Analyses Of Two End-User Software Vulnerability Exposure Metrics (Extended Version)[☆]

Jason L. Wright[a,*], Miles McQueen[a], Lawrence Wellman[a]

[a]*Idaho National Laboratory, PO Box 2315, Idaho Falls, Idaho, USA 83415-3544*

Abstract

Understanding the exposure risk of software vulnerabilities is an important part of the software ecosystem. Reliable software vulnerability metrics allow end-users to make informed decisions regarding the risk posed by the choice of one software package versus another. In this article, we develop and analyze two new security metrics: median active vulnerabilities (MAV) and vulnerability free days (VFD). Both metrics take into account both the rate of vulnerability discovery and the rate at which vendors produce corresponding patches. We examine how our metrics are computed from publicly available data sets and then demonstrate their use in a case study with various vendors and products. Finally, we discuss the use of the metrics by various software stakeholders and how end-users can benefit from their use.

1. Introduction

Every week new software vulnerabilities are discovered in many applications and patches are issued fixing previously discovered vulnerabilities. Various measurements of this effect have been proposed, but comparisons between similar products from different vendors or different products with the same vendor have been difficult. This article, an extended and revised version of the paper Wright et al. (2012), proposes two new end-user focused metrics that allow for cross product or cross vendor comparison. The metrics are based on measurements of the number and rate of vulnerability reports, and the patch development rate for individual software products. These measurements are related to events which are part of the vulnerability life cycle.

To quantitatively characterize the time between events in the vulnerability life cycle model, depicted in Figure 1 and fully described in Arora et al. (2008), ideally one would measure the time from discovery of a flaw until the time all end-user machines have been patched to address the issue. In practice, this has been demonstrated to be difficult since the times and dates for most events along the life cycle are not credibly and verifiably known.

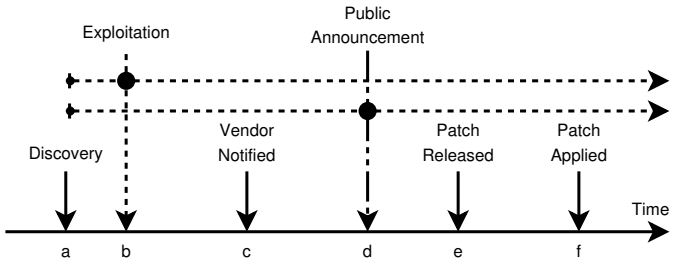


Figure 1: Vulnerability lifetime model

For instance, it is difficult to accurately record the initial discovery of a vulnerability (a), even for a discoverer because it is possible the vulnerability has been independently discovered by another party (Rescorla, 2005). On the other end of the life cycle, it has been shown that applying security patches (f) involves a half-life behavior and finally tapers off at approximately 5–10% of machines that will remain unpatched (Kandek, 2009).

In practice, we can measure the time from when a vulnerability is reported to a vendor (c) until the time when a patch is issued by that vendor (e). For instance, ZDI and iDefense both buy vulnerabilities from the security research community and then report them to the appropriate vendor. In doing so, they record the time from report to patch release. Essentially, this leaves us with only two stages in the vulnerability life cycle that can be accurately known:

[☆]NOTICE: This is the author's version of a work that was accepted for publication in Information Security Technical Report. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Information Security Technical Report, 17(4), April 2013, pp. 173–184, http://dx.doi.org/10.1016/j.istr.2013.02.002

This manuscript has been authored by Battelle Energy Alliance, LLC under Contract No. DE-AC07-05ID14517 with the U. S. Department of Energy. The U. S. Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for U. S. Government purposes.

*Corresponding author

Email addresses: jlwright@ieee.org (Jason L. Wright), miles.mcqueen@inl.gov (Miles McQueen), lawrence.wellman@inl.gov (Lawrence Wellman)

[1]http://www.inl.gov

- birth: vulnerability reported to the vendor (c), and
- death: patch issued by the vendor (e).

For this article we define vulnerability lifespan to be the time a vulnerability has spent in the vendor's queue. This is the time between the birth and death of the vulnerability. A vulnerability is considered "active" from the time it is reported to or discovered by the vendor until a patch is supplied by the vendor. Metrics based on the number of "active" vulnerabilities in a vendor's queue can be used to aid quantitative estimation of end-user exposure.

Simply examining the raw quantity of vulnerabilities reported for a product in databases like the National Vulnerability Database (NVD) or the Open Source Vulnerability Database (OSVDB) neglects the effect of the vendor response time to addressing vulnerabilities. Likewise, examining the lifespans of vulnerabilities from sources such as the Zero Day Initiative (ZDI) or iDefense neglects the number of vulnerabilities. New metrics which combine both quantity and lifespan of vulnerabilities for individual products would be useful.

In this article, we propose two new metrics that capture the effect of the number and rate of new vulnerabilities being found and their lifespans. The first metric, median active vulnerabilities (MAV), is the median number of software vulnerabilities which are known to the vendor of a particular piece of software but for which no patch has been publicly released by the vendor. The second metric, vulnerability free days (VFD) captures the probability that a given day has exactly zero active vulnerabilities.

1.1. Summary of contributions

We focus on the end-user software vulnerability exposure from individual products by defining two new end-user metrics and we use these two metrics in a case study of four browsers (across vendors) and two other products (within vendor) to discuss and demonstrate that:

- end-user vulnerability exposure should be considered as a combination of lifespans and vulnerability announcement rates (not lifespans alone), the proposed metrics capture both aspects;
- the two metrics may be easily estimated with reasonable accuracy, and thus are usable by end-user security practitioners and decision makers;
- individual products with the same functionality, e.g. browsers, may yield distinctly different end-user vulnerability exposure levels.

1.2. Organization of Article

The rest of this article is organized as follows. In Section 2 we provide an overview of vulnerability stakeholders. In Section 3, we expand on the two new end-user focused metrics and then in Section 4, we collect data on four browsers and examine the results of applying our metrics. Section 5 contains discussion and describes related work. Finally, Section 6 provides conclusions and areas for future work.

2. Overview of Vulnerability Stakeholders

It is important to consider the various stakeholders in software vulnerabilities because each stakeholder observes different effects as vulnerabilities are discovered, reported, and then mitigated. This section discusses each stakeholder in the process.

There are four primary stakeholders in the vulnerability disclosure process:

- vendors who produce software products,
- vulnerability researchers: individuals or firms, who actively search for vulnerabilities or buy them, and then report the vulnerability to the vendor,
- end-users: enterprises or individuals, who are confronted with the potential for loss from vulnerabilities.
- vulnerability repositories: who collect and disseminate the data required for accurate calculation of the metrics.

Each of these stakeholders will be discussed in the following subsections.

2.1. Software vendors

Software products have vulnerabilities. The absolute number of vulnerabilities within any given software product is currently not measurable with any degree of confidence (Ozment and Schechter, 2006; Rescorla, 2005). What can be determined, and what software vendors must confront, is the number of vulnerabilities being reported and how long it takes to produce a patch. The length of time it takes to produce a patch is directly under the control of the vendor and can be directly influenced by the quality and quantity of resources devoted to the task. It is a business decision, and each vendor (perhaps each vendor's product line) has their own unique costs and benefits to consider.

