

Assessing Vulnerabilities in Apache and IIS HTTP Servers

Sung-Whan Woo, Omar H. Alhazmi and Yashwant K. Malaiya

Computer Science Department

Colorado State University, Fort Collins, CO 80523

woo/omar/malaiya@cs.colostate.edu

Abstract

We examine the feasibility of quantitatively characterizing the vulnerabilities in the two major HTTP servers. In particular, we investigate the applicability of quantitative empirical models to the vulnerabilities discovery process for these servers. Such models can allow us to predict the number of vulnerabilities that may potentially be present in a server but may not yet have been found. The data on vulnerabilities found in the two servers is mined and analyzed. We explore the applicability of a time-based and an effort-based vulnerability discovery model. The effort-based model requires data of the current market-share of a server. Both models have been successfully used for vulnerabilities in the major operating systems. Our results show that both vulnerabilities discovery models fit the data for the HTTP servers well. We also examine a separate classification schemes for server vulnerabilities that based on the source of error, and then explore the applicability of the quantitative methods to individual classes.

1. Introduction

There has been considerable discussion of 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 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. has made a side-by-side comparison between various general servers and the number of vulnerabilities and severity. This study concluded that there is a need to develop some tools for estimating the risks posed by vulnerabilities [1].

Two of the major software components of the Internet are an HTTP (Hyper Text Transfer Protocol) server (also termed a web server) and the browser, which serves as the client. Both of these components were first introduced in 1991 by Tim Berners-Lee of CERN. They have now become indispensable parts of both organizational and personal interactions. The early web servers provided information using static HTML pages.

The web server now provides dynamic and interactive services between the server and client using database queries, executable script, etc. The web server is able to support functions such as serving streaming media, mail, etc. An HTTP server has thus emerged as a focal point for the Internet.

We examine the vulnerabilities in the two most widely-used HTTP servers, the Apache server, introduced in 1995, and the Microsoft IIS (Internet Information Services) server, originally supplied as part of the NT operating systems in 1995-96. While Apache has a much larger overall market share, roughly 63%, 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 HTTP server that is not open-source. Both Apache and IIS are generally comparable in features. IIS runs only under the Windows operating systems, whereas Apache supports all the major operating systems.

The security of systems connected to the Internet depends on several components of the system. These include the operating systems, the HTTP servers and the browsers. Some of the major security compromises arise because of vulnerabilities in the HTTP servers. A vulnerability is defined as “a defect which enables an attacker to bypass security measures” [2]. 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 organizations such as National Vulnerabilities Database [3], MITRE [4], Bugzilla [5], BugTraq [6], etc., as well as the developers of the software. The exploitations of some of the server vulnerabilities are well known. The Code Red worm [7], 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 subject to some security risk. While there have been many studies attempting to identify causes of vulnerabilities and potential counter-measures, the development of systematic quantitative methods to characterize security has begun only recently. There has been considerable debate comparing the security attributes of open source and commercial software [8].

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 not yet discovered vulnerabilities and the vulnerabilities discovery rate rather than by the vulnerabilities that have been discovered in the past and remedied by patches. Possible approaches for a quantitative perspective of exploitation trends are discussed in [9]. Probabilistic examinations of intrusions have been presented by several researchers [10, 11]. In [12], Rescorla has studied vulnerabilities in open source servers. The vulnerabilities discovery process in operating systems has just recently been examined by Rescorla [13] and by Alhazmi and Malaiya [14, 15, 16].

Servers are very attractive targets for malicious attacks. It is essential to understand the threat posed by both undiscovered vulnerabilities and recently discovered vulnerabilities for which a patch has not been developed or applied. At this time, despite the significance of security in the HTTP servers, very little quantitative work has been done to model the vulnerabilities discovery process for the servers. Such work would permit the developers and the users to better estimate future vulnerabilities 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 problem or attacks that the servers face, such as denial of service attacks (DoS) [17, 18], in which the authors suggests some countermeasures to be applied when an attack of this type takes place. In this paper, our focus is the discovery rates of vulnerabilities of all types.

