Seasonal Variation in the Vulnerability Discovery Process

HyunChul Joh and Yashwant K. Malaiya
Computer Science Department
Colorado State University, Fort Collins, CO 80523
[*dean2026, malaiya*]*@cs.colostate.edu*

Abstract

Vulnerability discovery rates need to be taken into account for evaluating security risks. Accurate projection of these rates is required to estimate the effort needed to develop patches for handling vulnerabilities discovered. Seasonal behaviors of the vulnerability discovery process for a multi-year life-cycle of software products are examined. A careful inspection of the data for several major operating systems, web servers and web browsers suggests presence of a seasonal behavior that is not considered by the vulnerability discovery models. This paper examines the statistical significance of the annual seasonal pattern in the vulnerability discovery rates using the seasonal index approach. The autocorrelation function is used to identify the periodicity. A time series analysis that combines the longer term trends with cycles caused by seasonality may predict the future pattern more accurately. The analysis of the datasets for eight major operating systems and four web related software systems (Windows NT, XP, 2000, Server 2003, MAC OS X, HP-UX, Solaris, Red Hat Linux, IIS, Apache, Internet Explorer and Firefox) shows that there is indeed an annual seasonal pattern. While all the programs exhibit a year-end peak, a higher incidence is also observed during the mid-year months for Microsoft products.

1. Introduction

A security vulnerability in a major application that has been discovered and disclosed, but not remedied represents a great risk to both organizations and individuals using that software. They impact the security of several classes of software systems. Operating systems form the complex foundation for computing systems. Also, web servers and browsers are significantly important when connectivity of computing systems has been essential thanks to the Internet. Unfortunately, a large number of vulnerabilities are discovered in OSes and web related software systems

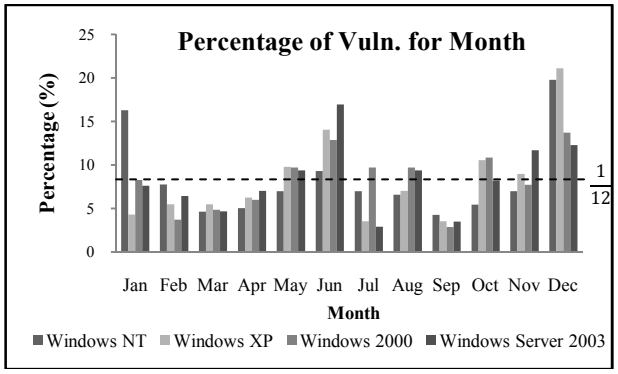

Figure 1. Vulnerability discovery process (%)

every year which represent a major security risk [1]. When we can predict the vulnerability discovery pattern expected and the attributes of the vulnerabilities discovered, we can allocate the needed resource at the right time for corrective measures, which can greatly reduce the security risks. Moreover, it can be used by end-users to assess risks and estimate potential security breaches.

Seasonal effect is well established research area in other disciplines such as the stock market [2], high-performance computing systems [3], epidemiology [4], power transmission [5], marine biology [6], and birth defects [7], etc. For example, stocks tend to have relatively higher returns for some specific calendar months. The higher return during November to April is termed the Halloween Effect [8]. The authors of [2] have identified the repetitive pattern in [9] with lags of 12, 24, and 36 months. As a result, seasonal analysis is a well known statistical approach in the repertoire of many researchers and analysts in those disciplines.

Here we examine the potential seasonal effect in the software vulnerability discovery process, which is suggested by the data for several software systems, for example the four Windows OSes considered in Figure 1. In the figure, the percentages of vulnerabilities found during each cumulative calendar month for the lifetime are shown: Windows NT (1995-2007; 258 vuln.), Windows XP (2001-2007; 256 vuln.), Windows 2000 (1997 – 2007 ; 350 vuln.) and Windows server 2003

978-0-7695-3601-9/09 $25.00 © 2009 IEEE
DOI 10.1109/ICST.2009.9

(2002-2007; 171 vuln.). We note that for Windows NT, 19.77% of the vulnerabilities were found in December, far in excess of 100/12 = 8.33% which would be expected if the vulnerability discovery rate were uniform. For all four operating systems, the fraction of vulnerabilities detected in June and December is significantly higher. We also note that the fraction of vulnerabilities detected in February, March, April and September is lower than average. This would suggest a possible seasonal pattern that may be taken into account to make more accurate predictions about the number of vulnerabilities expected to be discovered in a future period.