The number of vulnerabilities being reported for the product is, at best only indirectly influenced by the vendor. The vendor can adopt some form of secure software development process such as Microsoft's Secure Development Life Cycle (Microsoft, 2010), which in principle would reduce the number of vulnerabilities which would have otherwise occurred. But the vendor can control neither the level of attention of nor the tools available to vulnerability researchers. As the quantity and quality of researchers looking at the deployed product increases, we would expect the number of vulnerabilities reported to also increase. As the tools available to researchers for aiding the identification of vulnerabilities improve or represent new types of

attack, the number of vulnerabilities being reported would also be likely to increase.

From a business cost and end-user use perspective, vendors would prefer that vulnerabilities never be announced or even found. However, they have little opportunity to control the release of vulnerability information unless they develop contracts with those researchers identifying and demonstrating vulnerabilities. While this has occurred, there are difficulties such as the fact that buying the information does not imply control; for example, other researchers may find the same vulnerability. Consequently, vendors must balance resources expended to develop and deploy patches for vulnerabilities against the potential losses of revenue due to reduced end-user choice of their product.

The metrics defined in this article provide a benchmark to compare one vendor against another. Internally, product groups can compare themselves with other product groups within the same vendor. Just as companies doing hazardous work strive for long stretches with no safety accidents, striving for high vulnerability free days or low median active vulnerabilities could be a development goal itself.

2.2. Vulnerability researchers

Vulnerability research firms actively search for or buy vulnerabilities for some purpose. In this article we are only addressing the researchers who intend to report vulnerabilities to the vendor. The purpose may be to gain notoriety in the hopes of increasing business volume, develop relationships which lead to increased recognition and security related business opportunities, or perhaps, altruistically, to improve the security of software products. In many cases recognition of the security firm, whether organization or individual, seems to be important.

For example, the security firm Secunia earns money by publishing advisories to its customers based on the vulnerabilities discovered by its research. The parent company of ZDI, TippingPoint, produces advisories and intrusion detection system (IDS) signatures for its paying subscribers based on their internal research and the vulnerabilities purchased from other researchers.

In terms of the metrics defined in this article, vulnerability researchers have the ability to keep the software vendors honest. The researchers know when they discovered a vulnerability and more importantly when they reported it to the vendor. They are also best positioned to determine whether a particular patch or solution fixes the problem. Currently, estimating VFD and MAV requires no help from software vendors, but the estimates are not as precise as could be with more comprehensive data.

2.3. End-users

The end-users of software products which have vulnerabilities that have been discovered but remain unpatched expose themselves or their firms to risk. Ideally, end-users could know how many vulnerabilities exist in the software products they are using, determine the probability they will be exploited, and effectively determine the potential losses. But as discussed in Section 1, this information is neither dependably available nor verifiable. So new techniques are needed to help end-users assess their risk.

Vulnerabilities which have been publicly announced help end-users make rational decisions about whether to apply a patch if available, institute a workaround such as disabling the service or reconfiguring the process, or accept the risk. Vulnerabilities which have not been privately reported to the user, publicly announced, or mitigated by a third party such as Tippingpoint supplying IDS signatures for vulnerabilities they have purchased, leave the end-user relatively blind to the particular risk from these vulnerabilities. Vulnerabilities which have been discovered and reported to the vendor but not yet fixed constitute a risk which is mostly undetermined but may present opportunity for improved estimation. The end-user exposure to these software vulnerabilities are discussed in detail in Section 3.

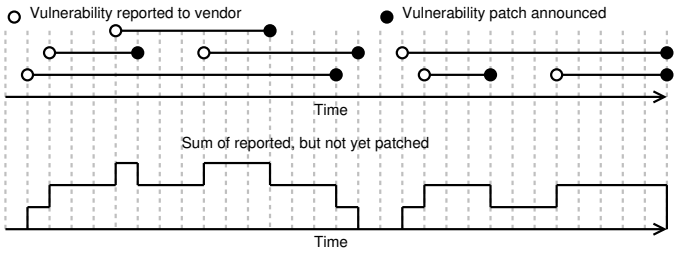
The MAV and VFD measure the relative exposure of end-users to vulnerabilities in products from software vendors. As a result, the metrics defined here, along with required features, could form the basis for choosing to use one software product versus another.

2.4. Vulnerability Repositories

Vulnerability repositories such as the NVD or OSVDB collect vulnerability information and disseminate it to the public. It is necessary that their data be publicly available to enable accurate, verifiable calculation of the metrics proposed in this article. Further, data needs to be available regarding the full life cycle of vulnerabilities; specifically, we require information to be collected on the time of report of a vulnerability until the time a corresponding patch is released. These requirements will become clear in Section 4.

3. Two End-User Exposure Metrics

It is useful to provide all stakeholders (vulnerability researchers, vendors, and end-users) with security metrics which support accountability and decision making. To this end, we define two vulnerability exposure metrics as proxies for a product's contribution to an end-user's level of vulnerability exposure. The first metric, Vulnerability Free Days (VFD), is the percentage of days in which the vendor's queue of reported vulnerabilities for the product is empty; viewed over the long term this is the probability that there are no active vulnerabilities on a given day. The second metric, Median Active Vulnerabilities per day (MAV), is the median number of vulnerabilities per day in a vendor's product queue. The median is used instead of mean because the median is less sensitive to extreme outliers and skewed distributions.



active vulnerabilities	# days	percent
0	3	10%
1	6	20%
2	17	57%
3	4	13%
MAV	2	
VFD	10%	

Figure 2: Example of MAV and VFD calculation

Figure 2 shows a hypothetical example. At the top, vulnerabilities are reported and patched as time moves from left to right. The bottom shows the running sum of active vulnerabilities. If we take each horizontal division as a day, there are 3 days with no vulnerabilities, 6 days with exactly 1 active vulnerability, 17 days with 2, and 4 days with 3. The MAV is the median number of active vulnerabilities: 2 (in other words, on a given day there is a 50% chance the vendor is working on 2 or fewer vulnerabilities). The VFD is $^3/_{30} = 10\%$.

These metrics are primarily intended for consumption by end-users, particularly those in charge of making policy decisions as to which software vendors and products should be used, or which should form part of an "allowed use" policy. Comparative evaluation of software products, or vendors as a whole, can be expressed by calculating and examining their MAV and VFD values. A product with a small median number of active vulnerabilities should have some preference over one with a higher median. The inverse is true with the vulnerability free days metric where a large number is preferred to a small number.

The information needed to calculate these two metrics for a product are the lifespan of each reported vulnerability, and the number and rate of vulnerability disclosures. While not currently easy to obtain, in principle this information would be easy to produce and verify by the vendors. The data could also be verified by the independent researchers who reported vulnerabilities, and the vendors could be induced to make the information free and easily accessible if end-user pressure is brought to bear.

4. Metrics Case Study

The proposed VFD and MAV vulnerability exposure metrics were estimated for several different software products: web browsers (Apple Safari, Google Chrome, Mozilla Firefox and Microsoft Internet Explorer) as well as Microsoft Office and Apple QuickTime. To develop the metrics for each product, data was collected in order to characterize their respective vulnerability lifespans, and number and rate of vulnerability disclosures. After some success in characterizing this information for each product, a simulation was written and used to estimate the metrics. The possibility for quick and easy short cuts for approximating the metrics are discussed at the end of the case study (Section 4.5).

In particular, we used several data sources to estimate the:

- arrival rate of vulnerability announcements,
- number of vulnerabilities announced, and
- lifespan of vulnerabilities.

The arrival rate of vulnerability announcements is the time between two different announcements of vulnerabilities for a given product. The number of vulnerabilities announced represents the integral number of vulnerabilities disclosed as part of a specific announcement. It is common that more than one vulnerability for a given product is announced on a given announcement day (e.g. Microsoft "patch Tuesday"). The lifespan of a vulnerability is the same as defined previously. It begins when the vulnerability is reported or discovered by the vendor and ends when the vendor supplies a patch.