The next section introduces the two vulnerabilities discovery models used. We then consider the total number of vulnerabilities in the two HTTP servers and examine how well the models fit the available data. We then partition the vulnerabilities into categories based on how such vulnerabilities arise, and consider the applicability of the models to individual partitions. Lastly, we discuss the major observations and present the conclusions.

2. Vulnerability discovery models

Use of reliability growth models is now common in software reliability engineering [19, 20]; SRGMs show that as bugs are found and removed, fewer bugs remain. Therefore, 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.

Vulnerabilities are a special class of defects that can permit circumvention of security measures. Some

vulnerabilities discovery models were recently proposed by Anderson [8], Rescorla [13], and Alhazmi and Malaiya [14]. The applicability of these models to several operating systems was examined in [17]. The results show that while some of the models fit the data for most operating systems, others do not fit well or provide a good fit only during a specific phase.

Here, we investigate the applicability of two of the most successful models for HTTP servers. The models used are time-based and effort-based models proposed by Alhazmi and Malaiya [14]. These two models have been found to fit datasets for several of the major Windows and Linux operating systems, as determined by goodness of fit and other measures. The first model considers calendar time as the independent variable. The model incorporates the effect of the rising and declining market share on the software. The second model requires explicit estimation of the effort using an effort measure, which is then used as an independent variable.

The Alhazmi-Malaiya Time-Based Model: This model, referred to as the 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 [14]. The first factor 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 first factor. While it is possible to obtain a more complex model, this model provides a good fit to the data, as shown below. Let us assume that the vulnerabilities discovery rate is given by the differential equation:

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

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, we obtain

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

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. The model derivation assumes the software is stable. It is applicable even when the software is gradually evolving, however the model parameters will take different values.

Equation 2 shows S-shaped plot for the time-based model, which is determined by values of A , B and C . Thus, the vulnerabilities discovery rate increases at the

beginning, reaches a steady rate and then starts declining. Consequently, 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, a steady rate of vulnerabilities finding yields a linear curve. Eventually, as the vulnerabilities discovery rate begins to drop, there is saturation due both to reduced attention and a smaller pool of remaining vulnerabilities.

The Alhazmi-Malaiya Effort-Based Model: Vulnerabilities are usually reported using calendar time, because it is easy to record vulnerabilities and link them to the time of discovery. This, however, does not take into consideration 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 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 in-house and external, would go toward a system with a larger installed base.

Using effort as a factor was first discussed in [21]. However, the authors did not suggest a unit or way of measuring effort. The Effort-based Model utilizes a measure termed *Equivalent Effort (E)*, which is calculated using

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

where U_i is the total number of all HTTP servers at the period of time i , n represents the last period of usage time, and P_i is the percentage of the servers using the specific server for which we are measuring E . N_i is the number of machines running the specific server during time i . The result is given in system-months. The measure E can be calculated for the servers using the data available at [3].

The model employs equivalent effort as a factor to model vulnerabilities discovery. 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 software reliability growth models (SRGMs).

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

$$\Omega(E) = B(1 - e^{-\lambda_{vu} E}), \quad (4)$$

where λ_{vu} is a parameter analogous to failure intensity in SRGMs and B is another parameter. B represents the

number of vulnerabilities that will eventually be found. We will refer to the model given by Equation 4 as the Effort-Based Model.

3. Vulnerabilities in HTTP server

In this section, the datasets for the total vulnerabilities of the Apache and Microsoft IIS web servers are fitted to the models. The goodness of fit is evaluated to determine how well the models reflect the actual vulnerabilities discovery process. The vulnerabilities data are from the National Vulnerabilities Database maintained by NIST. The market share data from Netcraft [23] was used. We note that Apache represents an open source software and IIS represents a closed source, i.e., a commercial system. It should also be noted that the number of vulnerabilities, either found or estimated as remaining, should not be the only measurement of a security threat. Factors such as patch development and application delays and vulnerabilities' exploitation rates also need to be considered. 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 with a higher market share.

