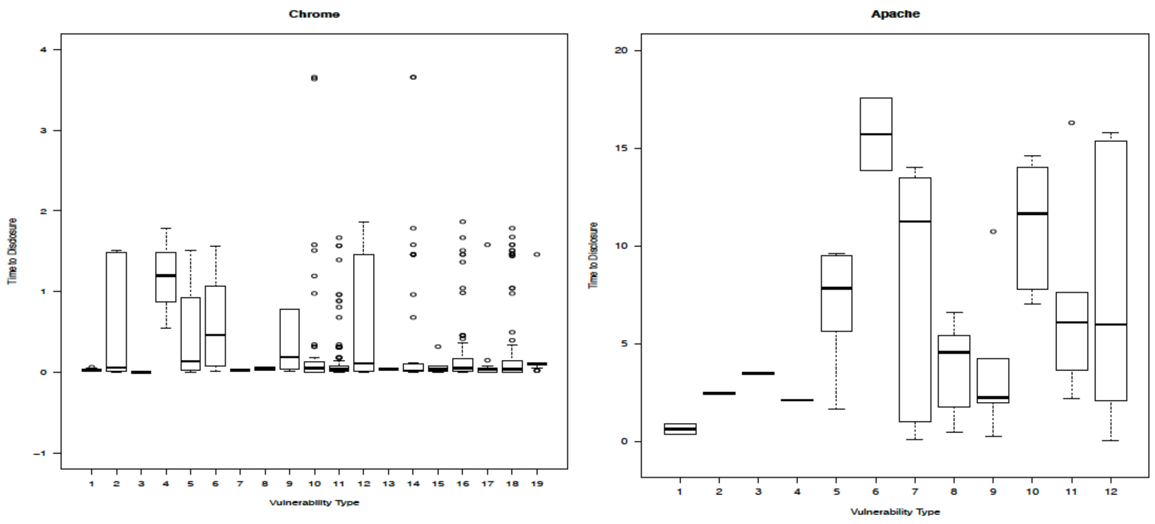


Figure 5. Chrome and Apache HTTP server vulnerabilities types and TTVD

3.4. Testing the Hypotheses

In this subsection, we first assess the individual relationships of CVSS Base metrics and types with TTVD using Spearman correlation. The reason we have chosen to use Spearman is that one variable (TTVD) is continuous whereas the other variables are either on an ordinal or categorical scale. The result of the correlation measurement is represented by a r and p-value. The r-value represents the strength of the correlation whereas the p-value shows the significance of the correlation. The correlation is considered significant if the p-value is less than 0.05. We then use machine learning models such as decision trees, neural networks, support vector regressions, relevance vector machines, and Gaussian processes to examine the selected attributes relationships when they are combined. The decision trees have been selected because they provide information about each factor's contribution to predictions and this helps to interpret a tree to make different levels of predictions. We have chosen the other regression models because they provide a relatively higher predictive accuracy with nonlinear approximation. After that, we examine the relationships between the vulnerability types and TTVD and analyze the effects of the VRP on the TTVD.

3.4.1. Hypothesis 1

A vulnerability that has an overall high CVSS Base severity has a short TTVD. We expect to find the vulnerabilities that have a high CVSS Base score severity (easy to exploit and has a severe exploitation impact) to have a significantly short TTVD with a p-value less than 0.05. The results show that the overall Base score values do not have a significant correlation with TTVD for both datasets: Apache (r = -0.085, p = 0.288) Chrome (r = 0.007, p = .834). We have also looked at the differences in the means of the Base score and the TTDV. The results show no noticeable difference: Apache (mean when the score is L = 3.275, M = 4.674, *High* = 4.484), Chrome (mean: L = 0.6243, M = 0.2356, *High* = 0.2465).