4.1. Data Sources

Data was gathered from the National Vulnerability Database (NVD) (NIST, 2011), iDefense Vulnerability Contributor Program (VCP) (Verisign, 2011), and the Zero Day Initiative (ZDI) (TippingPoint, 2011). The NVD data was used to characterize the arrival rate of vulnerability announcements and the number of vulnerabilities announced per instance. The ZDI and iDefense data were used to characterize vulnerability lifespans. In all cases, descriptive statistics are provided to give an idea of the behavior of the data harvested from each source.

The NVD consists of approximately 46,000 unique vulnerabilities enumerated by an identifier called a Common Vulnerability Enumeration Identifier (CVE). The database is freely available and further breaks down vulnerabilities by vendor, product, version, etc. (Common Platform Enumeration, CPE). For our research, the XML data feed provided by NVD was downloaded and imported into an SQL database so that our desired queries could be executed. The data was used for computing the arrival rates of vulnerabilities, and determining the number of vulnerabilities disclosed at each announcement.

The National Vulnerability Database has been widely criticized for the inaccuracies it contains. For example, Frei et al. (2006), Clark et al. (2010), and Zhang et al. (2011) all describe various inconsistencies in the NVD and other vulnerability databases. In this article, we are primarily describing the concept and potential usage of our metrics, so we are less concerned with the absolute consistency of the existing sources.

Product	N	Time Start	Time End
MS Internet Explorer	209	Jan 1, 2001	Jun 6, 2010
Mozilla Firefox	113	Jan 1, 2004	Jun 6, 2010
Google Chrome	30	Dec 12, 2008	Jun 6, 2010
Apple Safari	92	Jun 22, 2003	Jun 6, 2010
MS Internet Explorer 6	180	Jan 1, 2001	Jun 6, 2010
MS Internet Explorer 7	85	Jan 1, 2004	Jun 6, 2010
MS Internet Explorer 8	20	Jan 1, 2009	Jun 6, 2010
MS Office	246	Jan 1, 2005	Jun 6, 2012
Apple QuickTime	138	Jan 1, 2005	Jun 6, 2012

Table 1: Number of points and time span for each product in NVD.

To minimize the effects of the erroneous data in the NVD, the time span of analysis is limited for each product and only two fields were used: the Common Platform Enumeration (CPE) and the "first published" date. The vendor and product fields of the CPE were used to discriminate between products. Other parts of the CPE were ignored, except when making the distinction between Internet Explorer versions. The "first published" field of the NVD is used to examine the arrival rate of announcements and the number of vulnerabilities announced per day.

Limiting the dates for which we collected vulnerability data used for each product allows us to ignore the start-up effects of the NVD. As pointed out in Zhang et al. (2011), the early years of the NVD were unstable. Table 1 shows the time span considered for each product and the number of data points available in the time span. Before 2004, the Firefox browser was a product of Netscape Communications called Netscape Communicator. It is difficult to determine whether vulnerabilities were inherited from the Communicator product or introduced during the transition to the Mozilla Foundation Firefox product, so we limited our study of this product to vulnerabilities discovered after the transition. Google Chrome was introduced in December 2008, and Apple Safari was released in January of 2003. In addition to Microsoft Internet Explorer as a whole, individual versions are broken out separately.

The data from iDefense and ZDI is considered to be more reliable since they can directly observe the time between when they notify a vendor and when a corresponding patch is produced. The former time being directly controlled by iDefense/ZDI and the latter time being publicly observable. For vendors like Microsoft which produce their own security advisories, it might be possible to gather the required data from the issued advisories. However, using the simulation method described in Section 4.2 is a more general solution and integrates more reliable observations of vulnerability lifespans than those provided using the NVD alone.

4.1.1. Vulnerability Announcements

Figure 3 shows the histogram of vulnerability announcements for all of the studied products; while the mean and median values differ substantially, the histograms have roughly the same shape. Table 2 summarizes the statistical properties of the announcement rate. If one were to choose a web browser simply by the arrival rate of new vulnerability announcements, one would choose Apple Safari because the expected time between new vulnerability announcements is slightly over 25 days (more than 3 weeks), and the other browsers are less than 3 weeks. Firefox does not fare well at all with new vulnerabilities announced about 12 days apart.

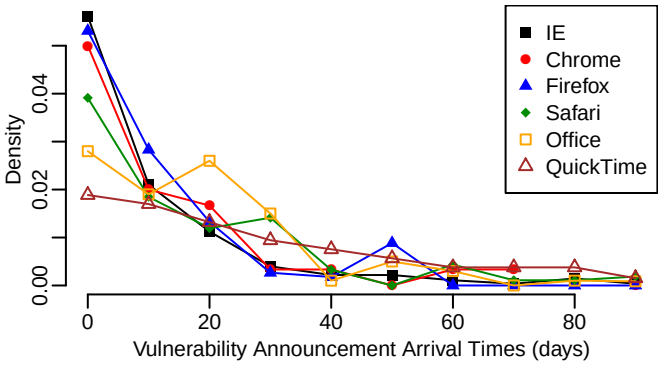


Figure 3: Histogram of vulnerability announcement rates.

Product	mean	median	σ	min / max
MS Internet Explorer	14.95	9.0	16.3	1 / 98
Mozilla Firefox	12.09	10.0	10.5	1 / 51
Google Chrome	17.17	10.5	18.4	1 / 80
Apple Safari	25.47	15.5	28.5	1 / 125
MS Internet Explorer 6	17.36	10.0	18.6	1 / 97
MS Internet Explorer 7	23.14	13.0	41.7	1 / 365
MS Internet Explorer 8	21.15	14.0	16.5	1 / 54
MS Office	24.11	22.0	20.0	1 / 113
Apple QuickTime	46.83	33.0	45.4	1 / 202

Table 2: Properties of vulnerability announcement rates (days).

4.1.2. Number of Announcements per day

However, because arrival rate is actually an announcement of at least one vulnerability and possibly more, we examine the distribution of the number of vulnerabilities on an announcement day. The distributions for each studied product are shown in Figure 4. Table 3 summarizes the number of vulnerabilities per announcement. Internet Explorer and Safari are close to 2 vulnerabilities per announcement on average where as Firefox averages more than 3 vulnerabilities per announcement. However, the median number of vulnerabilities announced on an announcement day for all products is 1, meaning that at least half of the announcements are for a single vulnerability.

4.1.3. Vulnerability Lifespans

The ZDI and iDefense databases consist of vulnerabilities for which the corresponding firm has paid a security researcher for a vulnerability. ZDI or iDefense then works with the affected vendor to responsibly disclose the vulnerability. Both companies provide free and online access to the data including the date the company reported the vulnerability to the vendor and the date at which the vulner-

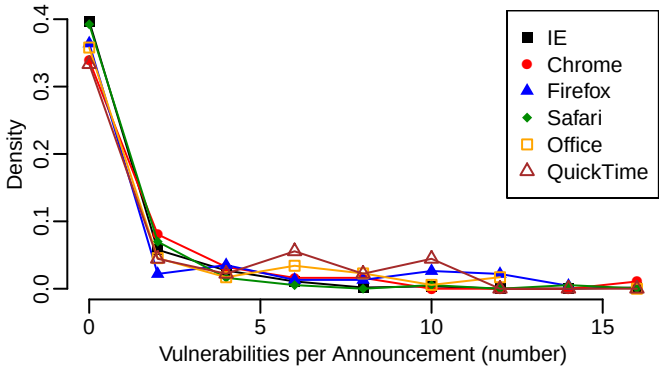


Figure 4: Histogram of vulnerabilities announced on announcement day.