Table 1 presents data obtained from NVD and Netcraft, showing the current web server market share and total number of vulnerabilities found to date. For servers with a lower percentage of the market, such as Sun Java System Web Server (SJSWS) and Zeus, the total number of vulnerabilities found is low. That does not mean that these systems are more secure, but merely that only limited effort has gone into detecting their vulnerabilities. A significant number of vulnerabilities have been found in both Apache and IIS, illustrating the impact of the market share on the motivation for exploring or finding vulnerabilities. In this study, we use market share as an indicator of effort for the effort-based model.

Table 1. Market share and vulnerabilities found

Web Server	Apache	IIS	SJSWS	Zeus	Other
Market Share	63.09%	29.48%	0.39%	0.59%	6.45%
Vulnerabilities	96	123	3	5	N/A
Release Year	1995	1995	2002	1995	N/A
Latest Version	2.2	6.0	6.1	4.3	N/A

Figure 1 shows the web server market share for Apache and IIS. As demonstrated by Figure 1, the number of web servers continues to grow steadily. Among the various web servers, Apache and Microsoft IIS dominate the web server market. Since the total share of all of SJSWS and Zeus added together represents less

than 10% of the market share, very few vulnerabilities have been found in them and hence the data for these servers has not been used in our study.

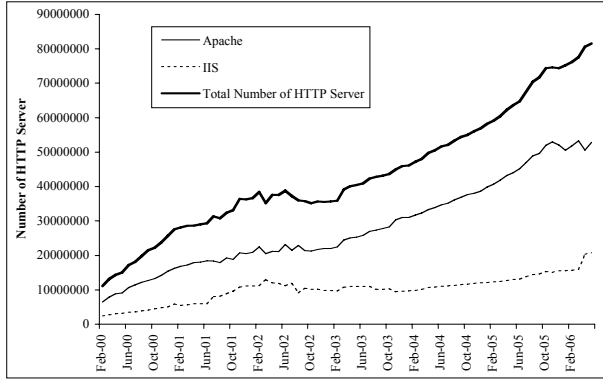


Figure 1. Server market share trends

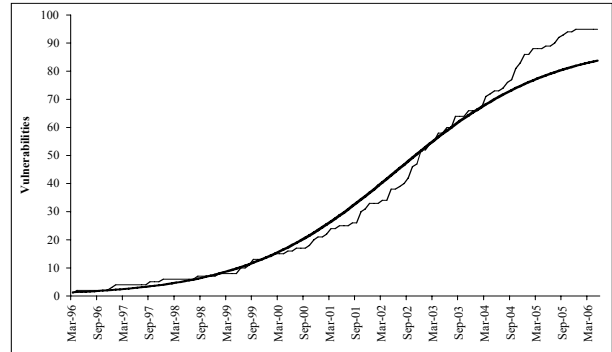
There is a marked gap between the Apache and IIS market shares, as shown in Figure 1. This difference in market share may be due to several factors. Perhaps the most important of these is that Apache is available for all major operating system platforms and can be obtained without cost. Apache may also have benefited from not having been exposed to serious security issues such as the Code Red [7] or Nimda worms that were faced by IIS in 2001.

The Apache HTTP server was first released in middle of 1995. Since then it has gained wide popularity and is used by over 50 million web server systems. In this section, we fit the vulnerabilities data for Apache to the time-based and the effort-based models. Figure 3 gives the vulnerabilities data from NVD for the period between March 1996 and May 2006, and the Netcraft market share data covers the same time period.