In the paper, we analyze the vulnerability datasets for major operating systems including the four Windows OSes (Windows NT, Windows XP, Windows 2000 and Windows server 2003), and four non-Windows OSes (SUN Solaris, Red Hat Linux, HP-UX and MAC OS X). They are all cumulative of total version. For comparison, we also examine the data for four major web related software systems (IIS web server, Apache web server, Internet Explorer, and Firefox web browser) to identify possible seasonal patterns. We have used two separate approaches since each method gives us some different information. First, a seasonal index is measured with chi-square test giving specific indexes for each month. Then autocorrelation function (ACF) analysis is used which gives information of correlated month.

We try to answer the question of existence of seasonal pattern and its significance based on the available vulnerability datasets in quantitative manner; we extend our previous short work [10] by more generalizing the datasets and discussing the phenomena of the results more in depth. We have grouped them into related families; Windows OSes, non-Windows OSes and web related software systems. Section 2 mentions about backgrounds. Section 3 presents the analysis of the data-sets along with related plots. The observations are discussed in the next section.

2. Background

Vulnerability can be defined as software defect or weakness in the security system which might be exploited by a malicious user causing loss or harm [11]. So far, only limited work has been done to characterize security vulnerabilities in a quantitative manner. Some vulnerability discovery models (VDMs) have been proposed recently that model the vulnerability discovery process. These include the Anderson thermodynamic model, Alhazmi-Malaiya Logistic (AML) model and the Rescorla linear/exponential models [12]. Some classical software reliability growth models have also been used as VDMs [13]. These VDMs can be used to formulate probabilistic methods to estimate characteristics of the vulnerability discovery process for specific systems.

Ozment [14] has proposed a standard set of terms relevant to measuring characteristics of vulnerabilities and their discovery process. Omar and Malaiya have compared several VDMs by fitting the data for major operating systems data, and have shown that the AML models fits better than other models in most cases [15]. However since AML is symmetrical model, it might be not perform well for asymmetric behavior discovery rate, so in [16], the authors examined Weibull distribution which can model asymmetrical behaviors too. Woo et al. [17, 18] have examined the vulnerability discovery trends for sets of web browsers and HTTP servers using AML model. Kim et al. [19] have considered the impact of shared code across successive versions and have proposed an enhanced version of the AML model by taking into account the superposition effects for multiple version software systems. In [20], using software security incident datasets, the authors compared the forecasts from time series models and from software reliability growth models.

The vulnerabilities found are generally disclosed by the finders using some of the common reporting mechanisms that have been developed. Some of them use a scoring system such as Common Vulnerability Scoring System (CVSS) [21]. The databases for the vulnerabilities and defects are maintained by organizations such as National Vulnerability Database (NVD) [22], Open Source Vulnerability Database [23], US-CERT [24], Secunia [25], etc. We have used the NVD database because it provides the most extensive datasets. NVD is the U.S. government repository of vulnerability management data collected and organized using specific standards. It includes databases of security checklists, security related software flaws, misconfigurations, product names, and impact metrics. NVD is synchronized with Common Vulnerabilities and Exposures (CVE) [26], which is a list of information security vulnerabilities and exposures that aims to provide common names for publicly known problems, so that any updates to CVE appear immediately on NVD. Since detected and qualified vulnerabilities take some time become an official CVE entry, the database does not reflect all the vulnerabilities.

The data about vulnerabilities are also provided as XML files for each year so that researchers can extract the specific information they need. In the paper, we have used those XML files (2002~2007). Year 2002 XML file includes vulnerabilities prior to and including 2002. Year 2008 is excluded for the paper since it does not reflect the entire year at the moment. Other XML files have entire records for each corresponding year if available. The NVD XML Schema File can be also found at the NVD website.