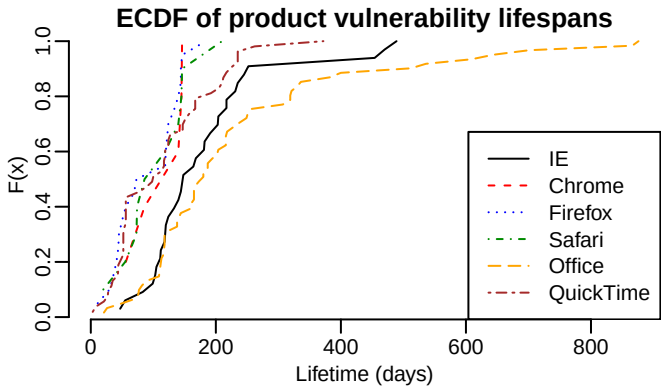


Figure 5: Empirical cumulative distribution functions of product vulnerability lifespans.

Product	mean	median	σ	min / max
MS Internet Explorer	2.105	1.0	2.075	1 / 17
Mozilla Firefox	3.158	1.0	3.811	1 / 16
Google Chrome	2.871	1.0	3.667	1 / 19
Apple Safari	2.279	1.0	4.108	1 / 36
MS Internet Explorer 6	2.188	1.0	2.121	1 / 15
MS Internet Explorer 7	1.733	1.0	1.332	1 / 6
MS Internet Explorer 8	1.221	1.0	1.221	1 / 5
MS Office	2.795	1.0	3.231	1 / 14
Apple QuickTime	3.067	1.0	3.055	1 / 11

Table 3: Properties of vulnerability announcement rates (number of announcements).

ability was publicly disclosed. We use the collected data for computing the distribution of vulnerability lifespans.

Table 4 shows the descriptive statistics for the distribution of the ZDI and iDefense lifespan data. Firefox has the clear lead at 91.6 days to address vulnerabilities and Internet Explorer lags far behind with a mean of 182 days to address vulnerabilities. Figure 5 shows a diagram of the empirical cumulative distribution functions of the lifespans for each browser. For each observed sample lifespan, the graph rises $1/N$ at that point along the horizontal axis. A rapid vertical rise shows a clustering of observed lifespans and small slope shows few observed lifespans of that value. Figure 5 is a more detailed examination of the distribution information in Table 4. For instance, MS Internet Explorer is shown to have an overall slower distribution of lifespans; part of this is caused by a small number of high value lifespans ($>$ 450 days). The other three browsers have similarly positioned and shaped lifespan distributions.

Product	N	mean (days)	σ (days)	min / max (days)
MS Internet Explorer	33	182.1	106.9	47 / 489
Mozilla Firefox	20	91.6	50.7	11 / 184
Google Chrome	5	114.6	41.3	56 / 146
Apple Safari	10	106.8	55.5	20 / 210
MS Office	61	235.3	186.5	21 / 876
Apple QuickTime	53	113.4	79.4	3 / 372

Table 4: Distribution of ZDI/iDefense lifespans for each product.

4.2. Model for Simulation

To facilitate estimation of the MAV and VFD metrics, a model and corresponding simulator were constructed. We employ a simulation because the exact data are not known and a closed form solution based on the empirical distributions is not yet available (though an approximation is found and discussed in Section 4.5). To generate a single simulation run, time is set to t_0 and a sample is taken from the announcement arrival rate distribution for the browser under study, Δt. Then, at time $t = t_0 + \Delta t$, a sample is taken from the distribution of the number of vulnerabilities announced on an announcement day. This determines how many vulnerabilities are terminated with the announcement, n. For each $i \in 1, \ldots, n$, a sample is taken from the lifetime distribution, l_i.

For the discrete event simulation, two events are generated:

- a vulnerability birth at time $t - l_i$ and
- a vulnerability death at time t.

Finally, t_0 is set to t and event generation continues until $t_0 > t_{\text{end}}$ where t_{end} is the simulated time.

To compute the MAV metric, the discrete number of vulnerabilities estimated to be in the vendors queue each day was put in rank order and the probability of each was computed. Finding the median is then a matter of finding number of vulnerabilities corresponding to the 50th percentile. The VFD metric is calculated by counting the number of days in the simulation with exactly zero vulnerabilities, then dividing by the simulation days to obtain the probability of no vulnerabilities. To minimize simulation warm-up and wind-down, the simulation was run for 100 different random seeds and over a simulated time of 100 years.

This simulation model is a $G/G/\infty$ queuing model: generalized arrival process, generalized service time, and an infinite number of servers. The arrival process is complicated by the fact that multiple vulnerabilities can be announced at a single point in time. Even if the underlying data could be mathematically modeled, the authors

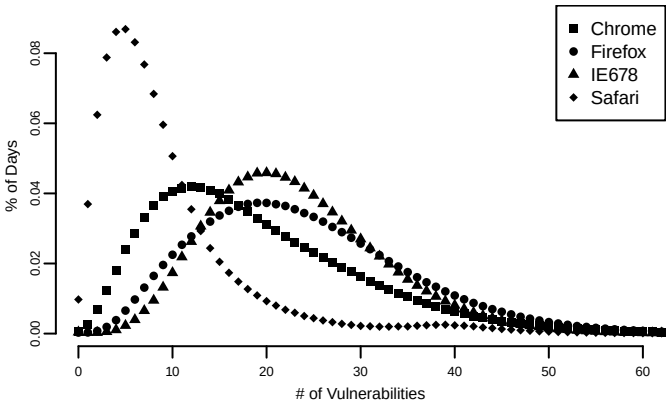


Figure 6: Percentage of days with the given number of vulnerabilities in a vendors queue

believe that there is no closed form solution for the MAV or VFD metrics.

Various statistical models were tried for each of the different probability distribution functions required by the simulation. Since the model parameters were not equally well characterized by the statistical models, the simulations were run using the raw data collected for each parameter as a discrete distribution function. The results of the simulations were used to calculate the VFD and MAV for each browser.

4.3. Estimation of MAV and VFD Across Vendors

For estimating the MAV metric, the arrival rate of announcements, number of vulnerabilities disclosed per announcement, and the vulnerability lifespans are random variables distributed as described in Section 4.1.3. The distributions were derived from the collected data. The simulation provided the results shown in Figure 6. The horizontal axis is the number of vulnerabilities in a vendor's queue and the vertical axis is the percentage of days which had that number of vulnerabilities. The MAV metric was then calculated as the median number of active vulnerabilities.

The MAV estimate for each of the four browsers was 9.55 for Safari, 19.1 for Chrome, 23.9 for Firefox, and 23.2 for Internet Explorer (this data is summarized in Table 5). So the estimated vulnerability exposure, MAV, due to deployment of a web browser is distinctly different depending on which web browser is in use. Safari is clearly superior to the other three browsers.