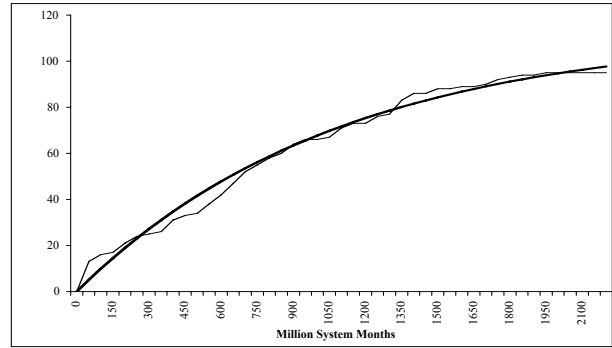
In Figure 2 and 3, the bold black lines indicate the fitted models, while the other lines show cumulative vulnerabilities for Apache. Figure 2 (a) shows cumulative vulnerabilities by month for the time-based model. At the beginning, the slope of the curve for Apache rises gently until about January 2000, after which the slope has remained steady. From the point of the three phases of the vulnerabilities discovery process [14], Apache has not yet entered the saturation phase. Apache currently appears to be in the linear phase, since the number of vulnerabilities still appears to be growing linearly. Despite having been on the market for several years, Apache has not reached the saturation phase possibly because of its larger market share; moreover, the number of systems using the Apache is still increasing. This means that vulnerabilities discovery for Apache can be expected to continue at a significant pace in near future

Figure 2 (b) shows cumulative vulnerabilities by number of Apache installations in terms of million system-months and the fitted effort-based model. This

effort-based model shows that Apache has not yet approached the saturation phase since the number of vulnerabilities continues to increase approximately linearly as the number of Apache servers increases.



(a) Time-based model



(b) Effort-based model

Figure 2. Fitting Apache vulnerabilities data

IIS was released in the early part of 1996. IIS is a popular commercial web server with about 15 million installations currently. We have used the vulnerabilities data from January 1997 to May 2006.

Figure 3 (a) shows the cumulative 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. In recent months, the vulnerabilities discovery rate for IIS has dropped to a very low point. A possible explanation for this can be that the number of IIS web servers installed appears to be stationary, unlike the Apache server which is still gaining in terms of new installations. Another possibility is that the number of remaining undiscovered vulnerabilities may actually have dropped significantly.

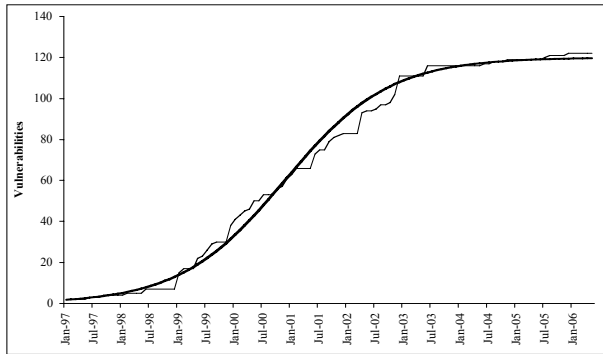
Figure 3 (b) shows cumulative vulnerabilities for the IIS server and the effort-based model by million system-months. Unlike Figure 2 (b), Figure 3 (b) shows a significant degree of saturation.

We examine the fit of the models to the data as shown in Figures 2 and 3. For χ^2 goodness of fit test, we chose an alpha level of 5%. Table 2 gives the chi-square values and parameter values for both the time-based and effort-

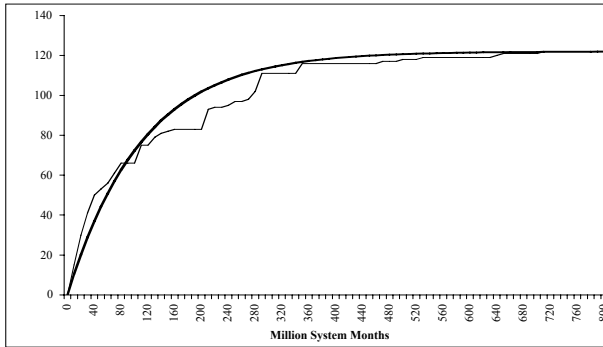
Table 2. χ^2 Goodness of fit test results for total number of vulnerabilities

	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	.00062	90.01	0.7675	64.24	148.78	0.999	112.5	.00092	23.726	61.66	.992
IIS	.00075	120	0.5959	35.54	138.81	1	122	.009	46.6	103	.998
Win 98	.0048	37.73	0.554	7.365	60.481	1	37	.0005	3.510	44.9853	1
Win NT4	.0006	136	0.522	35.58	103.01	1	108	.0030	15.05	42.5569	0.985

based models. For comparison, we also provide corresponding parameter values for the Windows 98 and NT operating systems, as well as the chi-square values.