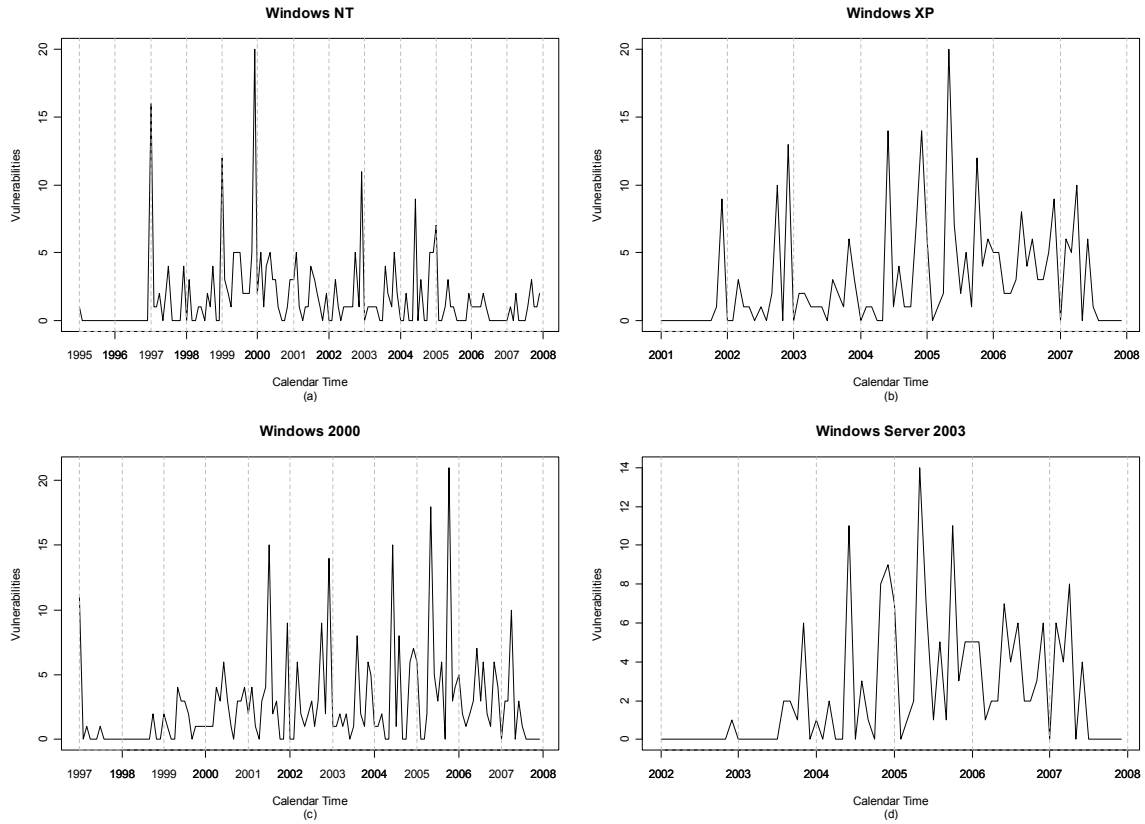

Figure 2. Vulnerabilities discovered for each month (Windows OSes)

3. Vulnerability Data Analysis

Here we present the analysis for the mentioned software systems grouped into three categories for convenience. We will examine the null hypothesis *H_0: no seasonality is present*. We will evaluate it using the seasonal index measure which states how much the average for a particular period tends to be above (or below) the expected value. The monthly seasonal index values are given by [27]:

$$s_i = \frac{d_i}{d} \tag{1}$$

where, s_i is the seasonal index for i^{th} month, d_i is the mean value of i^{th} month, d is a grand average. Hence, for example, a seasonal index of 1.25 indicates that the expected value for that month is 25% greater than 1/12 of the overall average where the expected value is 1.