However, there is a question of whether it is reasonable to group the data from Internet Explorer versions 6, 7, and 8 together since each version might have distinctly different values for the model parameters and thus different MAV metric values. So we further decomposed Internet Explorer, and recalculated the MAV for each version. Grouping the three versions together results in a higher overall MAV because the sets of vulnerabilities are not independent; a vulnerability may affect one or more major

Browser	MAV	σ
Apple Safari	9.55	8.58
Google Chrome	19.1	11.3
Mozilla Firefox	23.9	11.1
Internet Explorer (all)	23.2	8.94
Separate treatment of IE versions		
Internet Explorer 6	20.7	8.70
Internet Explorer 7	12.2	6.90
Internet Explorer 8	13.5	4.76

Table 5: Browser Median Active Vulnerabilities

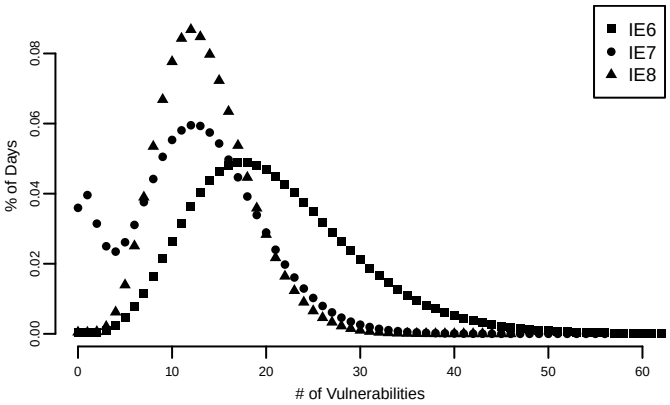


Figure 7: Days with given number of vulnerabilities (MS IE)

versions of the browser. This in turn affects the sampling of report rate, announcement rate, and lifespan.

Similar to Figure 6 the simulation results for Internet Explorer versions 6, 7, and 8 are shown in Figure 7. The MAV estimate was 20.9 for Internet Explorer version 6, 12.2 for Internet Explorer version 7, and 13.5 for Internet Explorer version 8. Internet Explorer 6 is clearly the poorest performer according to the MAV estimates. This is in line with the general security community expectations. The cause for Internet Explorer 6 showing so poorly while versions 7 and 8 are have quite similar MAV values is unknown. We speculate that the difference lies in the fact that Internet Explorer 7 and 8 have more common code than either have with version 6. Also, the Microsoft Security Development Life Cycle became a mandatory policy at Microsoft in 2004 (three years after the release of IE6, 2001, and two years before IE7, 2006) (Microsoft, 2010).

For estimating the VFD metric, the arrival rate of announcements and number of vulnerabilities announced per announcement are random variables distributed as described in Section 4.1.3. In Figure 8 the lifespan of vulnerabilities was varied from 1 day (vulnerabilities are addressed practically as soon as they are reported) to 182 days. The lifespan is varied along the horizontal axis, and the percentage of vulnerability free days is shown on the vertical axis. Our goal was to examine the behavior of the VFD metric as the result of different vulnerability lifespans for products.

The results are provided for Safari, Chrome, Firefox, and Internet Explorer (versions 6, 7, and 8 are treated

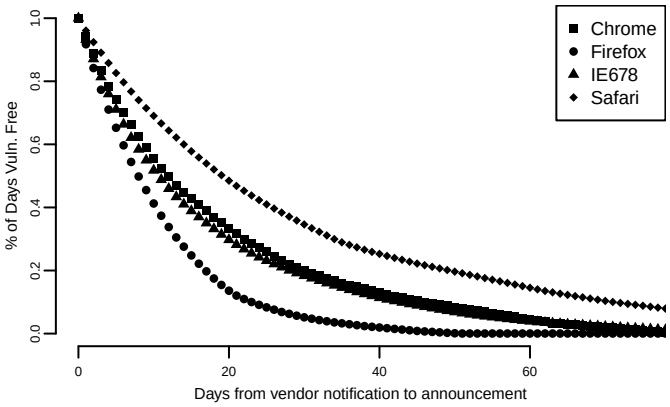


Figure 8: Vulnerability Free Days (VFD), as a function of lifespans

as an aggregate since we are examining vendor behavior). The most interesting result is that even for a vulnerability lifespan of 45 days, the percentage of days which are vulnerability free are less than 20% for Safari and less than 6% for the three other browsers. Even Safari, the best performing browser as judged by this metric, does not do well. When the lifespan is the length of those we actually measured, approximately 75 days for Safari and 146 days for Internet Explorer, the VFD for all browsers is less than 10%. A poor performance by all browsers.

4.4. Comparisons Within a Vendor

As mentioned earlier, the metrics can be used to compare different products from within the same vendor. The allows for a measurement of the relative effectiveness of corporate coding standards and policies.

Table 6 shows the MAV metric simulated for both Apple Safari and QuickTime. The difference in MAV shows that, on average, the QuickTime developers are working on 2 fewer vulnerabilities than the Safari team. There are several possible explanations for this. It might be that the Safari browser, being the default browser for MacOS systems, is under more scrutiny by security researchers (with more time dedicated to examining it perhaps more vulnerabilities are found). From Table 4, the lifespan of vulnerabilities is not largely different (106.8 and 113.4 days for Safari and QuickTime, respectively), and the number of vulnerabilities announced per announcement day (Table 3) is 2.279 and 3.067. The biggest difference is in the mean days between announcements: 25.47 and 46.83 days for Safari and QuickTime, respectively (Table 2). Therefore the time between announcements of vulnerabilities is the single largest factor determining the MAV difference for these products. Figure 9 shows the distribution of MAV from the simulation; the curves almost overlap except for a small shift in peak.

We also compared Microsoft Internet Explorer and the Microsoft Office suite (also Table 6). In this case, the browser (IE) has a considerably smaller MAV than the

Vendor	Product	MAV	σ
Apple	Safari	9.55	8.58
	QuickTime	7.41	5.51
Microsoft	IE (all)	23.2	8.94
	IE 6	20.7	8.70
	IE 7	12.2	6.90
	IE 8	13.5	4.76
	Office	27.1	10.3

Table 6: Comparison of products within a vendor

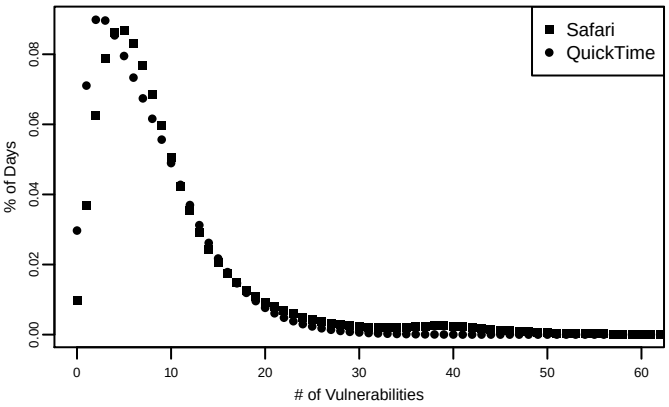


Figure 9: Comparison of Microsoft Products

application suite (27.1 and 23.2 for Office and IE, respectively). If compared to individual releases of IE, the difference is even more pronounced. We will restrict ourselves to comparisons against all IE data as this allows comparison over the same time period and presumably the same changes in corporate culture within Microsoft. Comparing the data for the two products from Table 2, 3, and 4, there are two primary factors in the higher MAV: vulnerabilities per announcement and vulnerability lifespan. In general there is a longer gap between announcements of vulnerabilities in Office (14.95 and 24.11 days for IE and Office, respectively), but the number announced on an announcement day and the mean lifespan is much higher. On average, Microsoft takes 53.2 days longer to fix a vulnerability in the Office suite than for IE. Figure 10 shows the distribution of MAV from the simulation for the two products.