(a) Time-based model



(b) Effort-based model

Figure 3. Fitting IIS vulnerabilities data

Table 2 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. Both data sets fit both models with χ^2 P-values ranging from 0.959 to nearly 1, indicating that the fit is quite significant. We can also note that parameter A is always less than 0.005 and parameter C is always less than 0.85, while parameter B corresponds approximately to the number of vulnerabilities.

4. Vulnerability categories

In the previous section we examined the application of the time-based and the effort-based model for the total number of vulnerabilities of Apache and IIS. In this, we apply these models to a classification schemes for server vulnerabilities.

Distinguishing among vulnerabilities is useful when we want to examine the nature and extent of the problem. It can help determine what protective actions would be most effective. Vulnerabilities taxonomy is still an evolving area of research. Several taxonomies have been proposed [24, 25, 26, 27, 28]. 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, we use the classification scheme employed by the National Vulnerability Database of the National Institute of Standards and Technology. This classification is based on the causes of vulnerabilities. The eight classes are as follows [3, 6]:

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%).

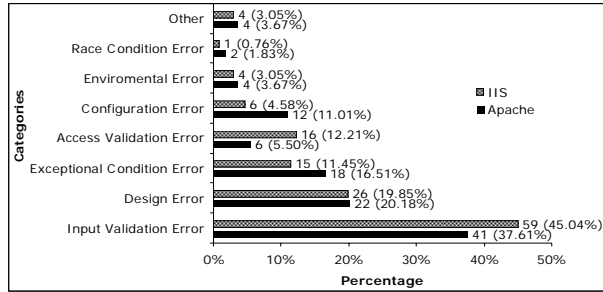
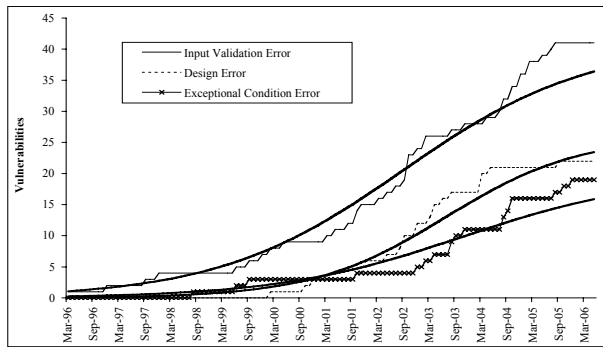
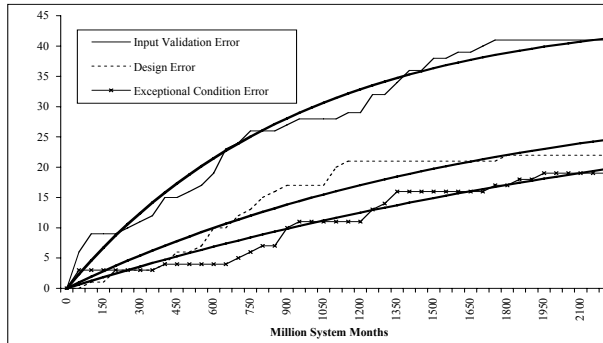


Figure 4. Vulnerabilities by category

Figure 4 compares vulnerabilities distributions in Apache and IIS. The categories with the highest proportions 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.