To see whether the seasonal indexes are statistically significant, chi-square (χ^2) test for the null hypothesis H_0 has been done. To be statistically significant, the calculated value of χ^2 statistic value (χ_s^2) must be greater than χ^2 critical value (χ_c^2) with small enough p-value. The other approach to characterize seasonality is to use the autocorrelation function (ACF). ACF analysis gives us specific relationship information between related months. With time series values of $z_b, z_{b+1}, \ldots, z_n$, the ACF at time lag k, denoted by r_k , is [28]:

$$r_k = \frac{\sum_{t=b}^{n-k}(z_t-\bar{z})(z_{t+k}-\bar{z})}{\sum_{t=b}^{n}(z_t-\bar{z})^2} \tag{2}$$

where $\bar{z} = \frac{\sum_{t=b}^{n} z_t}{(n-b+1)}$

Table 1: (Windows OSes)
Vulnerability Discovery Seasonal Indexes

	Win NT	Win XP	Win 2000	Win 2003
Jan	1.9535	0.5156	0.9943	0.9123
Feb	0.9302	0.6563	0.4457	0.7719
Mar	0.5581	0.6563	0.5829	0.5614
Apr	0.6047	0.7500	0.7200	0.8421
May	0.8372	1.1719	1.1657	1.1228
Jun	1.1163	1.6875	1.5429	2.0351
Jul	0.8372	0.4219	1.1657	0.3509
Aug	0.7907	0.8438	1.1657	1.1228
Sep	0.5116	0.4219	0.3429	0.4211
Oct	0.6512	1.2656	1.3029	0.9825
Nov	0.8372	1.0781	0.9257	1.4035
Dec	2.3721	2.5313	1.6457	1.4737
χ_c^2	19.68	19.68	19.68	19.68
χ_s^2	78.3720	88.5312	54.9142	35.9473
p-value	3.04E-12	3.23E-14	8.04E-08	1.73E-04

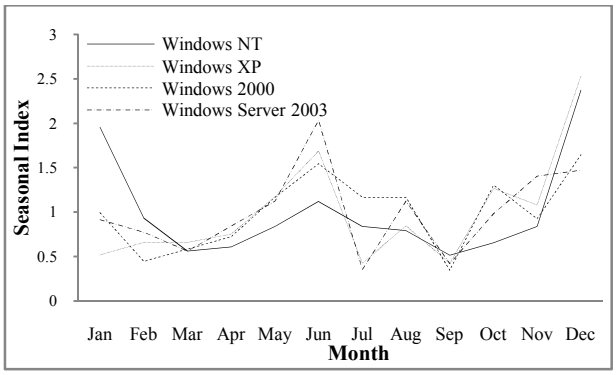

Figure 3. Seasonal Indexes (Windows OSes)

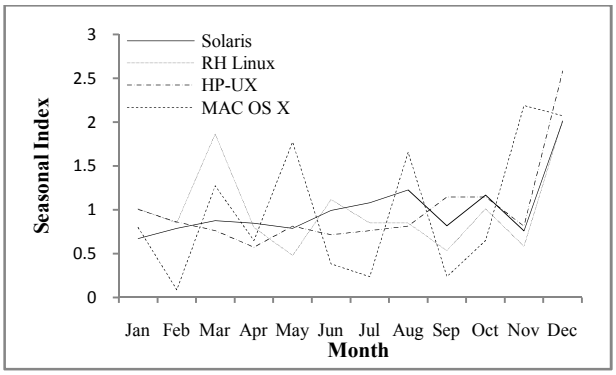

Figure 5. Seasonal Indexes (Non-Windows OSes)