3.4.2. Hypothesis 2

A vulnerability that has an overall high exploitability or impact values has a short TTVD. We also examined the correlation between the exploitability and the impact of individual values and TTVD. We expect to find a vulnerability with a high exploitability or impact values to have a significantly short TTVD. The overall impact values have no significant correlation with TTVD, Apache (r = 0.112, p = 0.165), Chrome (r = 0.02557355, p = 0.4704). On the other hand, the results show a significant negative correlation between the exploitability and TTVD for both software: Apache (r = -0.2711113, p = 0.000619), Chrome (r = -0.163, p = 0.00000369). The negativity explains that the higher the exploitability value, the easier the exploitation of the vulnerability.

3.4.3. Hypothesis 3

A vulnerability with a low access complexity or with a partial or complete value of the confidentiality, integrity, or availability is likely to have a significantly short TTVD. After looking at the exploitability factor data, we have found that the authentication and access vector metrics are constant whereas the access complexity shows variability. Therefore, we only examine the correlation between access complexity and TTVD. We expect to find a vulnerability with a low access complexity to have a significantly short TTVD. The results show a significant positive correlation between the access complexity and TTVD for both softwares: Apache (r = 0.307, p = 0.0000973) and Chrome (r = 0.1767618, p = 0.000000495). We also examined the correlation between the impact factor metrics and TTVD. We expect to find a vulnerability with a partial or complete value of the confidentiality, integrity, or availability to have a significantly short TTVD. The results show a significant positive correlation between the confidentiality and TTVD for the Chrome dataset (r = 0.0729, p = 0.000000495) and a significant positive correlation between the integrity and TTVD for the Apache dataset (r = 0.1826154, p = 0.0.0225).

3.4.4. Hypothesis 4

There is a combination of metrics that correlate with TTVD. To examine the correlation between the considered attributes and TTVD, we apply the regression models mentioned above. The regression test uses eight features: CVSS Base score, access vector, access complexity, authentication, impact on confidentiality, integrity, and availability, and the type of vulnerabilities. Each categorical input is transformed to corresponding discrete numerical values for mathematical regression models. It should be also noted that there are overlaps on the inputs (109 unique inputs out of 799 samples in the Chrome dataset and 63 unique inputs out of 156 samples in the Apache dataset). In other words, there are multiple outputs (TTVD) for the same inputs (the selected features). Table 6 shows the 10-fold cross-validation result on the Chrome and Apache data sets. The decision tree and Gaussian process show the best performance for both datasets. Interestingly, a simple decision tree (CART) can make an equivalent prediction to other algorithms. As can be seen from the residual plots in Figure 6, the large error is produced because of the outliers and it captures the majority of points with mean prediction over evenly distributed residual space.

Table 6. The best training RMSE with each algorithm

	Tree	NNet	RVM	SVR	GP
Chrome	12.09	12.58	12.3	12.81	12.38
Apache	33.56	37.76	35.74	37.4	31.94