4.5. Simplification of Metrics Calculations

Both the MAV and VFD metrics could be used by end-users when making software product purchasing or allowed use decisions. However, to gain use, they need to be able to be quickly calculated when the proper information is available. For end-users who are unable to deploy a simulation to calculate MAV and VFD it would be useful if there were short cut calculations to make first order estimates of the metrics. We formulated two short cuts and compared the results to those from the simulation. The formulas may be found in (1) and (2).

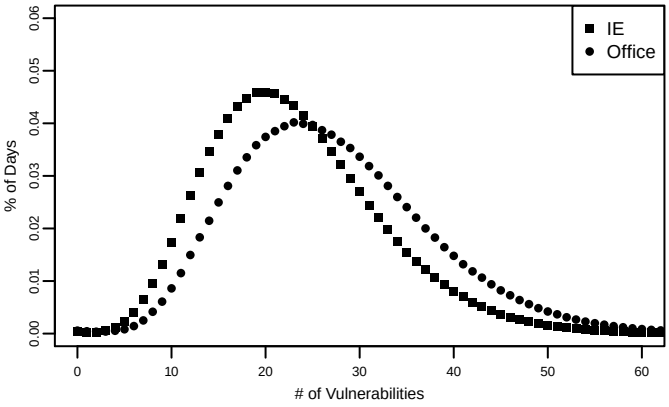


Figure 10: Comparison of Microsoft Products

Product	Simulated MAV	Short Cut MAV	% Error
MS Internet Explorer	23.2	24.6	6.09%
Mozilla Firefox	23.9	23.9	0.01%
Google Chrome	19.1	19.2	0.34%
Apple Safari	9.55	9.56	0.10%
MS Office	27.1	27.3	0.66%
Apple Quicktime	7.41	7.43	0.23%

Table 7: Comparison of simplified calculation of MAV to simulated.

$$MAV = \frac{\text{(Average Lifespan)(Average Reported)}}{\text{Average report rate}} \quad (1)$$

$$VFD = (1 - e^{-1})^{\text{MAV}} \approx 0.632^{\text{MAV}} \quad (2)$$

Equation (1) comes from treating the MAV as an average; the average number of vulnerabilities announced per day is the product of the number of announcements on an announcement day (Table 2) and the inverse of the days between announcements (Table 3). To get active vulnerabilities, we scale that quantity by the average lifespan of a vulnerability (Table 4).

Equation (2) was derived using a least squares fit of an exponential model to the data and observing the resulting base was close to $1 - e$. Additional terms could be added to the model, but the goal here was to provide a simple calculation for use in estimation.

Table 7 shows the result of using the simulation data versus the simplified calculation using (1). The simplified version does a reasonable job of estimating the results of the simulation and is easy calculated directly from available data. The worst estimation performance from Table 7 is Internet Explorer (6% error), yet even this calculation is less than 1.5 vulnerabilities in magnitude.

The idea behind using MAV to compute VFD is expressed from a software vendor point of view; namely, a vendor has some control over the number of developers assigned to addressing vulnerability reports. By adjusting the speed with which vulnerabilities are patched, a vendor can pick a target VFD probability and find the average lifespan needed to achieve it.

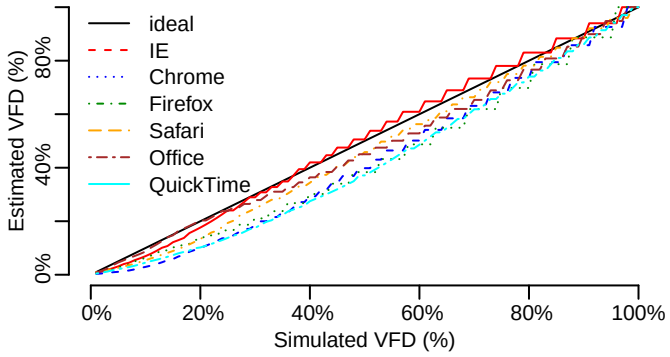


Figure 11: Estimation of VFD metric using shortcut formula

Figure 11 displays the simulated versus estimated VFD values. Ideally, the lines for each product would follow the line $y = x$; the departure from this is the estimation error. Generally, the curves follow a linear shape meaning that the first order effects of the simulation are captured by the estimation. The model fits well the behavior of VFD for Internet Explorer and Safari and somewhat less for Chrome and Firefox.

5. Discussion

This section discusses several observations made from examination of the ZDI/iDefense data and the NVD. In particular, we examine the effect of severity of a vulnerability versus lifespan (Section 5.1) and possible difference of severity scores based on dataset (Section 5.2). Finally, we examine related work and contrast our approach (Section 5.3).

5.1. Severity Versus Lifespan

One of the widely used metrics for vulnerability severity is the Common Vulnerability Scoring System (CVSS) maintained by NIST[2]. For all vulnerabilities collected from iDefense and ZDI, we correlated the CVSS score as recorded in NVD. CVSS scores vulnerabilities along several dimensions: impact to confidentiality, integrity, and availability; access vector; access complexity; etc. The resulting score is meant to be an ordinal score of the severity of a vulnerability. The question was then, do vulnerabilities with higher impact (higher CVSS score) take longer to fix (have a longer lifespan from report to patch)?

Figure 12a shows the distribution of lifespans for each ordinal value of CVSS score for all products in the ZDI and iDefense data set. From this graph, it is not at all clear that CVSS score predicts lifespan; in other words, the severity of a vulnerability does not clearly predict how long it will take the vendor to fix it. It is true that vulnerabilities with the highest CVSS scores (9.3 and 10.0) also

[2] `http://www.first.org/cvss`

account for the most extreme lifespans, the distribution is highly skewed towards lower lifespans.

Figures 12b through 12g are subsets of the data set separated by product. There is no clear indication in any of the graphs that CVSS score and lifespan are related. In the case of Google Chrome, all 5 vulnerabilities have a CVSS score of 9.3, and Apple Safari has one vulnerability scored at 6.8 and the remaining 9 are scored at 9.3.

We conclude then that CVSS score is not a good indicator of the lifespan of a vulnerability. Practically this means that the time vendors take to fix vulnerabilities has no relationship to the impact a vulnerability may have on the vendor's end-users.

5.2. Comparison of CVSS in Data Sets

It is possible that vulnerabilities collected by ZDI and iDefense differ in some way from those reported in the National Vulnerability Database. One of the few ways to compare the two data sets is by CVSS score. To test the hypothesis of a difference, we performed Kruskal–Wallis tests on CVSS scores separated by the factor of source (either directly from NVD or from the ZDI/iDefense data we collected), and Table 8 shows the results. This particular test was chosen because of its non-parametric nature (less sensitivity to skewed distributions).

Interestingly, for non-browser products (Apple QuickTime and Microsoft Office), we fail to reject the hypothesis of a difference in location between CVSS scores in the NVD and those vulnerabilities bought by ZDI/iDefense. However, in all cases, for the browsers we must reject the hypothesis that the CVSS scores are not shifted in location. The distribution of CVSS scores for the web browsers is significantly shifted upwards in the data from ZDI/iDefense; this means that the severity of those vulnerabilities is considerably higher.

Product	χ^2	p-value
Apple Safari	13.63	2.2×10^{-4}
Google Chrome	5.63	0.017
Microsoft Internet Explorer	27.54	1.5×10^{-7}
Mozilla Firefox	6.43	0.011
Apple QuickTime	0.67	0.411
Microsoft Office	0.42	0.517
All Products	657.1	$< 2.2 \times 10^{-16}$

Table 8: Tests for difference in CVSS scores based on data source ($df = 1$).