(a) Time-based model



(b) Effort-based model

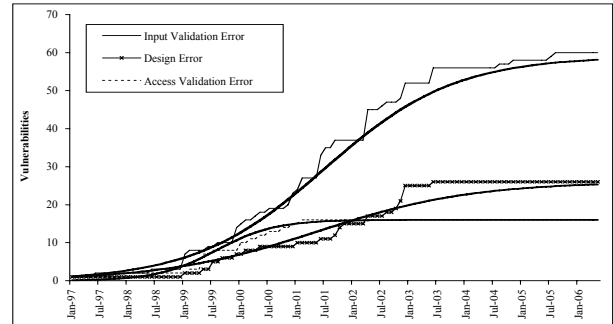
Figure 5. Fitting Apache by category

We plot the vulnerabilities for the major categories to determine whether there is an observable pattern at the level of individual classes. Since we noted a similar pattern for the uncategorized vulnerabilities, a possible fit was examined. Figures 5 and 6 show the fit for the Apache and Microsoft IIS, respectively.

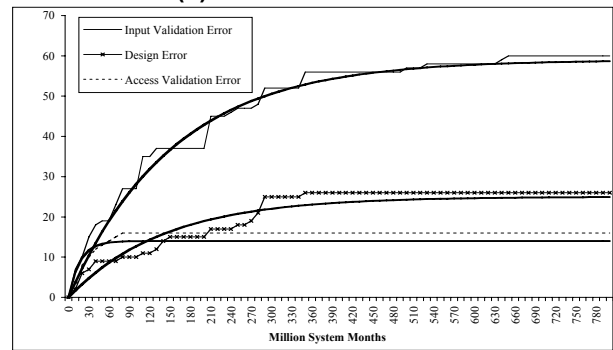
In Figure 5, we only consider the three major categories, examining only: input validation errors, design errors and exceptional handling condition errors.

As we mentioned above, the IIS model has a better fit than the Apache model, since IIS has reached the saturation phase. 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 total number of vulnerabilities. Our time-based and effort-based models are fitted for each category. It may be noted that the number of input validation errors and design errors are the most common category in Apache and IIS.

Table 4 shows the chi-square goodness of fit tests for the Apache and IIS models by category. Table 3 demonstrates that the chi-square value for each category are less the P-values are close to 1, the fit of input validation, design and exceptional condition error classes are significant for both models.



(a) Time-based model



(b) Effort-based model

Figure 6. Fitting IIS by category

5. 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 is likely to dominate.

Table 3. Apache and IIS's category chi-square analysis of goodness of fit

		Time-Based Model (Apache)						Effort-Based Model (Apache)				
		A	B	C	χ^2	$\chi^2_{critical}$	P-Value	B	λ_{VU}	χ^2	$\chi^2_{critical}$	P-value
Apache	Input Validation	.00113	41.06	.902	49.96	148.78	1	45.67	.00105	13.426	61.66	.999
	Design Error	.00248	25.998	11.05	45.08	148.78	1	34	.00058	14.77	61.66	.999
	Exceptional Condition	.00238	19.686	3.704	54.7	148.78	0.999	37.6	.00034	19.30	61.66	.997
IIS	Input Validation	.00122	59	.8899	27.12	138.81	1	59	.0065	13.71	103	1
	Design Error	.00234	26	1	41.14	138.81	1	25	.0071	21.14	103	1
	Access Validation	.01	16	10	33.17	138.81	1	14	.0604	24.66	103	1

The results of model fitting for the vulnerabilities classified by type are shown in Table 3. The fitting was done for the most common types of vulnerabilities for which the available data is statistically significant. It would be difficult to use these models to estimate the types of vulnerabilities that occur less frequently because the data may not be sufficiently statistically significant to make meaningful projections.

The effort-based model requires the use of the market share data, 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. Further research needs to be done to evaluate the predictive capabilities of the two models.

Even though the Apache's vulnerability discovery rate has not reached saturation yet as observed in this paper, we had applied the time-based model for Apache and the statistical results show that the fit is significant.