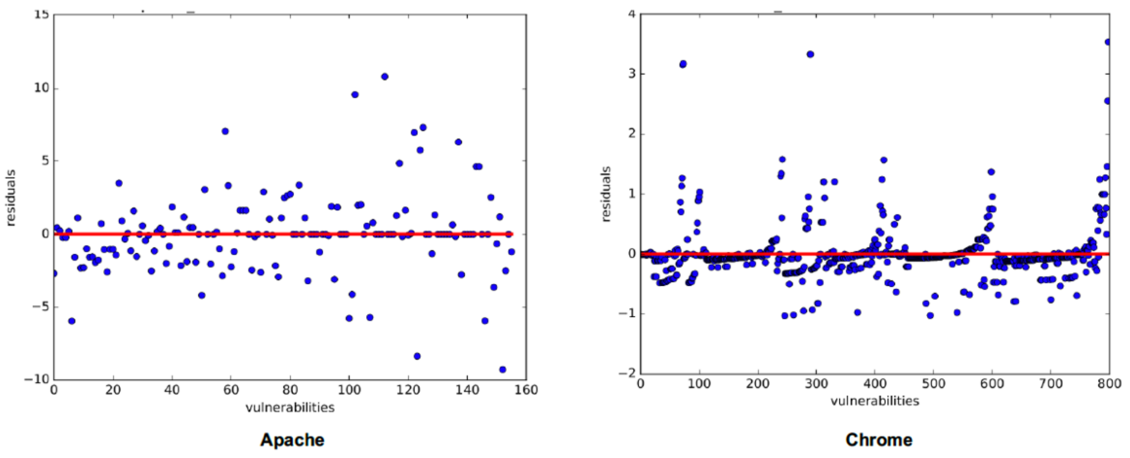


Figure 6. Residual plots of the decision tree fit on Chrome and Apache datasets

Figure 7 shows the generated decision trees that give us further reasoning analysis. First, the decision tree shows us that trees are split majorly by the CVSS Base score and the type of vulnerabilities with a bit contribution from access complexity and integrity impact. Interestingly, when the CVSS Base score was considered individually, it did not show a significant correlation. The Chrome TTVD estimation tree uses the Base score as the root to determine the TTVD while the Apache estimation tree uses the vulnerability type as a root node. It should be noted that encountering the type node means less or greater in the order of vulnerabilities shown in Figure 4. Based on the CEW number, the type in the decision tree can be determined.

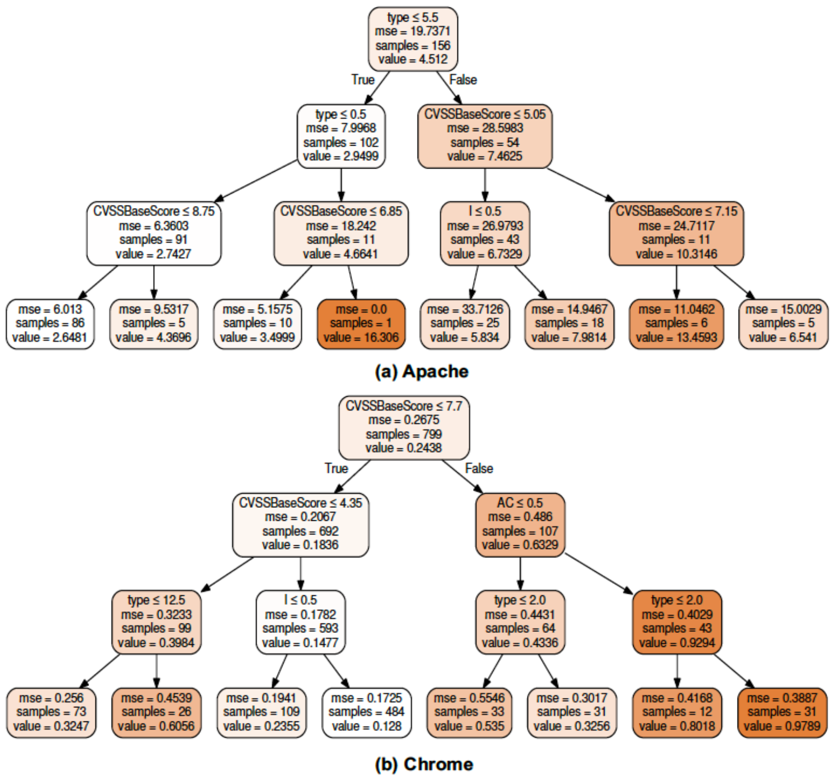


Figure 7. Top 3 levels of the decision trees

Figure 8 shows the neural network and its weights. The row index from 0 to 7 represents the features CVSS base score, access vector (AV), access complexity (AC), authentication (AU), impact on confidentiality (C), impact on integrity (I), impact on availability (A), and the type of vulnerabilities respectively. The columns 2 and 5 represent corresponding hidden units. We observe similar factoring from neural network weights. In Chrome data, the weights for the CVSS Base score (first row in the image) have a higher value than the other attributes and the weights for the vulnerability type have a relatively strong value. Here, we observe higher weight values in the 1st and 4th hidden units. Although it is not presented here, the output layer enforces more on the 2nd and 3rd layer. Thus, eventually, the Base score contributes more to the final prediction.

