available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Modeling vulnerability discovery process in Apache and IIS HTTP servers

Sung-Whan Woo, HyunChul Joh, Omar H. Alhazmi, Yashwant K. Malaiya*

Computer Science Department, Colorado State University, 1873 Campus Delivery, Fort Collins, CO 80523-1873, USA

ARTICLE INFO

Article history:
Received 29 April 2010
Received in revised form
18 August 2010
Accepted 18 October 2010

Keywords:
Vulnerability discovery model (VDM)
Risk evaluation
Web server
Quantitative modeling
Security

ABSTRACT

Vulnerability discovery models allow prediction of the number of vulnerabilities that are likely to be discovered in the future. Hence, they allow the vendors and the end users to manage risk by optimizing resource allocation. Most vulnerability discovery models proposed use the time as an independent variable. Effort-based modeling has also been proposed, which requires the use of market share data. Here, the feasibility of characterizing the vulnerability discovery process in the two major HTTP servers, Apache and IIS, is quantitatively examined using both time and effort-based vulnerability discovery models, using data spanning more than a decade. The data used incorporates the effect of software evolution for both servers. In addition to aggregate vulnerabilities, different groups of vulnerabilities classified using both the error types and severity levels are also examined. Results show that the selected vulnerability discovery models of both types can fit the data of the two HTTP servers very well. Results also suggest that separate modeling for an individual class of vulnerabilities can be done. In addition to the model fitting, predictive capabilities of the two models are also examined. The results demonstrate the applicability of quantitative methods to widely-used products, which have undergone evolution.

1. Introduction

There has been considerable discussion of Web server security in recent years. However, much of this has been qualitative, often focused on detection and prevention of individual vulnerabilities. Quantitative data is sometimes cited, but frequently without any significant critical analysis. Methods need to be developed to allow security related risks to be evaluated quantitatively in a systematic manner. A study by Ford et al. (2005) compares several software venders by considering the number of vulnerabilities and severity, and suggests a need to use quantitative approaches for estimating the risks posed by vulnerabilities.

Two of the major software components of the Internet are HTTP (Hyper Text Transfer Protocol) servers (also termed Web servers) and browsers, which serves as clients. Both components were introduced in 1991 by Tim Berners-Lee of CERN (2010). They have now become indispensable parts of both organizational and personal interactions. The early Web servers provided information using static HTML pages. The Web servers now provide dynamic and interactive services to the clients using database queries, executable script, etc. Web servers can support functions such as serving streaming media, email, etc. In the emerging cloud computing systems, the HTTP servers support virtual implementations of applications and operating systems. HTTP servers have thus emerged as a focal point for the Internet.

Here the vulnerabilities in the two most widely-used HTTP servers are examined, the Apache Web server introduced in 1995, and the Microsoft Internet Information Services (IIS)

* *Corresponding author*. Tel.: +1 970 491 1056; fax: +1 970 491 2466.
E-mail addresses: woo@cs.colostate.edu (S.-W. Woo), dean2026@cs.colostate.edu (HyunChul Joh), omar@cs.colostate.edu (O.H. Alhazmi), malaiya@cs.colostate.edu (Y.K. Malaiya).

0167-4048/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2010.10.007

Web server, originally supplied as part of the NT operating systems in 1995–1996. Fig. 1 shows the major versions' timeline for the two Web servers. While Apache has a much larger overall market share, roughly 55% on March 2010, IIS may have a higher share of the corporate Websites. The market share for other servers is very small, and thus they are not examined here. IIS is the only major HTTP server that is not open source. Both Apache and IIS are generally comparable in features. IIS runs only under the Windows operating systems and comes bundled with some of the versions, whereas Apache supports all the major operating systems. Table 1 gives a brief functional comparison between the two.

The security of systems connected to the Internet depends on several components of the system. These include the operating systems, HTTP servers and browsers. Some of the major security compromises arise because of vulnerabilities in the HTTP servers. A vulnerability is defined as a software defect or weakness in the security system which might be exploited by a malicious user causing loss or harm (Pfleeger and Pfleeger, 2003). The vulnerabilities found are disclosed by the finders using some of the common reporting mechanisms available in the field. The databases for the vulnerabilities are maintained by several organizations such as National Vulnerability

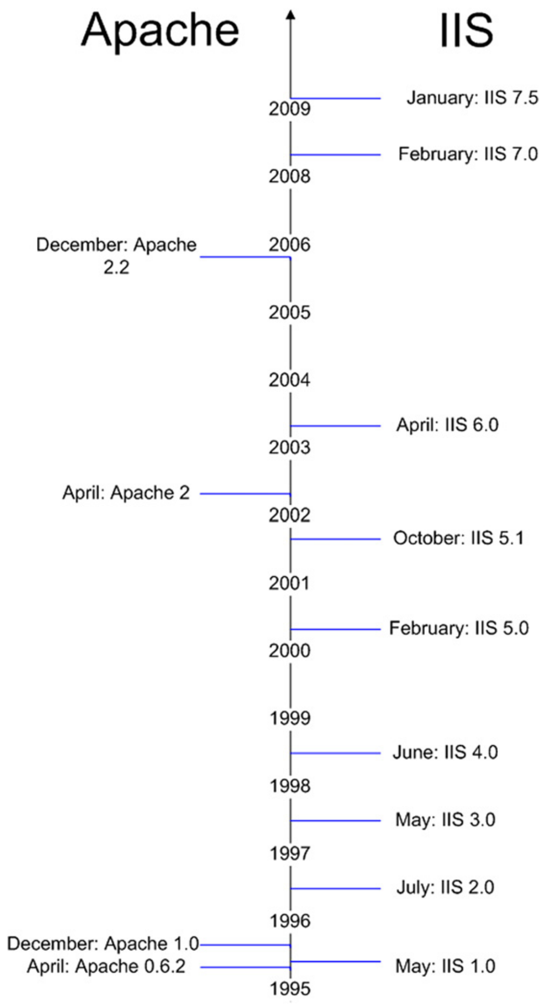


Fig. 1 – Version release timeline for Apache and IIS.

Table 1 – Functional support comparison (IIS vs. Apache, 2010).

Feature	IIS	Apache
ASP	Native	With Chilisoft, Apache::ASP, or modmono
Active directory authentication	Yes	With third-party modules
Live configuration editing	Yes	No
CGI, Perl, Python, PHP, JSP	Yes	Yes
Runs under Windows	Yes	Yes
Runs under Linux, Unix, OS X	No	Yes

Database (NVD, 2010), Open Source Vulnerability Database (OSVDB, 2010), BugTraq (Securityfocus, 2010), etc., as well as the vendors of the software. The exploitations of some of the server vulnerabilities are well known. The Code Red worm (Moore et al., 2002), which exploited a vulnerability in IIS (described in Microsoft Security Bulletin MS01-033, June 18, 2001), appeared on July 13, 2001, and soon spread world-wide in unpatched systems.

All the computing systems connected to the network are subjects to some security risk. While there have been many studies attempting to identify causes of vulnerabilities and potential countermeasures, the development of systematic quantitative methods to characterize security have begun only recently. There has been considerable debate comparing the security attributes of open source and proprietary software (Anderson, 2002). However, for a careful interpretation of the data, rigorous quantitative modeling methods are needed. The likelihood of a system being compromised depends on the probability that a newly discovered vulnerability will be exploited. Thus, the risk is better represented by the vulnerabilities which are not yet discovered and the vulnerability discovery rate rather than by the vulnerabilities that have been already discovered in the past and remedied by patches.

Possible approaches for a quantitative perspective of exploitation trends are discussed by Hallberg et al. (2001). Probabilistic examinations of intrusions have been presented by several researchers (Browne et al., 2001; Madan et al., 2004). Rescorla (2005) has studied vulnerabilities in open source servers. The vulnerability discovery process in operating systems has recently been examined by Rescorla (2003) and by Alhazmi and Malaiya (2005a,b, 2008).

HTTP servers are very attractive targets for malicious attackers. The servers can represent the first line of defense that, if bypassed, can compromise the integrity, confidentiality and availability attributes of the enterprise security. Thus, it is essential to understand the threat posed by both undiscovered vulnerabilities and recently discovered vulnerabilities for which patches have not been developed or applied. Despite the significance of security in the HTTP servers, only limited work has been done related to the vulnerability discovery process for the servers (Woo et al., 2006; Alvarez and Petrovic, 2003). Such work would permit the developers and the users to better estimate future vulnerability discovery rates. It would also be highly desirable to be able to project what types of vulnerabilities are more likely to be discovered.