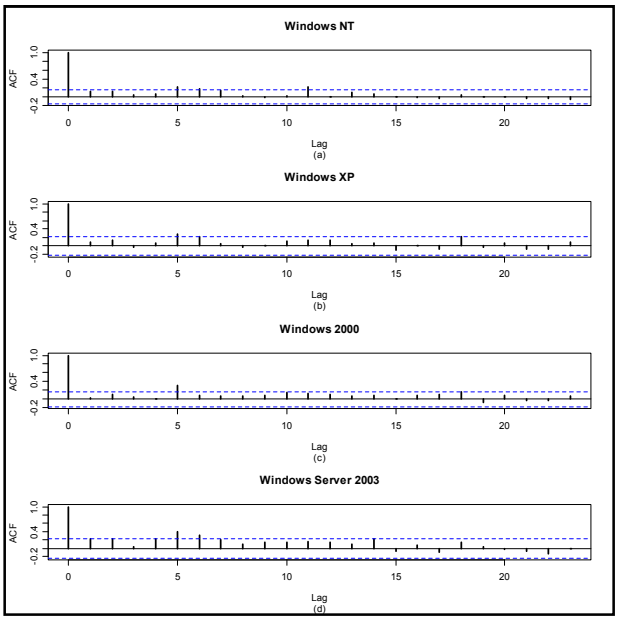

Figure 4. Autocorrelation functions (Windows OSes)

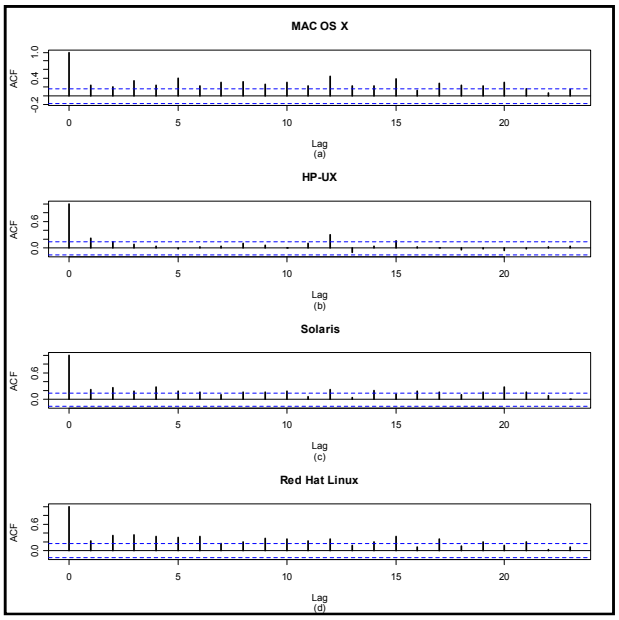

Figure 6. Autocorrelation functions (Non-Windows OSes)

Table 2. Individual ACF Values for Figure 4

Windows NT ; 95% confidence interval = (-0.1569, 0.1569)							
1^{0}	0.116^{1}	0.119^{2}	0.04^{3}	0.049^{4}	**0.212^{5}**	**0.168^{6}**	0.131^{7}
0.004^{8}	-0.018^{9}	0.012^{10}	**0.21^{11}**	-0.005^{12}	0.094^{13}	0.052^{14}	-0.01^{15}
-0.022^{16}	-0.049^{17}	0.032^{18}	-0.013^{19}	-0.01^{20}	-0.041^{21}	-0.055^{22}	-0.067^{23}
Windows XP ; 95% confidence interval = (-0.2138, 0.2138)							
1^{0}	0.089^{1}	0.125^{2}	-0.028^{3}	0.078^{4}	**0.269^{5}**	**0.217^{6}**	0.049^{7}
-0.043^{8}	0.017^{9}	0.103^{10}	0.128^{11}	0.134^{12}	0.058^{13}	0.061^{14}	-0.101^{15}
0.004^{16}	-0.086^{17}	**0.226^{18}**	-0.041^{19}	0.066^{20}	-0.072^{21}	-0.077^{22}	0.082^{23}
Windows 2000 ; 95% confidence interval = (-0.1705, 0.1705)							
1^{0}	0.028^{1}	0.11^{2}	0.04^{3}	-0.005^{4}	**0.295^{5}**	0.074^{6}	0.057^{7}
0.055^{8}	0.081^{9}	0.141^{10}	0.13^{11}	0.096^{12}	0.061^{13}	0.076^{14}	0^{15}
0.074^{16}	0.094^{17}	0.168^{18}	-0.08^{19}	0.081^{20}	-0.041^{21}	-0.033^{22}	0.064^{23}
Windows Server 2003 ; 95% confidence interval = (-0.2309, 0.2309)							
1^{0}	0.221^{1}	**0.234^{2}**	0.037^{3}	0.227^{4}	**0.387^{5}**	**0.308^{6}**	0.203^{7}
0.09^{8}	0.149^{9}	0.151^{10}	0.159^{11}	0.134^{12}	0.094^{13}	0.218^{14}	-0.06^{15}
0.071^{16}	-0.085^{17}	0.152^{18}	0.045^{19}	-0.028^{20}	-0.068^{21}	-0.127^{22}	-0.004^{23}