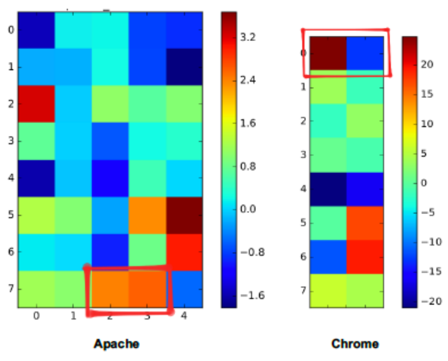


Figure 8. First layer neural network weights

3.5. Threats to Validity

In this paper, we have considered the datasets for only two products: the Apache HTTP server, and Google Chrome. However, they are both very significant examples. The Apache HTTP server has 183 vulnerabilities belonging to different categories. The lines of the code vary between 50,712 LOC to 358,633 LOC. Chrome has a larger number of vulnerabilities, more than 1200. Its size in the lines of the code is 4,490,488 LOC. It also has a greater variety of vulnerabilities. We are aware that the vulnerabilities release date provided by NVD does not reflect the actual disclosure date to the public. We are not aware of any other data sources that provide reliable and complete data on the actual disclosure or the discovery date.

4. Discussion

We have noticed that there are more types of vulnerabilities in Chrome than in the Apache HTTP server. One possibility could be because of the VRP. As more eyes are in the code, there is a higher chance that not only the number of the vulnerabilities will be discovered, but also the varieties of types. However, further research is required to investigate this observation. We have also observed that a specific type of vulnerability has a shorter TTVD. Looking at the CEW detailed description found in [18], we have found that most of these vulnerabilities have either more than one method of detection or have a medium or a high likelihood of exploitability (provided by CWSS measure [1]). Besides, some vulnerability types have a longer TTVD. We have realized that these vulnerabilities are very popular, so it could be that the developers or testers are aware of them and they discover most of them during the development phase.

We have also observed that the CVSS Base scores did not correlate with the TTVD when considered individually. The authors in [12] have observed the same thing when they compare the 0-Day vulnerability lifespan with the CVSS Base score. However, when the CVSS score was combined with other correlated attributes such as types, access complexity, and some impact metrics, CVSS Scores have been found indicative. According to Forman [32], a feature (or two features) that is completely useless by itself (themselves) can be useful when taken with others (together). We have also observed that the rewarded vulnerabilities have smaller TTVD. This could be explained by the fact that VRPs have a larger and diverse group of security researchers and contributors that outperformed the internal security teams or penetration testing teams [33].

5. Conclusion and Future Work

In this paper, we have empirically explored characterizing the likelihood of vulnerability discovery using vulnerability lifecycle events, namely the time of birth and disclosure (TTVD) and intrinsic vulnerability attributes including CVSS Base metrics and vulnerabilities types. The results show that the likelihood of vulnerability discovery can be characterized by some individual CVSS Base metrics attributes including access complexity metric, confidentiality, and integrity metrics as well as to the vulnerabilities' types. Machine learning analysis shows that relying on measuring the correlation of the individual attributes alone is a good thing to be performed first, however, combining several attributes together could provide insightful results.

The results have also opened up an avenue for future work. First, the NVD should enhance its data about version information and release dates. The recommendation provided by Glanz et al. [34] should be taken into consideration by the NVD. Further, even though the vulnerabilities types have shown great insight, finding a way to order them in a particular manner is very important for assessing the risk of vulnerability disclosure and discovery. Furthermore, looking for other attributes that can be related to the TTVD is at the top of the list of our priority.