Some of the available work on HTTP servers discusses some specific problems or attacks that the servers face, such as denial of service attacks (DoS) (Aura et al., 2000; Kargl et al., 2001), in which the authors suggest some countermeasures to be applied when attacks for these types take place. Here, our focus is the discovery rates of vulnerabilities in the two most popular servers, which have undergone significant evolution. Unlike some of the recent studies on the discovery rates in specific operating systems (Alhazmi and Malaiya, 2008), complete data for all the versions of the two servers is examined here. This is an extension of the work reported in Woo et al. (2006).

This paper is organized as follows. The next section introduces the vulnerability discovery models used and discusses the significant factors that affect software vulnerability discovery rates. In Section 3, we consider the aggregate vulnerabilities in the two HTTP servers and examine how the models fit the available data. In Section 4, the datasets are partitioned into categories based on the causes of vulnerabilities, and the applicability of the models to individual categories is considered. In the next section, the vulnerabilities are divided based on the severity levels and the fit is again evaluated. In Section 6, we examine the predictive capabilities of the models using the available datasets. Finally, the major observations are discussed and the conclusion is presented.

2. Vulnerability discovery models

Use of quantitative reliability growth models is now common in the software reliability engineering (Musa, 1999; Lyu, 1995). Software reliability growth models (SRGM) model the testing process when as bugs are found and removed, fewer bugs remain. The bug finding rate gradually drops and the cumulative number of bugs eventually approaches saturation. Such growth models are used to determine when a software system is ready to be released and what future failure rates can be expected.

The Vulnerability Discovery Models (VDM) can be regarded to be related to the reliability growth models since vulnerabilities are a special class of defects or bugs that can permit circumvention of the security measures. Some vulnerability discovery models were proposed by Anderson (2002), Rescorla (2003), Alhazmi and Malaiya (2005a,b), and Joh et al. (2008). Most of these models give the cumulative number of vulnerability by calendar time. These models are classified as the time-based models.

The time-based models consider calendar time as the independent variable. These models incorporate the effect of the rising and declining market share on the software. The other models referred to as the effort-based models, require explicit estimation of the effort using an effort function, which is then used as an independent variable. Fig. 2 shows the classification of vulnerability discovery models. Vulnerability discovery models are separated into the time-based model and effort-based model. The time-based model uses calendar time as the main factor and the effort-based model uses the number of installed system as the main factor. Examples of the time-based models are the Alhazmi–Malaiya Logistic (AML) Model, Anderson Thermodynamic Model, Weibull model, Rescorla Exponential Model and Logarithmic Poisson Model (Alhazmi and Malaiya, 2008; Joh et al., 2008).

The applicability of these models to several operating systems was examined in Alhazmi and Malaiya (2005a,b). The results indicate that while some of the models fit the data for most operating systems well, others do not fit well or just provide a good fit only during a specific phase. Here the applicability of the two most successful models for HTTP servers is investigated. The models used here are the logistic time-based model and the effort-based model proposed by Alhazmi and Malaiya (2005a,b). These two models have been found to fit datasets of the major Windows and Linux operating systems, as determined by goodness of fit and other measures (Alhazmi and Malaiya, 2008; Alhazmi et al., 2007). In the discussion below the Alhazmi–Malaiya Logistic Model is referred to as the time-based model, and Alhazmi–Malaiya effort-based model is termed the effort-based model. Several time-based models have been proposed, however the Alhazmi–Malaiya Logistic Model has selected for analysis and comparison here since it has generally provided a better fit compared with other models (Alhazmi and Malaiya, 2008). The effort-based model has been selected because it is the only model proposed in the literature that uses effort instead of time. The two models contrast two different approaches.

The Alhazmi–Malaiya logistic model: This time-based model assumes that the rate of change of the cumulative number of vulnerabilities Ω is governed by two factors, as given in Equation (1) below (Alhazmi and Malaiya, 2005a,b). The second factor on the right hand side declines as the number of remaining undetected vulnerabilities declines. The other factor increases with the time needed to take into account the rising share of the installed base. The saturation effect is modeled by the second factor. It is possible to obtain a more detailed model. However, this model provides a good fit to the data, as observed below. The model assumes that the vulnerability discovery rate is given by the differential equation

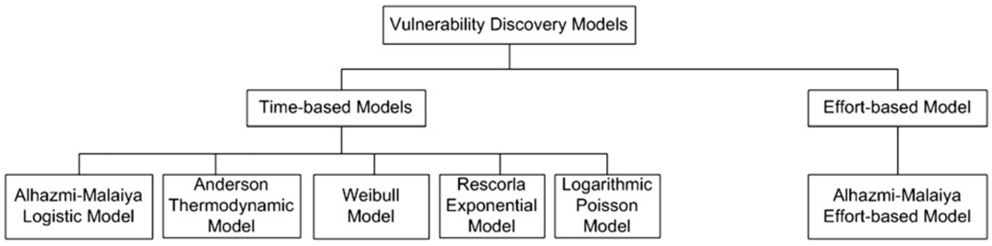

Vulnerability Discovery Models
Time-based Models
Effort-based Model
Alhazmi-Malaiya Logistic Model
Anderson Thermodynamic Model
Weibull Model
Rescorla Exponential Model
Logarithmic Poisson Model
Alhazmi-Malaiya Effort-based Model

Fig. 2 – Vulnerability discovery models.

$$\frac{d\Omega}{dt} = A\Omega(B - \Omega) \tag{1}$$

where Ω is the cumulative number of vulnerabilities, t is the calendar time, and initially $t = 0$. A and B are empirical constants determined from the recorded data. By solving the differential equation, the following equation is obtained:

$$\Omega(t) = \frac{B}{BCe^{-ABt} + 1} \tag{2}$$

where C is a constant introduced while solving Equation (1). Equation (2) gives us a three-parameter model given by the logistic function. In Equation (2), as t approaches infinity, Ω approaches B. Thus, the parameter B represents the total number of accumulated vulnerabilities that will eventually be found.

An S-shaped curve can be plotted using Equation (2). Fig. 3 shows hypothetical plot for the time-based model, which is determined by the values of A, B and C. The two transition points 1 and 2 in the figure can be obtained by equating the second derivative of Equation (2) to zero (Alhazmi and Malaiya, 2006). The two transition points separate the three phases: the learning phase from the release of the system until the first transition point. After the learning phase, a linear phase is observed from the first transition point to the second transition point. The last phase reflects decreasing vulnerability discovery rate that results in saturation. In the first phase, the cumulative number of vulnerabilities shows an increasing rate at the beginning as the system begins to attract an increasing share of the installed base. After some time, there is a steady rate of vulnerability discovery, which yields a linear trend. Eventually, as the vulnerability discovery rate begins to drop, there is saturation. It can be due to a smaller pool of remaining vulnerabilities, or reduced vulnerability finding effort when the attention has been diverted to a newer product.

The original model derivation assumed that the software is stable (Alhazmi and Malaiya, 2005a,b). However, in this case it is being applied to software that has evolved for several years. Software evolution is the process that describes a gradually changing software system. It has been suggested that the software evolution affects the S-shape of the time-based model since new vulnerabilities may get introduced along with the new version of software in the process of evolution. Kim et al. (2007) have considered the impact of software evolution and have suggested that the S-shaped model may still apply while the significance of parameters may change. They have proposed a model that can be applied to two or more successive versions of an evolving software, provided the difference between any two successive versions can be determined. In practice this would require the source codes for each pair of successive versions to be manually examined since not only additions, deletions and changes to the code can occur, but the sections of the same code may get reorganized. In addition, the commercial software systems are generally closed-source. However in many cases, a simple unimodal time-base model may still apply, when modeling involves a sufficiently long period such that shorter term variations are of minor significance.

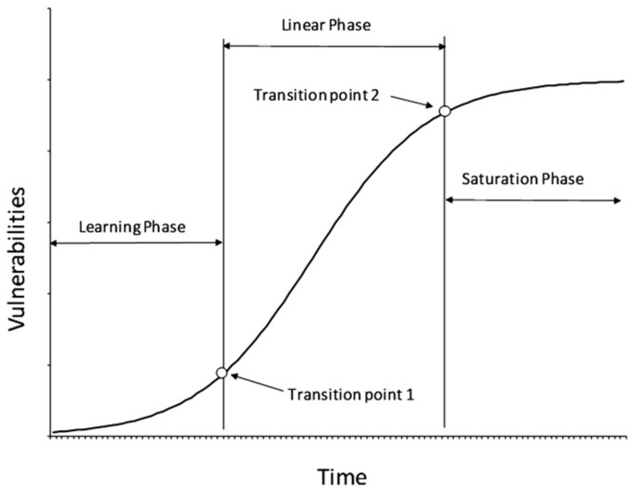


Fig. 3 – AML time-based model.