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}), which can then be used to estimate the total number of vulnerabilities of a comparable system. D_{KD} gives the defects per thousand lines of code and V_{KD} is the number of vulnerabilities per thousand lines of code. Table 4 shows some of 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 SLOC numbers, since both offer the same features. The Apache source code size for Windows is 328 Ksloc, larger than for Unix. A few of the Apache vulnerabilities may be applicable to only a specific platform. In Table 4, we observe that 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 has 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 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.

Table 4. Known D_{KD} vs. known V_{KD}

App-lication	Ksloc	Known Defects	D_{KD}	Known Vulnerabilities	V_{KD}	Ratio V_{KD}/D_{KD}
Apache	227(Unix)	4148	18.27	96	0.423	0.0232
IIS	N/A	N/A	N/A	123	N/A	N/A
Win 98	16,000	10,000	0.625	91	0.0057	0.0091
Win NT	18,000	10,000	0.556	230	0.0128	0.023

After comparing the vulnerabilities trends of the web servers discussed in this paper, it is expected that fewer vulnerabilities will be discovered in IIS in the future. This may lead to the conclusion that IIS is more secure than Apache in this respect. However, this is simply due to the fact that IIS has reached saturation phase, even though more IIS vulnerabilities have been found in the past. Other factors such as patch release, number of remaining vulnerabilities, economic aspects etc., also need to be considered when choosing a web server.

6. Conclusions

This study has demonstrated that the vulnerabilities discovery process in servers follows a pattern, which can be modeled. It is therefore possible to make reasonable projections about the number of remaining vulnerabilities and vulnerabilities discovery rates.

We also examined the application of the models to vulnerabilities belonging to specific categories. The fit was significant for both the time-based and the effort-based models. The distribution of the vulnerabilities into specific categories was also analyzed and compared with distributions in the operating systems. The results show that the distributions are comparable for the distributions of the operating systems. It was observed that a larger number of input validation error vulnerabilities constitute a high risk. This suggests that more effort should be spent on testing in order to target vulnerabilities from this class, thereby minimizing the number of high risk vulnerabilities.

The results indicate that the models originally proposed for operating systems are also applicable to servers. These models can be used to estimate vulnerabilities discovery rates, which can be integrated with risk assessment models

in the future. A model recently proposed by Sahinoglu [29] needs such an assessment for estimating risk and cost of loss. Furthermore, these models can be integrated into the development process to create more secure software systems [30].

Further work is needed to evaluate the prediction accuracy of the models so that the users can measure how accurately these models can predict future vulnerabilities discovery rates [31]. Further research is also needed to evaluate the degree of confidence that can be attained when these methods are used to predict the type of vulnerabilities that are anticipated and their severity levels.