Also in Table 8, we compared all CVSS scores from the NVD versus all scores of vulnerabilities bought by ZDI/iDefense. As with browsers, there is a significant shift upwards in severity for vulnerabilities in the ZDI/iDefense data set. This can be clearly seen in Figure 13. This figure also demonstrates the skewed nature of CVSS scores in both NVD and ZDI/iDefense. One possible explanation for the difference between NVD and ZDI/iDefense CVSS scores is that ZDI/iDefense may not be willing to purchase low impact vulnerabilities.

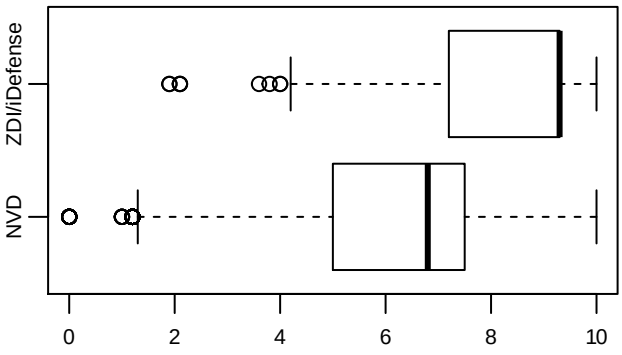


Figure 13: Distribution of CVSS scores: NVD versus ZDI/iDefense.

5.3. Related Work

Software life cycle metrics are a well studied aspect of development. These metrics concentrate on the rate at which defects are detected in the various stages of the life cycle of software. Less well understood are metrics for the security vulnerability life cycle.

Several approaches to understanding the life cycle of vulnerabilities have been undertaken over the past few years. The approaches fall mostly into two methods: examining one or a few software packages in detail or looking for large scale trends.

Ozment and Schechter (2006), for example, falls into the former category. They examined the discovery of vulnerabilities in the OpenBSD operating system across several years and versions to determine whether it is getting fundamentally more secure over time. Their conclusion was that the rate of newly discovered vulnerabilities appeared to be slowing for "foundational" code.

Also in this category is Schryen (2011), who examined 17 different products (open source and closed source). This work concentrated on the question of whether open source products are more secure than closed source products. Schryen concludes that there is no empirical evidence that open source products and closed source products differ significantly. Comparing Mozilla Firefox (open source) against Internet Explorer (closed source) based on the MAV and VFD, the same conclusion might be drawn.

Frei et al. (2006) is an example of the latter category where all vulnerabilities in the NVD and other sources are examined to find global trends. This work does not help, though, when considering individual products or vendors and comparing them.

Arnold et al. (2009) examined a single product: the Linux kernel. They found a significant number of software bugs that were later discovered to be vulnerabilities. These delayed impact vulnerabilities highlight the difficulties in obtaining accurate and verifiable dates for discovery of vulnerabilities. In the case of delayed impact vulnerabilities, the discoverer either did not check whether a bug

was also a vulnerability or its impact was not realized until well after the bug was reported.

More recently, Clark et al. (2010) took a new approach where the first four vulnerabilities for a particular release of a particular piece of software were examined. Using this approach, they claim that extrinsic properties to software development are more indicative of vulnerability discovery than are intrinsic properties like software quality. Their approach is applied across vendors, open source versus closed source, etc.

Arora et al. (2008) examined the vulnerability life cycle by concentrating on an optimal policy for disclosure. Their work provides the model used for discussion of the life cycle in Section 1. However, the approach of optimizing the disclosure policy based on economic factors relies on many variables which are simply not credibly known.

Our approach differs in that we are not wholly interested in the life cycle. Instead, we examined a method for ranking products across vendors or products within a single vendor on the basis of their raw number of vulnerabilities and the speed with which they address them. Our result thus far has been to demonstrate the applicability of the metrics against a small set of products.

As far as vulnerability metrics are concerned, several reports concentrate on the total number of vulnerabilities announced over a given time (per year or per half year) and the number of fixed vulnerabilities over the same time for example: Cenzic, Inc. (2010); IBM X-Force (2010). At a gross level, this information is similar to our MAV metric, but it is not as granular. A vulnerability can last for a year or a day between report and patch and the total announced minus the number fixed will stay the same using this type of counting. The MAV metric takes both the total number of announced vulnerabilities and their lifespan into account in per day units.

Finally, an interesting metric was proposed in Acer and Jackson (2010), which attempts to combine: patch deployment, vulnerability severity, and user-installed browser plug-ins. The authors gather "user-agent" strings reported by browsers visiting a site created by the authors. From this, the number of users who are not completely up to date with patches are counted, and the "best" browser is the one with the fewest number of users who are not fully patched. However, this method depends on random sampling (possibly achievable with strategically placed collectors) and only addresses software which report complete version information. For non-browser products, it is not clear how measurements could be conducted, and even for browsers, the authors found that Internet Explorer does not report all of the necessary information.

6. Conclusions and Future Work

Two new software vulnerability exposure metrics were proposed with the end-user in mind. Both Vulnerability Free Days and Median Active Vulnerabilities were demonstrated in a case study of the four browsers Safari, Chrome, Firefox, and Internet Explorer. Estimation values for the metrics were generated through simulation. Short cut estimations were shown to be practical. Based on the derived exposure metrics for each browser, there are large differences in vulnerability exposure, with Safari having the lowest exposure.

The exposure metrics are sensitive to both lifespans and the number of vulnerabilities being discovered and reported. Firefox, which produces patches quickest, still has one of the worst vulnerability exposures because so many vulnerabilities are discovered and reported. It was also noted that it may not be realistic for any of the browsers to get to even 50% Vulnerability Free Days.

Characterization of the lifespans, vulnerability announcement rates, and the number of vulnerabilities per announcement is continuing as more data is collected and more sophisticated statistical methods are used. The two vulnerability exposure metrics, Vulnerability Free Days and Median Active Vulnerabilities, might be more practical for end-users in a slightly modified disclosure process.

We have begun investigation into a disclosure process which emphasizes the needs of the end-users, the diversity of end-users and software product vendors, and the value of transparency. We also intend to explore and develop a recommended disclosure process for critical infrastructure control systems based on these metrics and the unique aspects of control systems. In all of the new disclosure processes transparency is a critical element. Each process demands that the requisite data be free and easily accessible to end users. The metrics described in this article will form an integral part of these processes.

Acknowledgment

The authors would like to thank Debbie McQueen for her assistance in gathering the ZDI and iDefense data.

References

Acer, M., Jackson, C., May 2010. Critical vulnerability in browser security metrics. In: Web 2.0 Security and Privacy. IEEE Symposium on Security and Privacy. Oakland, CA, USA.

Arnold, J., Abbott, T., Elhage, N., Thomas, G., Kaseorg, A., May 2009. Security impact ratings considered harmful. In: 12th workshop on Hot Topics in Operating Systems. USENIX.

Arora, A., Telang, R., Xu, H., April 2008. Optimal policy for software vulnerability disclosure. Management Science 54 (4), 642–656.

Cenzic, Inc., 2010. Cenzic web application security trends report q3/q4 2010.
URL `http://www.cenzic.com/resources/reg-not-required/trends/`

Clark, S., Frei, S., Blaze, M., Smith, J. M., December 2010. Familiarity breeds contempt: the honeymoon effect and the role of legacy code in zero-day vulnerabilities. In: Annual Computer Security Applications Conference ACSAC. pp. 251–260.