The Alhazmi–Malaiya effort-based model: Vulnerabilities are usually reported using calendar time, because it is easy to record and link them to the time of discovery. This, however, does not consider the changes occurring in the environment during the lifetime of the system. A major environmental factor is the number of installations, which depends on the share of the installed base of the specific system. It is much more rewarding to find or exploit vulnerabilities that exist in a large number of computers. Hence, it can be expected that a larger share of the effort going into the discovery of vulnerabilities, both internal and external, would go toward a system with a larger installed base.

Using effort as a factor was first discussed in Alhazmi and Malaiya (2005a,b). It is based on the assumption that the vulnerability finding effort is proportional to actual usage, as given by the total number of installed systems. The effort-based model utilizes a measure termed an *equivalent effort* (E), which is calculated using (Woo et al., 2006)

$$E = \sum_{i=0}^{n}(U_i \times P_i) = \sum_{i=0}^{n} N_i \tag{3}$$

where U_i is the total number of installations of the HTTP servers at the period of time i (Netcraft, 2010), n represents the last usage period, and P_i is the percentage of the servers using the specific software for measuring E. N_i is the number of machines running the specific server during time i. The result is obtained in terms of system-months. The measure U_i can be calculated using the data about total number of web servers. Equivalent effort reflects the effort that would have gone into finding vulnerabilities more accurately than using time alone. This is somewhat analogous to using CPU time for SRGMs (Musa, 1999).

If the vulnerability detection rate with respect to the effort is proportional to the fraction of remaining vulnerabilities, then an exponential model like the exponential SRGM can be applied. This model can be expressed as follows:

$$\Omega(E) = B\left(1 - e^{-\lambda_{vu}E}\right) \tag{4}$$

where λ_{vu} is a parameter analogous to failure intensity in the SRGMs and B is another parameter which represents the number of vulnerabilities that will eventually be found. The model given by Equation (4) will be referred as the effort-based model. Several factors impact the parameters of the vulnerability discovery models. These include code size, software age and popularity examined below.

Code Size of Software: Several studies (Fenton and Rafail, 1990; Hatton, 1997) have examined the relationship between the code size and number of defects. The studies show that the number of defects or errors increases as code size increases. A first order approximation assumes a linear relationship, which allows a measure defect density to be defined. Since the vulnerabilities are a class of defects, it can be similarly defined as a measurement called *vulnerability density* (Alhazmi and Malaiya, 2005a,b). Available data allows us to calculate the densities of the discovered vulnerabilities for some of the major software systems, as given in Section 7.

Software Age: The time-based models model this more explicitly. In the effort-based model, it would be modeled by the specific SRGM used.

Market Share: This is modeled more explicitly in the effort-based model. The effect of the market share rise and fall is implicit in the AML model.

3. Aggregate vulnerabilities in HTTP server

In this section, the datasets for the total vulnerabilities of the Apache and IIS Web servers are fitted to the models. The goodness of fit is evaluated to determine how well the models reflect the actual vulnerability discovery process. The vulnerability datasets, here, are from NVD. NVD is maintained by National Institute of Standards and Technology, and sponsored by the department of Home Land Security, thus it can be considered to be a more reliable source. The market share data from Netcraft (2010) was used. Note that Apache represents an open source software system, whereas IIS represents a proprietary closed-source system. In this section, all vulnerabilities are considered without regard to how they arise or the extent of their impact.

Market share is one of the most significant factors impacting the effort expended in exploring potential vulnerabilities. Higher market share indicates more incentive to explore and exploit vulnerabilities for both exports and non-exports, since both would find it more profitable or satisfying to spend their time on a software system with a higher market share.

Table 2 presents data obtained from NVD and Netcraft in January 2009, showing the current Web server market share and total number of vulnerabilities found to date. Google Web server is omitted from the table although its market share is approximately 5%, because very little information about Google Web server is publically available. For servers with a lower percentage of the market, such as Google Web server, Nginx and Lighttpd, the total number of vulnerabilities found is zero or very few. However, that does not mean that these systems are more secure, but merely that only limited effort has gone into finding their vulnerabilities. A significant number of vulnerabilities has been found in both Apache and IIS, illustrating the impact of the market share on the motivation for exploring or finding vulnerabilities. Here, the market share is used as an indicator of effort for the effort-based model.

Fig. 4 shows the Web server market share for Apache and IIS. As demonstrated by the figure, the number of Web servers continues to grow steadily. Among the various Web servers, Apache and IIS dominate the Web server market. Other Web servers such as Nginx and Lighttpd occupy a very small share of the market, as shown in Table 2. Since the total share of all the Web servers, except Apache and IIS, represents less than 15% of the market share, few vulnerabilities have been found in them and hence the data for these servers has not been used here.

There is a marked gap between the Apache and IIS market shares, as shown in Fig. 4. This difference in market share may be due to several factors. Perhaps the most important of these is that Apache is available for all major operating system platforms and can be obtained without cost. Apache may also have benefited from not having been exposed to serious security issues such as the Code Red (Moore et al., 2002) or Nimda worms (Machie et al., 2001) that were faced by IIS in 2001.

Since its release in 1995, Apache HTTP server has achieved and maintained a large installed base and was used by over 90 million Web server systems during January 2009. In this section, the vulnerability dataset for Apache is fitted to the time-based and the effort-based models. Figs. 5 and 6 give the vulnerability data from NVD for the period between January 1996 and December 2008. Netcraft provides the market share data covers this time period.

In Figs. 5 and 6, the solid lines indicate the fitted models, while the X marks show cumulative vulnerabilities for the servers. Fig. 5(a) shows cumulative vulnerabilities by month for the time-based model. The slope of the curve for Apache rises gently until about January 2000, after which the slope had remained steady until the end of 2007. From the point of the

Table 2 – Market share and vulnerabilities found.

Web server	Apache	IIS	Nginx	Lighttpd	Other
Market share	52.26%	32.91%	1.87%	1.61%	11.35%
Vulnerabilities	132	137	0	18	N/A
Release year	1995	1995	2005	2003	N/A
Latest version	2.2.11	7.5	0.6.35	1.4.20	N/A

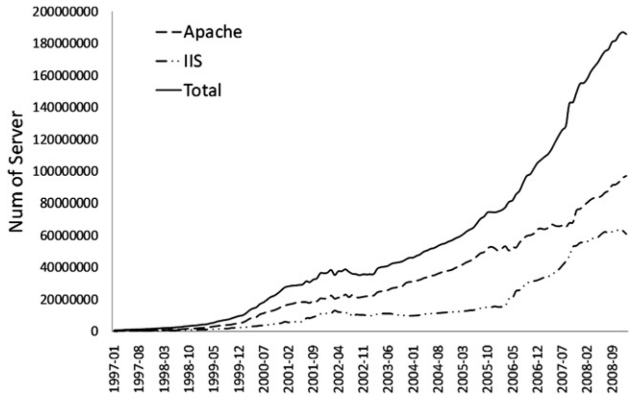


Fig. 4 – Web server market share trends.

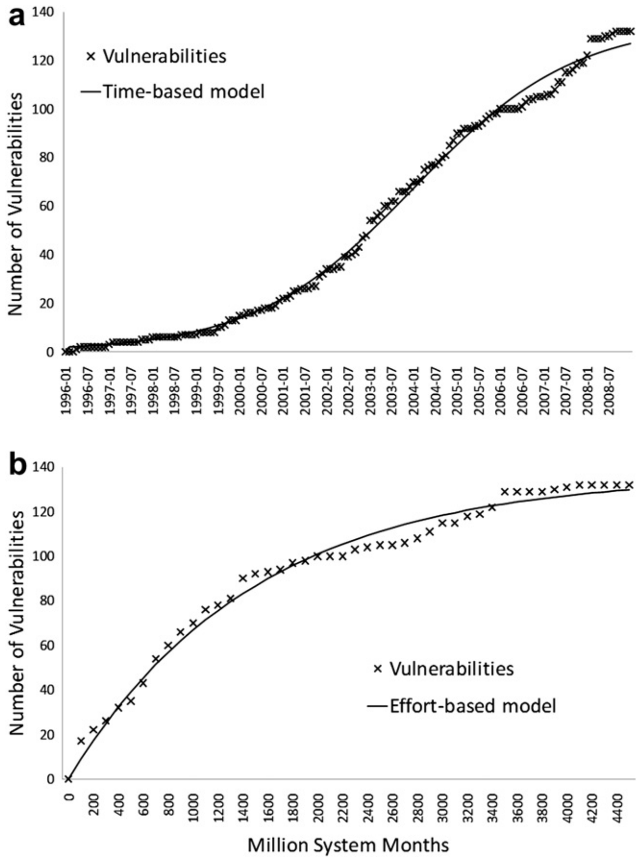


Fig. 5 – Fitting Apache aggregated vulnerability data. (a) Time-based model; (b) Effort-based model.