Table 3. Individual ACF Values for Figure 6

MAC OS X ; 95% confidence interval = (-0.1705, 0.1705)							
1^{0}	**0.252^{1}**	**0.21^{2}**	**0.352^{3}**	**0.237^{4}**	**0.399^{5}**	**0.233^{6}**	**0.306^{7}**
0.319^{8}	**0.261^{9}**	**0.299^{10}**	**0.228^{11}**	**0.452^{12}**	**0.23^{13}**	**0.218^{14}**	**0.387^{15}**
0.129^{16}	**0.292^{17}**	**0.254^{18}**	**0.215^{19}**	**0.308^{20}**	0.163^{21}	0.064^{22}	0.154^{23}
HP-UX ; 95% confidence interval = (-0.1460, 0.1460)							
1^{0}	**0.218^{1}**	0.133^{2}	0.088^{3}	0.045^{4}	-0.022^{5}	0.017^{6}	0.047^{7}
0.099^{8}	0.07^{9}	0.002^{10}	0.099^{11}	**0.299^{12}**	-0.098^{13}	0.051^{14}	**0.158^{15}**
0.014^{16}	0.013^{17}	-0.037^{18}	-0.015^{19}	-0.048^{20}	-0.024^{21}	0.028^{22}	0.044^{23}
Solaris ; 95% confidence interval = (-0.1460, 0.1460)							
1^{0}	**0.228^{1}**	**0.266^{2}**	**0.188^{3}**	**0.278^{4}**	**0.187^{5}**	**0.165^{6}**	0.107^{7}
0.165^{8}	**0.163^{9}**	**0.186^{10}**	0.064^{11}	**0.22^{12}**	0.043^{13}	**0.209^{14}**	0.122^{15}
0.172^{16}	**0.164^{17}**	0.094^{18}	**0.161^{19}**	**0.29^{20}**	**0.165^{21}**	0.076^{22}	0.01^{23}
Red Hat Linux ; 95% confidence interval = (-0.1512, 0.1512)							
1^{0}	**0.227^{1}**	**0.337^{2}**	**0.36^{3}**	**0.319^{4}**	**0.306^{5}**	**0.325^{6}**	0.148^{7}
0.19^{18}	**0.27^{9}**	**0.25^{10}**	**0.213^{11}**	**0.252^{12}**	0.116^{13}	**0.188^{14}**	**0.318^{15}**
0.084^{16}	**0.258^{17}**	0.105^{18}	**0.189^{19}**	0.11^{20}	**0.203^{21}**	0.026^{22}	0.071^{23}