References

1. B. Martin, “Common Weakness Scoring System (CWSS),” The Mitre Corporation, June 2011
2. OWASP Risk Rating Methodology, (https://owasp.org/www-community/OWASP_Risk_Rating_Methodology, accessed May 20 2020)
3. A. Younis, Y. K. Malaiya, and I. Ray, “Assessing Vulnerability Exploitability Risk using Software Properties,” *Software Quality Journal*, Vol. 24, No. 1, pp. 159-202, DOI: 10.1007/s11219-015-9274-6, March 2016
4. M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, “Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 105-113, DOI: 10.1145/1835804.1835821, 2010
5. S. F. Accenture, B. P. E. Zurich, and B. T. E. Zurich, “Modeling the Security Ecosystem-The Dynamics of (In)Security PRIvacy-Aware Secure Monitoring (PRISM) View Project BETEUS View Project,” *Springer*, pp. 79-106, DOI: 10.1007/978-1-4419-6967-5_6, 2010
6. S. Frei, M. May, U. Fiedler, and B. Plattner, “Large-Scale Vulnerability Analysis,” in *Proceedings of the 2006 SIGCOMM Workshop on Large-scale Attack Defense, LSAD'06*, Vol. 2006, pp. 131-138, DOI: 10.1145/1162666.1162671, 2006
7. L. Allodi and F. Massacci, “A Preliminary Analysis of Vulnerability Scores for Attacks in Wild: The EKITS and SYM Datasets,” in *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 17-24, DOI: 10.1145/2382416.2382427, 2012
8. K. Nayak, D. Marino, P. Efstathopoulos, and T. Dumitraş, “Some Vulnerabilities are Different than Others: Studying Vulnerabilities and Attack Surfaces in the Wild,” in *Lecture Notes in Computer Science* (*Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*), Vol. 8688 LNCS, pp. 426-446, DOI: 10.1007/978-3-319-11379-1_21, 2014
9. C. Sabottke, O. Suciu, T. Dumitraş, and T. Dumitras, “Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits,” in *Proceedings of the 24th USENIX Conference on Security Symposium*, pp. 1041-1056, August 2015
10. A. Younis, Y. K. Malaiya, C. Anderson, and I. Ray, “To Fear or Not to Fear that is the Question: Code Characteristics of a

Vulnerable Function with an Existing Exploit," in *Proceedings of the 6th ACM Conference on Data and Application Security and Privacy*, pp. 97-104, DOI: 10.1145/2857705.2857750, March 2016