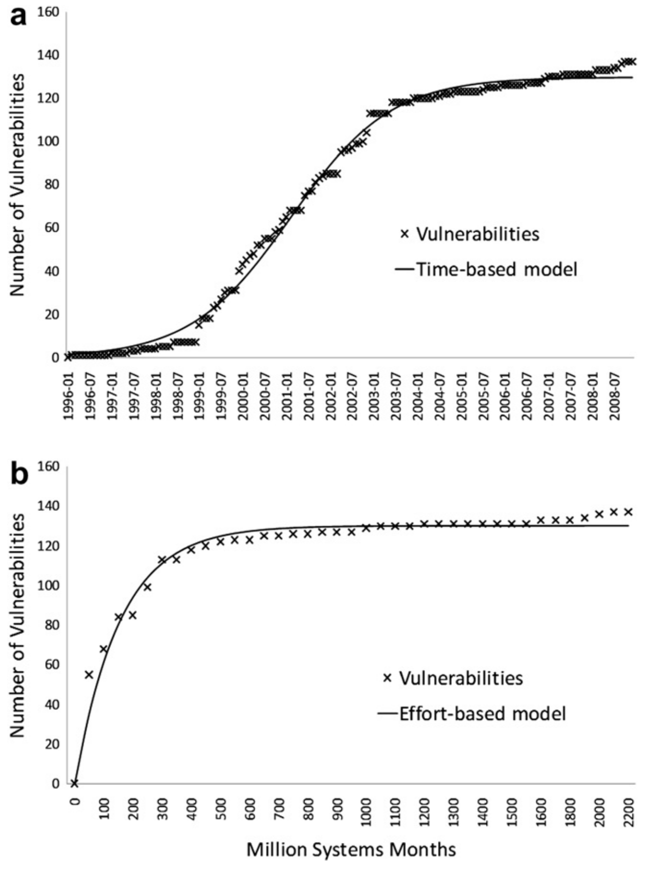


Fig. 6 – Fitting IIS aggregated vulnerability data. (a) Time-based model; (b) Effort-based model.

three phases of the vulnerability discovery process (Alhazmi and Malaiya, 2005a,b), Apache may be entering the saturation phase, since only three vulnerabilities were found in 2008.

Fig. 5(b) shows cumulative vulnerabilities by the number of Apache installations in terms of million system-months and the fitted effort-based model. This effort-based model also shows that Apache is approaching the saturation phase since any vulnerability has not been found after 4000 million system-months as the number of Apache severs increases.

IIS, released in 1995, is the only major proprietary Web server with over 60 million installations during January 2009. The vulnerability dataset from January 1996 to December 2008 is used in this analysis.

Fig. 6(a) shows the cumulative number of vulnerabilities by month and the fitted time-based model for the IIS Web server. The time-based and effort-based models fit the data for IIS very well. The IIS Web server appears to have reached the saturation phase, since 2004 the vulnerability finding rate has been low. There was a recent increase because of the six vulnerabilities found during 2008. A possible explanation for this could be that the number of IIS Web servers installed has increased since 2006 and a new version of IIS was released in February 2008.

Fig. 6(b) shows the cumulative number of vulnerabilities for the IIS server and the effort-based model by million system-months. The figure shows a significant degree of saturation.

The fit of the models to the data as shown in Figs. 5 and 6 have been examined. Table 3 gives the χ^2 (chi-square) values and parameter values for both the time-based and effort-based models. For χ^2 goodness of fit test, the alpha level is 5%. For comparison, the corresponding parameter values are also provided for the Windows 98 and NT operating systems, as well as the chi-square values.

Table 3 shows that the chi-square values are less than the critical values. This demonstrates that the fit for Apache, IIS, Windows 98 and NT is significant for the both models with *P*-values ranging from 0.64535 to 1, indicating that the fit is statistically significant. It is observed that parameter *A* is always less than 0.005 and parameter *C* is also less than 0.71, while parameter *B* corresponds approximately to the number of vulnerabilities.

4. Vulnerability categories

In the previous section, the application of the two models for the total number of vulnerabilities of Apache and IIS has been examined. Here, the models applied to portioned data using a classification scheme for server vulnerabilities.

Distinction among vulnerabilities is useful when developers want to examine the nature and extent of the problem. It can help determine what kinds of protective actions would be most effective. Vulnerability taxonomy is still an evolving

Table 3 – Chi-square goodness of fit test results for total vulnerabilities.

	Time-based model						Effort-based model				
	A	B	C	χ^2	χ^2_{critical}	P-value	B	λ_{vu}	χ^2	χ^2_{critical}	P-value
Apache	0.0003	135.57	0.5745	24.69	186.1458	1	136.50	0.0007	16.224	61.65623	0.99996
IIS	0.0005	129.74	0.7034	79.82	186.1458	1	130.13	0.0064	14.688	55.75848	0.99961
Windows NT	0.00018	253.3	0.1293	136.14	173.0041	0.64535	N/A	N/A	N/A	N/A	N/A
Windows 98	0.0004	100.94	0.1002	81.11	114.2679	0.99688	N/A	N/A	N/A	N/A	N/A

area of research. Several taxonomies have been proposed (Aslam and Spafford, 1996; Bishop, 1999; Landwehr et al., 1994; Seacord and Householder, 2005; Gopalakrishna et al., 2005; Venter et al., 2008). An ideal taxonomy should have such desirable properties as mutual exclusiveness, clear and unique definition, and coverage of all software vulnerabilities.

Vulnerabilities can be classified using schemes based on cause, severity, impact and source, etc. In this analysis, the classification scheme is used employed by the NVD of the National Institute of Standards and Technology. This classification is based on the causes of vulnerabilities. The eight classes are as follows (NVD, 2010; OSVDB, 2010):

1. *Input Validation Error (Boundary condition error, Buffer overflow)*: Such types of vulnerabilities include failure to verify the incorrect input and read/write involving an invalid memory address.
2. *Access Validation Error*: These vulnerabilities cause failure in enforcing the correct privilege for a user.
3. *Exceptional Condition Error*: These arise due to failures in responding to unexpected data or conditions.
4. *Environmental Error*: These are triggered by specific conditions of the computational environment.
5. *Configuration Error*: These vulnerabilities result from improper system settings.
6. *Race Condition Error*: These are caused by the improper serialization of the sequences of processes.
7. *Design Error*: These are caused by improper design of the software structure.
8. *Others*: Includes vulnerabilities that do not belong to the types listed above, sometimes referred to as nonstandard.

Unfortunately, the eight classes are not completely mutually exclusive. Because a vulnerability can belong to more than one category, the summation of all categories for a single software system may add up to more than the total number of vulnerabilities (also the percentages may exceed 100) (Gopalakrishna and Spafford, 2005).

Fig. 7 compares vulnerability distributions in Apache and IIS. The categories with the highest numbers are input validation errors, followed by design errors. There is a slight difference in category ordering between Apache and IIS, with Apache having more configuration errors than access validation errors; however, IIS has more access validation errors. While IIS has been more vulnerable to access validation errors, the fact that Apache has been more vulnerable to configuration errors may be due to Apache's more complex installation requirements.

To determine whether there is an observable pattern at the level of individual classes, the vulnerabilities are plotted for the major categories. Since a similar pattern for the uncategorized vulnerabilities is noted, a possible fit was examined. Figs. 8 and 9 show the fit for the IIS and Apache respectively. It may be noted that the number of input validation errors and design errors are the most common category in Apache and IIS. We chose to fit the categories which had enough data points available for fitting. In the figures, these three major categories are shown: input validation errors, design errors and exceptional handling condition errors.

The categorized number of vulnerabilities shows the same pattern as demonstrated by the total number of vulnerabilities. Thus, each category shows a related pattern with regard to the total number of vulnerabilities. Time-based and effort-based models are fitted for each category. Table 4 shows the chi-square goodness of fit tests for the Apache and IIS models by category. The table demonstrates that the chi-square values for each category are less than the corresponding chi-square critical values and the *P*-values are close to 1. The fits for input validation, design and exceptional condition error classes are significant for both models.