Frei, S., May, M., Fiedler, U., Plattner, B., September 2006. Large-scale vulnerability analysis. In: SIGCOMM Workshop on Large-Scale Attack Defense (LSAD). ACM, pp. 131–138.

IBM X-Force, February 2010. 2010 trend and risk report.
URL `http://www-935.ibm.com/services/us/iss/xforce/trendreports/`

Kandek, W., July 2009. The laws of vulnerabilities 2.0. In: BlackHat. Las Vegas, NV, USA.
URL http://laws.qualys.com/

Microsoft, November 2010. Simplified Implementation of the Microsoft SDL.
URL http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=12379

NIST, 2011. National vulnerability database.
URL http://nvd.nist.gov/

Ozment, A., Schechter, S. E., July 2006. Milk or wine: Does software security improve with age? In: 15th USENIX Security Symposium. USENIX, pp. 93–104.

Rescorla, E., January 2005. Is finding security holes a good idea? IEEE Security and Privacy 3 (1), 14–19.

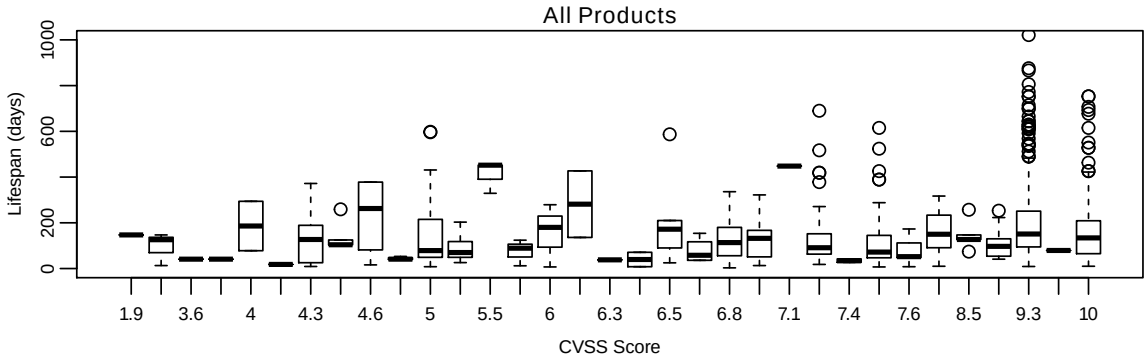
Schryen, G., May 2011. Is open source security a myth? Communications of the ACM 54 (5), 130–140.

TippingPoint, 2011. Zero Day Initiative.
URL http://www.zerodayinitiative.com/about

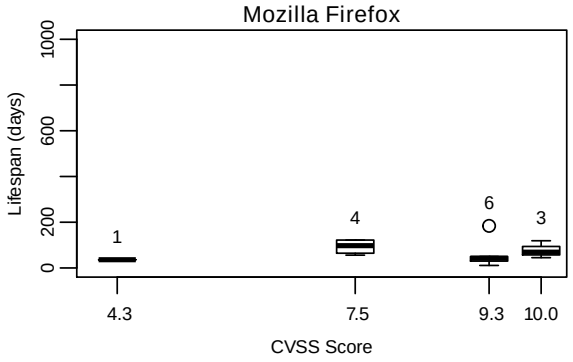
Verisign, 2011. iDefense vulnerability contributor program.
URL http://www.verisigninc.com/en_US/products-and-services/network-intelligence-availability/idefense/public-vulnerability-reports/index.xhtml

Wright, J. L., McQueen, M., Wellman, L., August 2012. Analyses of two end-user software vulnerability exposure metrics. In: 7th International Conference on Availability, Reliability, and Security (ARES). IEEE, Prague, Czech Republic.

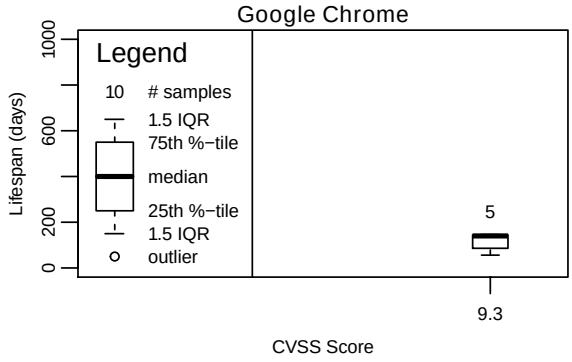
Zhang, S., Caragea, D., Ou, X., August 2011. An empirical study on using the National Vulnerability Database to predict software vulnerabilities. In: International Conference on Database and Expert Systems. Vol. 22.



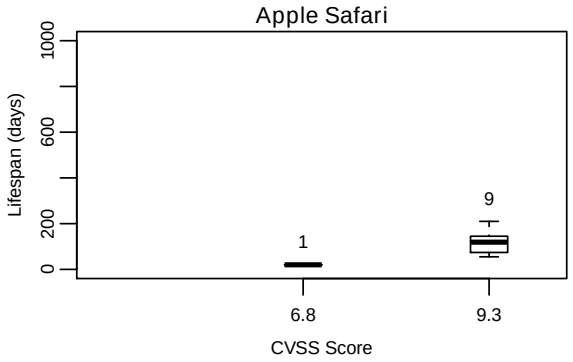
(a)



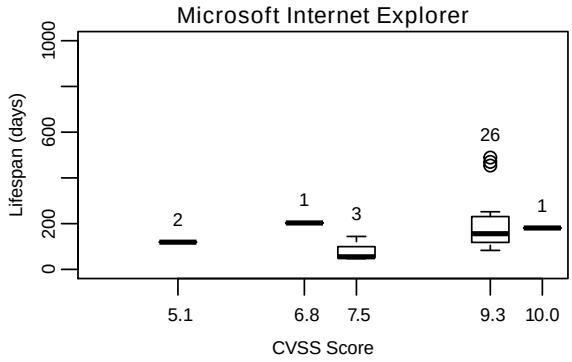
(b)



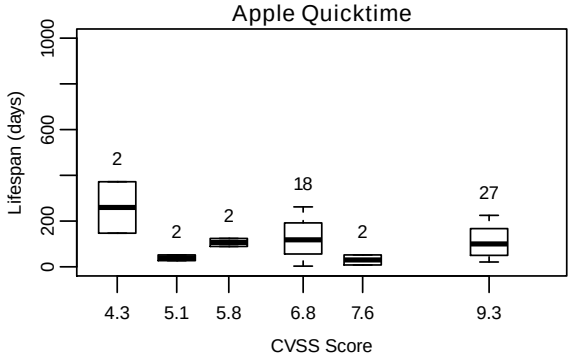
(c)



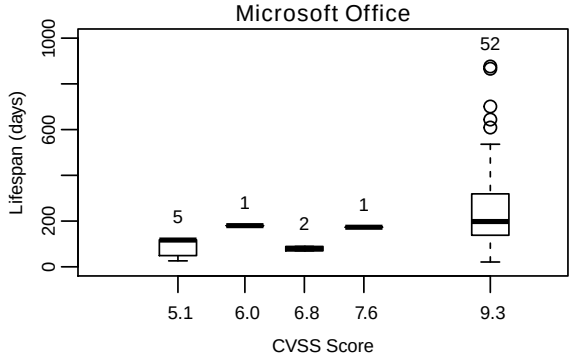
(d)



(e)



(f)



(g)

Figure 12: Comparing CVSS score to vulnerability lifespan