References

- [1] R. Ford, H. Thompson and F. Casteran, *Role comparison report—web server role*. Technical Report, Security Innovation, 2005.
- [2] E.E. Schultz, D.S. Brown and L. T. Longstaff, *Responding to computer security incidents*, Technical report, Lawrence Livermore National Laboratory, July 1990.
- [3] National Vulnerability Database. <http://nvd.nist.gov/>, April 2006.
- [4] Mitre Corp, Common Vulnerabilities and Exposures, <http://www.cve.mitre.org/>, April 2006.
- [5] Apache Software Foundation Bug System, <http://issues.apache.org/bugzilla/>, April 2006.
- [6] Securityfocus, <http://www.securityfocus.com/>, April 2006.
- [7] D. Moore, C. Shannon and K.C Claffy, “Code-red: a case study on the spread and victims of an internet worm”, *In Internet Measurement Workshop*, 2002, pp. 273–284.
- [8] R. Anderson, “Security in open versus closed systems—the dance of boltzmann, coase and moore”. *In Conf. on Open Source Software: Economics, Law and Policy*, 2002, pp. 1–15.
- [9] J. Hallberg, A. Hanstad, and M. Peterson, “A framework for system security assessment”, *Proc. 2001 IEEE Symposium on Security and Privacy*, May 2001, pp. 214–229.
- [10] H.K. Browne, W.A. Arbaugh, J. McHugh and W. L. Fithen, “A trend analysis of exploitations”, *In IEEE Symposium on Security and Privacy*, 2001, pp. 214–229.
- [11] B.B. Madan, K. Goseva-Popstojanova, K. Vaidyanathan, and K.S. Trivedi, “A method for modeling and quantifying the security attributes of intrusion tolerant systems”, *Perform. Eval.*, 2004, pp. 167–186.
- [12] E. Rescorla, “Is finding security holes a good idea?”, *IEEE Security and Privacy*, 2005, pp. 14–19.
- [13] E. Rescorla, “Security holes... who cares?”, *Proc. 12th USENIX Security Symposium*, 2003. Pp. 75-90.
- [14] O.H. Alhazmi, and Y.K. Malaiya, “Quantitative vulnerability assessment of system software”, *Proc. Annual Reliability and Maintainability Symposium*, Jan. 2005, pp. 615–620.
- [15] O.H. Alhazmi, and Y.K. Malaiya and I. Ray, “Security vulnerabilities in software systems: A quantitative perspective. *Proc. Ann. IFIP WG11.3 Working Conference on Data and Information Security*, Aug. 2005, pp. 281–294.
- [16] O.H. Alhazmi, and Y.K. Malaiya, “Modeling the vulnerability discovery process”, *Proc. 16th International Symposium on Software Reliability Engineering*, Nov. 2005, pp. 129–138.
- [17] T. Aura, M. Bishop and D. Sniegowski, "Analyzing Single-Server Network Inhibition", *Proceedings of the 13th IEEE Computer Security Foundations Workshop*, July 2000, pp.108-117.
- [18] F. Kargl, J. Maier and M. Weber, “Protecting web servers from distributed denial of service attacks”, *Proc. 10th International WWW Conference*, 2001, pp. 514–524.
- [19] J. Musa, *Software Reliability Engineering*. McGraw-Hill, 1999.
- [20] Lyu, M.R., *Handbook of Software Reliability*. McGraw-Hill, 1995.
- [21] O. H. Alhazmi, and Y. K. Malaiya, “Prediction capability of vulnerability discovery process”, *Proc. Reliability and Maintainability Symposium*, Jan. 2006. pp. 86-91
- [22] B. Littlewood, S. Brocklehurst, N.E. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, J. McDermid, and D. Gollmann, “Towards operational measures of computer security” *Journal of Computer Security*, 1993, pp. 211–230.
- [23] Netcraft, <http://news.netcraft.com/>, April 2006.
- [24] T. Aslam and E.H. Spafford, *A taxonomy of security faults*, Technical report, Carnegie Mellon, 1996.
- [25] M. Bishop, “Vulnerability analysis: An extended abstract”, *Proc. Second International Symposium on Recent Advances in Intrusion Detection*, Sept. 1999, pp. 125-136.
- [26] C.E. Landwehr, A.R. Bull, J.P. McDermott and W. S Choi, “A taxonomy of computer program security flaws”, *ACM Comput. Surv.*, 1994, pp. 211–254.
- [27] C.R. Seacord and A.D. Householder, *A structured approach to classifying vulnerabilities*, Technical Report CMU/SEI-2005-TN-003, Carnegie Mellon, 2005.
- [28] R. Gopalakrishna, E.H. spafford, and J. Vitek, “Vulnerability Likelihood: A Probabilistic Approach to Software Assurance,” Technical Report, CERIAS, 2005.
- [29] M. Sahinoglu, “Quantitative risk assessment for dependent vulnerabilities”, *Proc. Reliability and Maintainability Symposium*, Jan. 2006, pp. 82-85.
- [30] Seacord, R., *Secure Coding in C and C++*. Addison Wisely, 2005.
- [31] O.H. Alhazmi and Y.K. Malaiya, "Measuring and Enhancing Prediction Capabilities of Vulnerabilities Discovery Models for Apache and IIS HTTP Servers ", to be presented at *International Symposium on Software Reliability Engineering.*, Nov. 2006.