5. Vulnerability severity level

Severity is another way of classifying vulnerabilities. The severity level of a vulnerability indicates how serious the impact of an exploitation can be. Three severity levels are often defined; high, medium and low. Some other organizations such as Secunia (2010) use three to five levels and use their own definition for severity. The NVD of the National Institute of Standards and Technology has used Common

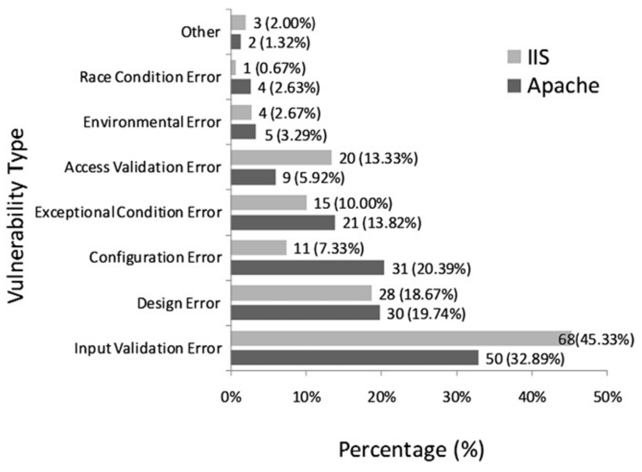


Fig. 7 – Vulnerabilities by type.

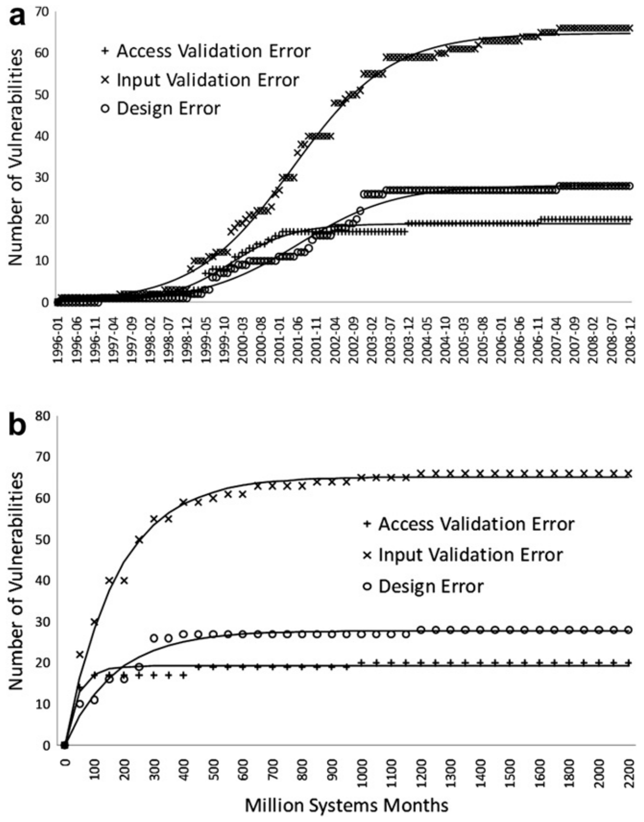


Fig. 8 – **Fitting IIS vulnerabilities by category. (a) Time-based model; (b) Effort-based model.**

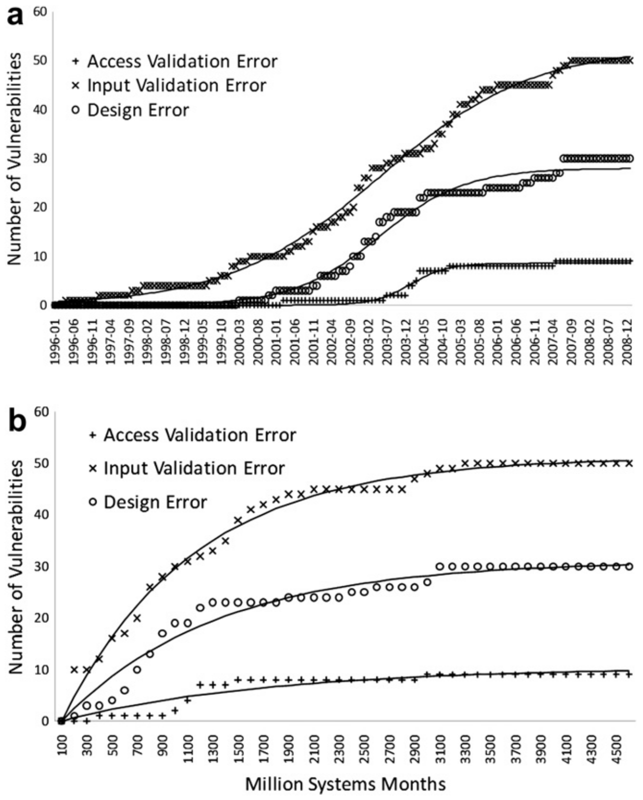


Fig. 9 – **Fitting Apache vulnerabilities by category. (a) Time-based model; (b) Effort-based model.**

Vulnerability Scoring System (CVSS) (First of Incident, 2010) metric for vulnerability severity with ranges from 0.0 to 10.0; CVSS uses many factors to determine the severity where the range from 0.0 to 3.9 corresponds to low severity, 4.0 to 6.9 to medium severity and 7.0 to 10.0 to high severity. The NVD (2010) describes three severity levels as follows:

1. *High Severity*: vulnerabilities make it possible for a remote attacker to violate the security protection, or permit a local attack that gains complete control, or are otherwise important enough to have an associated CERT/CC advisory or US-CERT alert.
2. *Medium Severity*: vulnerabilities are those not meeting the definition of either 'high' or 'low' severity.
3. *Low Severity*: vulnerabilities typically do not yield valuable information or control over a system but may provide the attacker with information that may help him find and exploit other vulnerabilities or may be inconsequential for most organizations.

The distributions of the severity levels of the Apache and IIS vulnerabilities show similarity. About 60–70% of total vulnerabilities are the medium severity, followed by about 30–40% with high severity, with low severities at about 4–6%.

The time-based and effort-based models have been applied to the three severity classes. In Figs. 10 and 11, the solid lines indicate the fitted the time-based and effort-based models for each severity level. The two figures show the result of fitting the time-based and effort-based models to the three severity classes. The plots suggest that especially for the medium severity vulnerabilities, the IIS vulnerability data appears to be in the saturation phase while the Apache vulnerabilities are still being discovered.

Table 5 shows the chi-square goodness of fit tests and the parameter values for Apache and IIS by severity level. The parameter values are obtained from data corresponding to Figs. 10 and 11 using regression analysis. As before, for chi-square goodness of fit test, the test alpha level is 5%. This chi-square test shows that the fits for the three severity categories are significant, and the chi-square tests show that the vulnerabilities classified by severity datasets fit the model well.

The fraction of high and medium severity vulnerabilities is substantial and presents a significant risk to the HTTP servers potentially leading to problems such as unauthorized system access, denial of service (DoS) attack, exposure of sensitive information, etc. Fig. 12 plots the percentage of the cumulative number of vulnerabilities for each severity class for each month for Apache and IIS. Both Apache and IIS show a similar pattern. Note that the first few points in the plots are not significant, since they represent only a small number of vulnerabilities. The plots suggest that a larger fraction of the high severity vulnerabilities is found early, while the medium severity vulnerabilities represent about 70% share after three or four years. This data suggests that there may be a deliberate effort to focus on high severity vulnerabilities in early phase. This is supported by the observations about the patching rate

Table 4 – Chi-square goodness of fit test results for vulnerabilities by category.

		Time-based model						Effort-based model				
		A	B	C	χ^2	χ^2_{critical}	P-value	B	λ_{vu}	χ^2	χ^2_{critical}	P-value
Apache	Input validation	0.00036	52.75307	1.342086	19.87597	186.1458	1	51.2449	0.000958	0.006129	61.65623	0.99998
	Design error	0.00064	27.99523	67.37607	27.92677	186.1458	1	31.1015	0.000817	0.003253	61.65623	0.99999
	Access validation	0.001474	8.554207	22060622	120.5440	186.1458	1	10.4792	0.0006	0.06152	61.65623	1
IIS	Input validation	0.000559	64.79746	1.964722	24.007	186.1458	1	65.20718	0.005732	3.242342	55.75848	1
	Design error	0.000887	18.89434	15.97189	81.139	186.1458	1	19.24239	0.022189	2.060538	55.75848	1
	Access validation	0.000574	28.06895	5.850765	31.74241	186.1458	1	27.79932	0.005897	3.050723	55.75848	1

of input validation errors which tend to have higher severity levels. Di Penta et al. (2009) have empirically shown that buffer overflows are patched significantly faster than other types of vulnerabilities because they represent the kind to which the developers tend to respond faster.

6. Predictive capability of models

Even when for a VDM shows the nice goodness of fit during the period covered, it is possible that the model may not be able to predict well in the future if the model does not anticipate changes in the trend that is actually encountered. In the software reliability engineering field, the predictive capability for a number of reliability growth models has been investigated in the past (Musa and Okumoto, 1984; Malaiya et al., 1992). A VDM having a good predictive capability should be able to estimate the future behavior, for example total number of vulnerabilities using currently available datasets. It can be used to estimate the resources needed for maintenance and the risk estimation.