11. A. Younis, Y. K. Malaiya, and I. Ray, "Using Attack Surface Entry Points and Reachability Analysis to Assess the Risk of Software Vulnerability Exploitability," in *Proceedings of 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering, HASE 2014*, DOI: 10.1109/HASE.2014.10, 2014
12. M. McQueen, T. McQueen, W. Boyer, and M. Chaffin, "Empirical Estimates and Observations of 0day Vulnerabilities," (https://ieeexplore.ieee.org/abstract/document/4755605/?casa_token=gf4z5-32oO0AAAAA:6pl3f2yzMR9fGaYm0ap_lXafVqQCCO4qNiIWl9qzBhxdaEBk2MyATwANDYDzD_LT0hfea8AshQ, accessed January 2009)
13. NVD - Home, (https://nvd.nist.gov/, accessed May 21 2020)
14. O. S. -S, "Guidelines for Security Vulnerability Reporting and Response. c2004," (http://www.oisafety. org/guidelines/Guidelines, accessed May 21 2020)
15. The CERT Division | Software Engineering Institute, (https://www.sei.cmu.edu/about/divisions/cert/index.cfm, accessed May 21 2020)
16. W. Arbaugh, W. Fithen, and J. McHugh, "Windows of Vulnerability: A Case Study Analysis," (https://ieeexplore.ieee.org /abstract/document/889093/?casa_token=Cp2JuRWLF5EAAAAA:7jNmY5s8n5WgsHYItCvV-vnjoWpaB_eOZxqYY-71gXesT6yn6Gw85MFKS04Lrd59s46PjPWUmg, accessed December 2000)
17. P. Mell, K. Scarfone, and S. Romanosky, "A Complete Guide to the Common Vulnerability Scoring System Version 2.0," (http://www.first.org/cvss/cvss-guide.pdf, accessed June 2007)
18. CWE - Common Weakness Enumeration, (https://cwe.mitre.org/, accessed May 21 2020)
19. M. Hafiz and M. Fang, "Game of Detections: How are Security Vulnerabilities Discovered in the Wild?" *Empirical Software Engineering*, Vol. 21, No. 5, pp. 1920-1959, DOI: 10.1007/s10664-015-9403 7, October 2016
20. Google Chrome Version History - Wikipedia, (https://en.wikipedia.org/wiki/Google_Chrome_version_history, accessed May 21 2020)
21. Chrome Releases, (https://chromereleases.googleblog.com/, accessed May 21 2020)
22. Welcome! - The Apache HTTP Server Project, (https://httpd.apache.org/, accessed May 21 2020)
23. Apache HTTP Server – Wikipedia, (https://en.wikipedia.org/wiki/Apache_HTTP_Server, accessed May 21, 2020)
24. S. -C. -G, Newsletter and undefined 2000, "Full Disclosure and the Window of Exposure," (https://www.mendeley.com /catalogue/fceceeb1-8021-30a1-aac6-0da5b105200b/, accessed June 2014)
25. S. Muegge and S. Murshed, "Time to Discover and Fix Software Vulnerabilities in Open Source Software Projects: Notes on Measurement and Data Availability," (https://ieeexplore.ieee.org/abstract/document/8481833/?casa_token=_AOGGP7YAnsAAAAA:Agnz012T8OxA1Dh7YIbuy_PcujWbWvkDst89Wdyo7ha-ftHXn9Y2ebP5Ccr_xRuD9TP-spJmHg, accessed October 2018)
26. H. Joh and Y. K. Malaiya, "Defining and Assessing Quantitative Security Risk Measures Using Vulnerability Lifecycle and CVSS Metrics," (http://www.cs.colostate.edu/~malaiya/p/johrisk11.pdf, accessed May 22 2020)
27. T. Sommestad, H. Holm, and M. Ekstedt, "Effort Estimates for Vulnerability Discovery Projects," (https://ieeexplore.ieee.org/ abstract/document/6149570/?casa_token=ohd5jKeIcGkAAAAA: oNn-H1sJjUJwmTo-Kea6RX47pomKJ-yQt0iZckT3uTnMFC9Tgin_rYQkXtJsWguIdhNMSZTRug, accessed February 2012)
28. H. Holm, M. Ekstedt, and T. Sommestad, "Effort Estimates on Web Application Vulnerability Discovery," (https://ieeexplore.ieee.org/abstract/document/6480453/?casa_token=7HDCaZgP05gAAAAA:P5pu2w0RwFa77WSSea1hgRu0UkwRE5BZhL9PLtqg0skzoi1sh0midPinDyN16Z2I3wdDQ1IDpA, accessed March 2013)
29. An Empirical Study of Vulnerability Rewards Programs - Google Scholar, (https://scholar.google.com/scholar?hl=en&as_sdt=0%2C11&q=An+empirical+study+of+vulnerability+rewards+programs&btnG=, accessed May 22 2020)
30. A. Younis, Y. K. Malaiya, and I. Ray, "Evaluating CVSS base Score using Vulnerability Rewards Programs," *ICT Systems Security and Privacy Protection*, Vol. 471, pp. 62-75, 2016
31. YEARFRAC Function - Office Support, (https://support.office.com/en-us/article/yearfrac-function-3844141e-c76d-4143-82b6-208454ddc6a8, accessed May 25 2020)
32. A. M. De Guyon, "An Introduction to Variable and Feature Selection André Elisseeff," (http://www.jmlr.org/papers/v3/ guyon03a.html, accessed 2003)
33. M. Zhao, J. Grossklags, and P. Liu, "An Empirical Study of Web Vulnerability Discovery Ecosystems," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1105-1117, DOI: 10.1145/2810103.2813704, October 2015
34. L. Glanz, S. Schmidt, S. Wollny, and B. Hermann, "A Vulnerability's Lifetime: Enhancing Version Information in CVE Databases," in *Proceedings of the 15th International Conference on Knowledge Technologies and Data-Driven Business*, No. 28, pp. 1-4, DOI: 10.1145/2809563.2809612, 2015