Here, the starting point for comparing the two models is chosen to be when cumulative installations exceed 100 million and 50 million for Apache and IIS Web servers respectively, since only after some significant past data, the effort-based model can project the future trend. The predictive capabilities for the two models can be comparable only when the two models have data points for the same specific time points. Since the time-based model has many more data points compared with the effort-based model, the models are applied for the calendar time with somewhat unequal interval of months during the estimations. For each time point, the available partial data at the point are fitted to the models using regression analysis to estimate model parameters. Then the

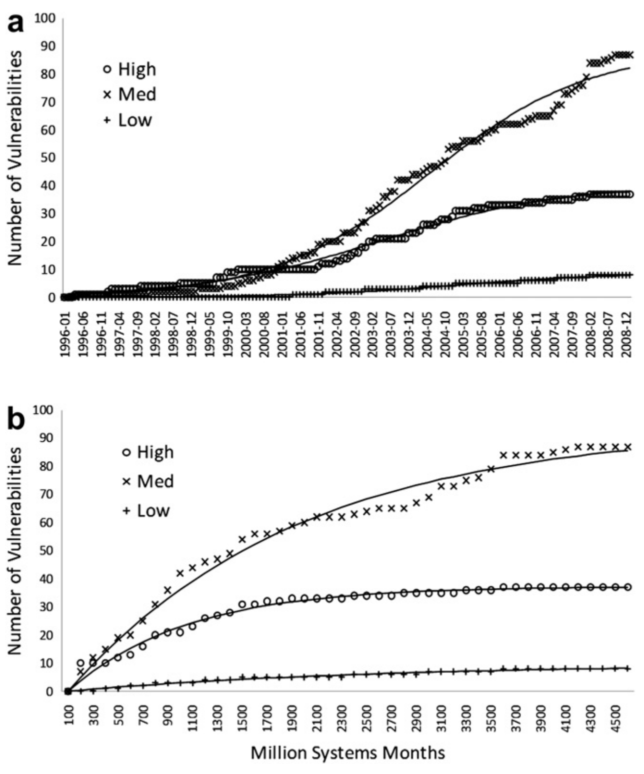


Fig. 10 – Fitting Apache vulnerabilities by severity level. (a) Time-based model; (b) Effort-based model.

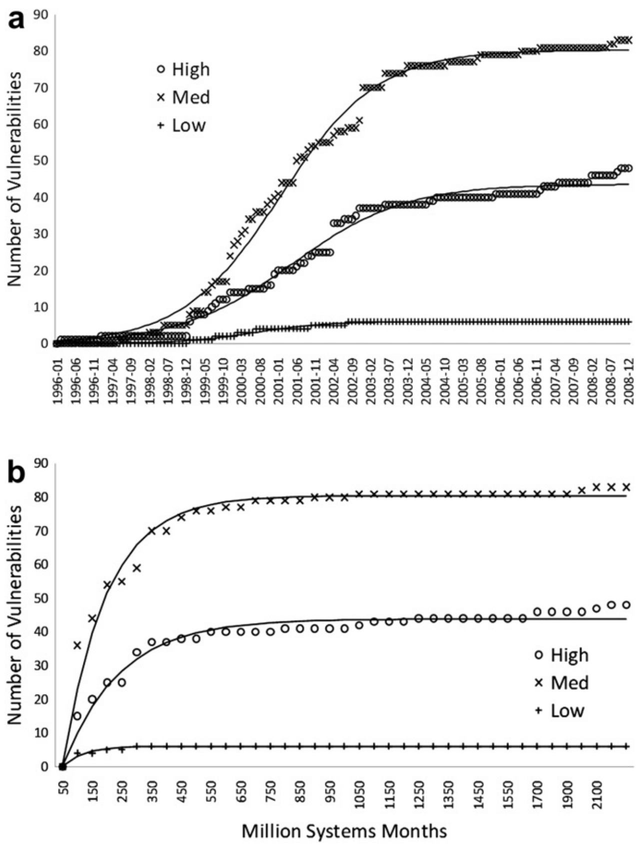


Fig. 11 – Fitting IIS vulnerabilities by severity level. (a) Time-based model; (b) Effort-based model.

Table 5 – Chi-square goodness of fit test results for vulnerabilities by severity level.

		Time-based model						Effort-based model				
		A	B	C	χ^2	$\chi^2_{critical}$	P-value	B	λ_{vu}	χ^2	$\chi^2_{critical}$	P-value
Apache	High	0.000337	135.575	0.574563	28.85319	186.1458	1	37.32424	0.001061	12.88105	61.65623	0.9999
	Medium	0.000298	40.56143	0.805567	55.79249	186.1458	1	93.90457	0.000541	9.335251	61.65623	1
	Low	0.000339	88.93068	1.092679	24.48047	186.1458	1	9.835684	0.0004	2.359279	61.65623	0.9999
IIS	High	0.000542	129.7421	0.703459	32.68021	186.1458	1	43.83469	0.005239	6.066258	55.75848	0.9996
	Medium	0.000487	43.63152	1.577562	72.58211	186.1458	1	80.51197	0.006809	9.343797	55.75848	1
	Low	0.000562	80.2965	1.271963	10.52017	186.1458	1	5.987127	0.014206	0.460018	55.75848	0.9999

models with the estimated parameters for each time point are used to predict the final number of vulnerabilities at the end of the time period. The estimated final values for each time point are compared with the actual numbers of vulnerabilities to calculate normalized estimation error rate.

We use prediction error (PE) as a metric for comparison (Malaiya et al., 1992). PE is a measure of how well a model predicts throughout the test phase, and indicates the general bias of the model, whether the model tends to overestimate or underestimate relative to the reality. Prediction error PE is given by

$$PE = \frac{1}{n}\sum_{t=1}^{n}\frac{\Omega_t - \Omega}{\Omega} \quad (5)$$

where n is total number of data points during the prediction period, and Ω is the actual number of vulnerabilities whereas Ω_t is the estimated final number of vulnerabilities at time t. The normalized errors ($(\Omega_t - \Omega)/\Omega$) of the estimated values for the two Web servers are shown in Fig. 13(a) and (b). The PE values are given in Table 6, which suggests that the VDMs tend to underestimate Ω.

In all cases, the prediction error approaches zero as more and more of the data becomes available. For the Apache Web server, in Fig. 13(a), the effort-based model yields a lower prediction error than the time-based model. Also the effort-based model stabilizes to the 0% error value line faster. The time-based model shows a somewhat similar pattern with the effort-based model but a bit less stable. For the IIS, the effort-based model and the time-based model both yield generally similar prediction patterns each other in Fig. 13(b). However it needs to be kept in mind that the effort-based model requires the use of market share data which may not always be readily available. In some cases, the effort variation assumed by the AML model is consistent with real data usage. However, in other cases, it is possible that usage may vary in a different

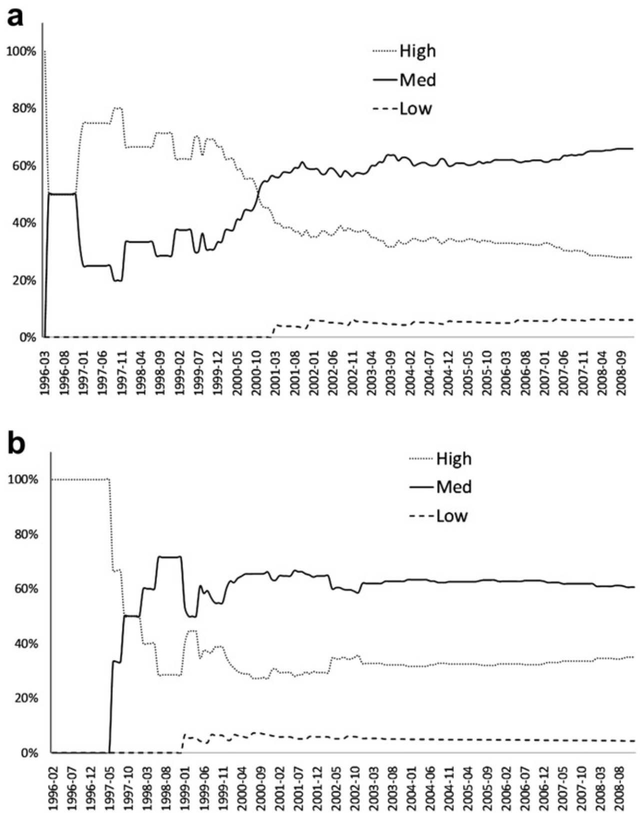


Fig. 12 – Severity of vulnerabilities detected. (a) Apache; (b) IIS.

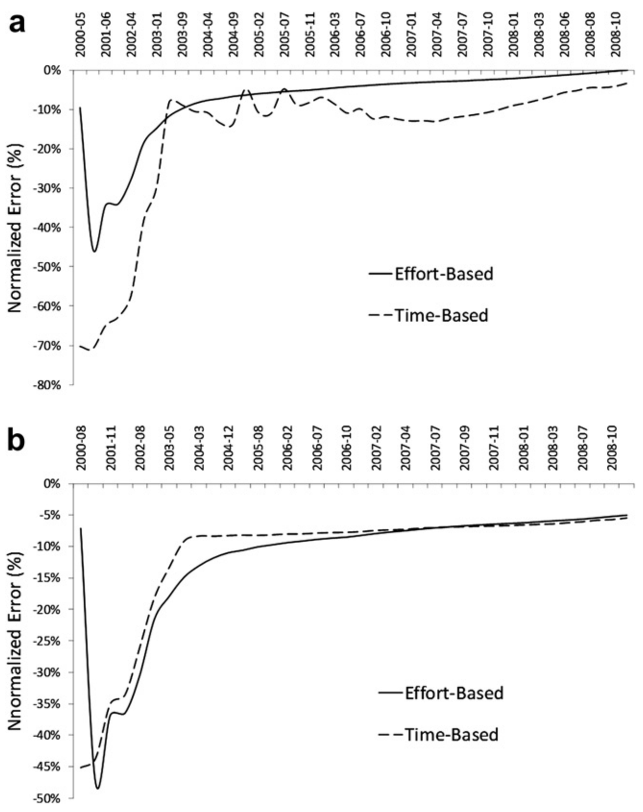


Fig. 13 – Prediction error. (a) Apache; (b) IIS.

Table 6 – Prediction error.

	Effort-based model	Time-based model
Apache	−0.0749	−0.1665
IIS	−0.1177	−0.1153

manner or unpredictably. In such situations, explicitly use of the effort variation, as measured by the installed base, may provide more accurate predictions.

Since the models generally tend to underestimate, it is likely that a recalibration (Brocklehurst et al., 1990) can further improve the predictive capability.

7. Discussion

When the total number of vulnerabilities is examined, both the time-based and effort-based models fit the datasets well, even when the vulnerabilities are categorized by type. This suggests that the models can be used to estimate the number of vulnerabilities expected to be discovered in a given period, and which types are likely to dominate.

The results of model fitting for the vulnerabilities classified by type are shown in Table 4. The fitting was done for the most common types of vulnerabilities for which the available data points are enough to be statistically significant.

The effort-based model requires the number of users for target products in market share, which may be difficult to obtain. The time-based model does not require this data; it can therefore be a feasible alternative when market share data is unavailable.

Static analysis has been used in software reliability engineering, where some of the systems' attributes are estimated empirically even before testing begins. Similar static analysis can be carried out by utilizing metrics such as software size and estimated number of total defects. These methods can potentially be used to estimate Defect density (D_{KD}) and Vulnerability density (V_{KD}) as follows:

$$D_{KD} = \frac{\text{Known Defects}}{\text{Ksloc}} \tag{6}$$

$$V_{KD} = \frac{\text{Known Vulnerabilities}}{\text{Ksloc}} \tag{7}$$

D_{KD}, defects per thousand lines of code, and V_{KD}, vulnerabilities per thousand lines of code, can then be used to estimate the total number of vulnerabilities of a comparable system. Table 7 shows the major attributes of the Apache server and two other major operating systems for comparison. Unfortunately, some of the important metrics for the Microsoft IIS server are not available. For proprietary systems, such data can be hard to obtain outside of the developing organization.

The sizes of IIS and Apache may be comparable in terms of Source Lines Of Code (SLOC) numbers since both offer the same features. The code size for Apache was determined using the SLOCCount (2010) tool. Source code size of Windows 98 and NT 4.0 is given by McGraw (2003). The Apache 2.2.11 source code size for Windows is 240 Ksloc, smaller than for Unix version of Apache 2.2.11 of 373 Ksloc. A few of the Apache vulnerabilities may be applicable to only a specific platform. In Table 7, vulnerability density values for the Windows operating systems are significantly less than for Apache. This may be due to the fact that Windows operating systems have large segments that do not play a role in accessibility, while servers are smaller and therefore vulnerabilities are more concentrated in the code. This assumption is supported by the fact that the defect density to the vulnerability density ratio is higher in Windows NT 4.0, a server operating system, than in Windows 98, a client operating system. Note that V_{KD}/D_{KD} ratios are within a narrow range.

The data for IIS suggests that the vulnerability discovery rate has slowed down significantly since 2004. However, several vulnerabilities were found in IIS in 2008. This may be caused by a new version of IIS being released and the expansion of IIS market share. Factors such as patch releases, number of remaining vulnerabilities, economic aspects etc., also need to be considered when evaluating Web servers.

One interesting fact is that Apache, IIS and SUN Web servers share one common vulnerability (CVE-2008-2579), even though the three software systems do not share any source code. The vulnerability is unspecified in the Oracle WebLogic Server (2010) Plug-in for the Web servers' component in some Oracle BEA Systems (2010) product suite. The vulnerability allows unauthorized disclosure of information, modification, and disruption of service.

The results show that the two models are found to be applicable even though the two software systems have gone through a number of successive versions. Kim et al. (2007) have examined the sharing of the code among successive versions for some software systems and have suggested that for evolving software the overall affect may be explained by a superposition of the trends for vulnerabilities for each individual version.

It should also be noted that the number of vulnerabilities, either found or estimated as remaining, should not be the only

Table 7 – Known D_{KD} vs. known V_{KD}.

Application	Ksloc	Known defects	DKD	Known vulnerabilities	VKD	Ratio
V_{KD}/D_{KD}						
Apache	373 (Unix)	1380	3.699	132	0.353	0.0954
IIS	N/A	N/A	N/A	137	N/A	N/A
Win 98	16,000	10,000	0.625	91	0.0057	0.0091
WinNT 4.0	18,000	10,000	0.556	197	0.0109	0.0197

measurement of a security threat. Factors such as patch development and application delays and vulnerabilities' exploitation rates also need to be considered.

8. Conclusions and future work

Here, the applicability of quantitative models for the number of vulnerabilities and vulnerability discovery rates for the two most popular HTTP servers are explored. Results demonstrate that the vulnerability discovery in the Web servers follows certain patterns, which can be modeled. The results show that when all the vulnerabilities are examined, both models fit the datasets well. The models were found to provide significant fit even when vulnerabilities are categorized by cause or severity levels. This suggests that the models can be used to estimate not only the number of vulnerabilities expected to be discovered but also the likely distribution in terms of categories by origin and severity levels.

It was observed that a number of input validation error vulnerabilities can be large which would constitute a significant risk. The results can be used to optimize the distribution of testing and patch development effort by allocating more effort to vulnerabilities from classes that represent a higher risk, thereby reducing the overall risk due to vulnerabilities.

The results indicate that the models originally proposed for operating systems are also applicable to HTTP servers. These models can be used to estimate vulnerability discovery rates, which can be integrated with risk assessment models in the future (Sahinoglu, 2006). Furthermore, these models can be used to optimize the development and maintenance process to achieve more secure software systems (Seacord, 2005). Also these models for vulnerability discovery are also applicable for when applied to a subset of vulnerabilities of the same severity attribute or belonging to the same category. This can used to decide how and where effort should be allocated to develop patches quickly as the vulnerabilities are discovered. Also developers can specifically focus on types of vulnerabilities that are most likely or significant to reduce the probability of generating them. The methods can also be used to optimally test the patches or additional code. For instance, currently the web servers have more input validation error vulnerabilities. Developers then can choose to allocate more effort on reducing the input validation error vulnerabilities. The users of the software systems can use these models to evaluate their risk and to come up with optimal patch application strategies.

The results demonstrate the applicability of the two models for software products that have undergone evolution. However, the methods in this study do not make use of detailed information on evolution that may be available. Further research is needed to evaluate the impact of evolution of software products that go through many versions by explicitly or implicitly considering the shared code, vulnerabilities inserted and removed in the process and the impact on resource allocation for testing and patch development. Use of a recalibration may further improve the predictive capabilities. Since the web-related software systems are not likely to stop evolving soon, their security attributes will need to be reassessed time to time.

References

Alhazmi OH, Malaiya YK. Quantitative vulnerability assessment of system software. In: Proc. annual reliability and maintainability symposium; Jan. 2005a. p. 615–20.

Alhazmi OH, Malaiya YK, Ray I. Security vulnerabilities in software systems: a quantitative perspective. In: Proc. Ann. IFIP WG11.3 working conference on data and information security; Aug. 2005b. p. 281–94.

Alhazmi OH, Malaiya YK. Prediction capability of vulnerability discovery process. In: Proc. reliability and maintainability symposium; Jan. 2006. p. 86–91.

Alhazmi OH, Malaiya YK. Application of vulnerability discovery models to major operating systems. IEEE Transactions on Reliability; March 2008:14–22.

Alhazmi OH, Malaiya YK, Ray I. Measuring, analyzing and predicting security vulnerabilities in software systems. Computers & Security May 2007;26(3):219–28.

Alvarez G, Petrovic Slobodan. A new taxonomy of web attacks suitable for efficient encoding. Computers & Security July 2003;22(5):435–49. doi:10.1016/S0167-4048(03)00512-1. ISSN: 0167-4048.

Anderson R. Security in open versus closed systems—the dance of Boltzmann, Coase and Moore. In: Conf. on open source software: economics, law and policy; 2002. p. 1–15.

Aslam T, Spafford EH. A taxonomy of security faults. Technical report. Carnegie Mellon; 1996.

Aura T, Bishop M, Sniegowski D. Analyzing single-server network inhibition. In: Proceedings of the 13th IEEE computer security foundations workshop; Jul. 2000. p. 108–17.

BEA Systems, http://www.bea.com; April 2010.

Bishop M. Vulnerability analysis: an extended abstract. In: Proc. second international symposium on recent advances in intrusion detection; Sep. 1999. p. 125–36.

Brocklehurst S, Chan PY, Littewood B, Snell J. Recalibrating software reliability models. IEEE Transactions on Software Engineering 1990;16(4):456–70.

Browne HK, Arbaugh WA, McHugh J, Fithen WL. A trend analysis of exploitations. In IEEE symposium on security and privacy; 2001. p. 214–29.

Conseil Européen pour la Recherche Nucléaire (CERN), http://info.cern.ch/; April 2010.

Di Penta M, Cerulo L, Aversano L. The life and death of statically detected vulnerabilities: an empirical study. Information and Software Technology October 2009;51(10):1469–84.

Fenton NE, Rafail J. Prediction and control of ADA software defects. Journal of Systems and Software; July 1990:199–207.

First of incident response and security teams (FIRST.org, Inc.), Common vulnerability scoring system (CVSS), http://www.first.org/cvss/; April 2010.

Ford R, Thompson H, Casteran F. Role comparison report—web server role. Technical Report. Security Innovation; 2005.

Gopalakrishna R, Spafford EH. A trend analysis of vulnerabilities. CERIAS, Purdue University; May 2005. CERIAS TR 2005-05.

Gopalakrishna R, Spafford EH, Vitek J. Vulnerability likelihood: a probabilistic approach to software assurance. Technical Report. CERIAS; 2005.

Hallberg J, Hanstad A, Peterson M. A framework for system security assessment. In: Proc. 2001 IEEE symposium on security and privacy; May 2001. p. 214–29.

Hatton L. Reexamining the fault density-component size connection. IEEE Software; March 1997:89–97.

IIS vs. Apache, Looking Beyond the Rhetoric, http://www.serverwatch.com/tutorials/article.php/3074841; April 2010.

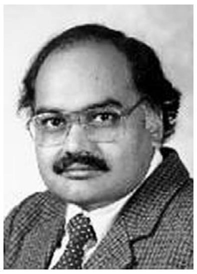
Joh H, Kim J, Malaiya YK. Vulnerability discovery modeling using Weibull distribution. In: 19th international symposium on software reliability engineering; 2008. p. 299–300.

Kargl F, Maier J, Weber M. Protecting web servers from distributed denial of service attacks. In: Proc. 10th international WWW conference; 2001. p. 514–24.
Kim J, Malaiya YK, Ray I. Vulnerability discovery in multi-version software systems. In: Proc. 10th IEEE Int. symp. on high assurance system engineering (HASE), Dallas; Nov. 2007. p. 141–8.
Landwehr CE, Bull AR, McDermott JP, Choi WS. A taxonomy of computer program security flaws. ACM Computing Surveys; 1994:211–54.
Lyu MR. Handbook of software reliability. McGraw-Hill; 1995.
Machie A, Roculan J, Russell R, Velzen MV. Nimda worm analysis. Tech. Rep.. In: Incident analysis. SecurityFocus; September 2001
Madan BB, Goseva-Popstojanova K, Vaidyanathan K, Trivedi KS. A method for modeling and quantifying the security attributes of intrusion tolerant systems. Performance Evaluation; 2004: 167–86.
Malaiya YK, Karunanithi N, Verma P. Predictability of software reliability models. IEEE Transactions on Reliability December 1992;41(4):539–46.
McGraw G. From the ground up: the DIMACS software security workshop. IEEE Security and Privacy March/April 2003;1(2): 59–66.
Moore D, Shannon C, Claffy KC. Code-red: a case study on the spread and victims of an internet worm. In: Internet measurement workshop; 2002. p. 273–84.
Musa JD, Okumoto K. A logarithmic Poisson execution time model for software reliability measurements. In: Proceedings of 7th international conference on software engineering, Silver Spring, MD; March 1984. p. 230–38.
Musa J. Software reliability engineering. McGraw-Hill; 1999.
Netcraft, http://news.netcraft.com/; April 2010.
National Vulnerability Database (NVD), http://nvd.nist.gov/; April 2010.
Oracle WebLogic Server, http://www.oracle.com/technology/products/Weblogic/index.html; April 2010.
Open Source Vulnerability Database (OSVDB), http://osvdb.org; April 2010.
Pfleeger CP, Pfleeger SL. Security in computing. 3rd ed. Prentice Hall PTR; 2003.
Rescorla E. Security holes. Who cares?. In: Proc. 12th USENIX security symposium; 2003. p. 75–90.
Rescorla E. Is finding security holes a good idea? IEEE Security and Privacy; 2005:14–9.
Sahinoglu M. Quantitative risk assessment for dependent vulnerabilities. In: Proc. reliability and maintainability symposium; Jan. 2006. p. 82–85.
Seacord CR, Householder AD. A structured approach to classifying vulnerabilities. Technical Report CMU/SEI-2005-TN-003. Carnegie Mellon; 2005.
Seacord R. Secure coding in C and C++. Addison Wisely; 2005.
Secunia, http://secunia.com/; April 2010.
Securityfocus, http://www.securityfocus.com/; April 2010.
SLOCCount, http://dwheeler.com/sloccount; April 2010.
Venter HS, Eloff JHP, Li YL. Standardizing vulnerability categories. Computers & Security May–June 2008;27(3–4):71–83. doi:10.1016/j.cose.2008.04.002. ISSN: 0167-4048.
Woo S-W, Alhazmi OH, Malaiya YK. Assessing vulnerabilities in Apache and IIS HTTP servers. In: Proc. IEEE Int. symp. on dependable, autonomic and secure computing (DASC'06); Sept.–Oct. 2006. p. 103–10.

Sung-Whan Woo is a PhD candidate in the Computer Science Department at Colorado State University. He received an M.S. in the Computer Science from Colorado State University in 2006. His primary research interests are in string searching problems, graph theory (Interval graph) and software vulnerability (web server and web browser vulnerabilities).

HyunChul Joh is a Ph.D. student in computer science department at Colorado State University. His research focuses on modeling the discovery process for security vulnerabilities and risk metrics. He received an M.S. in computer science from Colorado State University in 2007 and a B.E. in Information and Communications Engineering from Hankuk University of Foreign Studies in Korea in 2005.

Omar H. Alhazmi received the Ph.D. in Computer Science from Colorado State University in 2006, and his Master's degree from Villanova University in 2001. His work has involved collection and analysis of vulnerability data for several major applications and developing analytical models. He has published eleven papers on quantitative vulnerability discovery modeling. He has been working at the National Information Center's Research Center in Riyadh since 2007.



Yashwant K. Malaiya is a Professor in the Computer Science Department at Colorado State University. He received M.S. in Physics from Sagar University, MScTech in Electronics from BITS Pilani, and PhD in Electrical Engineering from Utah State University. He has published more than 160 papers in the areas of fault modeling, software and hardware reliability, testing and testable design, and quantitative security risk evaluation. He served as the General Chair of 1993 and 2003 IEEE International Symposium on Software Reliability Engineering (ISSRE). He co-edited the IEEECS Technology Series book *Software Reliability Models, Theoretical Developments, Evaluation and Applications*. He is a recipient of the IEEE Third Millennium Medal, and the IEEE Computer Society Golden Core award.