***Bold** fonts indicate outside of chosen upper or lower confidence intervals and ^{Superscripts} represent time lags*

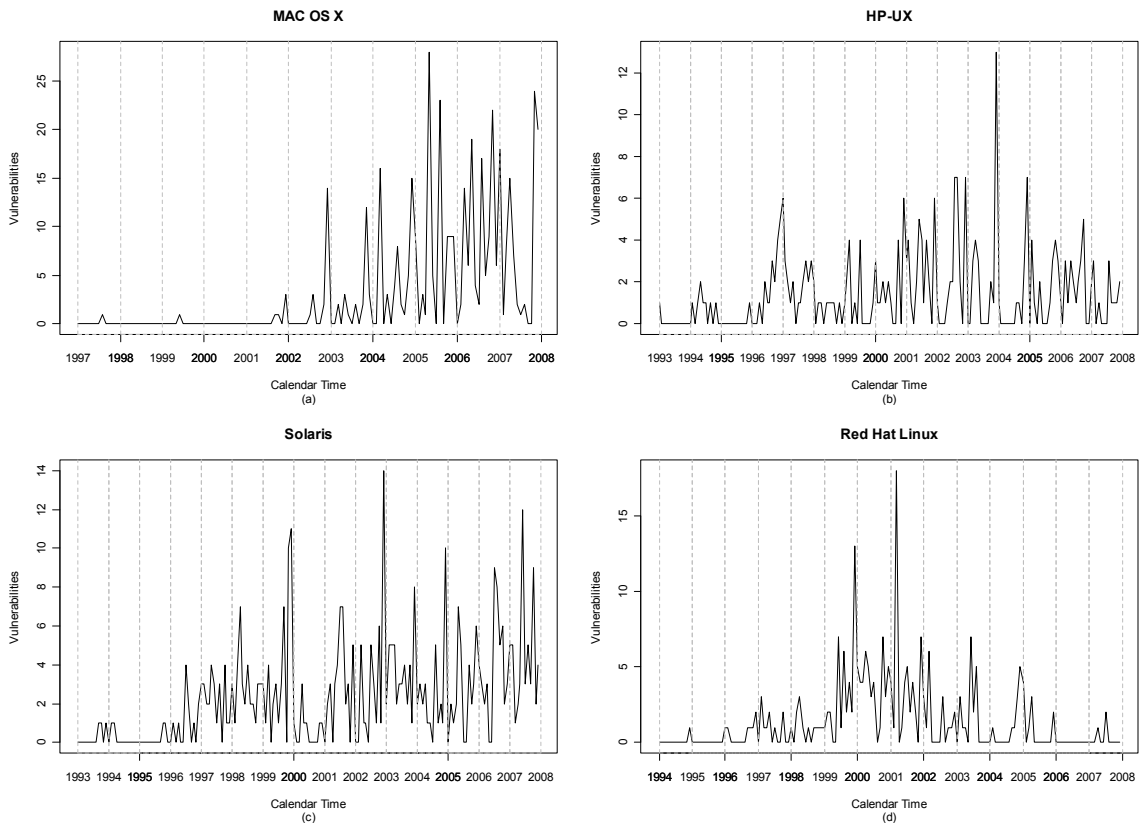

Figure 7. Vulnerabilities discovered for each month (Non-Windows OSes)

ACF measures the linear relationship between time series observations separated by a lag of k time units. When an ACF value is located outside of chosen upper or lower confidence intervals, there is a significant relationship associated with that time lag. An event occurring at time $t + k$ ($k > 0$) is said to lag behind an event occurring at time t, the extent of the lag being k. Results of the analysis for the three groups of chosen software systems using seasonal index and the ACF analysis are discussed below.

3.1 Windows Operating Systems

Here, we examine the Windows OSes. Figure 2 shows the number of vulnerabilities found along with the calendar time. Total cumulative numbers of vulnerabilities are 258, 256, 350, 171 for (a), (b), (c) and (d) respectively. The mid-year (summer) and the year-end (winter) months appear to have most of the peaks suggesting the possibility of seasonality in the discovery process for the operating systems. We examine the significance of this observed seasonality.

Seasonal index & Chi-square test. A time series data is not uniformly distributed and seasonal patterns are present in a dataset when certain months have significantly more incidences of vulnerabilities reported than other months. Table 1 shows seasonal indexes for each month of the Windows OSes. A seasonal index describes how much the average for that particular period tends to be above or below the grand average [27]. In Figure 3, seasonal index values for mid-year (June) and year-end (December) have higher values, significantly above 1.0 which is the expected value.

Table 4: (Non-Windows OSes) Vulnerability Discovery Seasonal Indexes

	Solaris	RH Linux	HP-UX	MAC OS X
Jan	0.6699	1.0133	1.0040	0.7980
Feb	0.7864	0.8533	0.8606	0.0887
Mar	0.8738	1.8667	0.7649	1.2709
Apr	0.8447	0.8000	0.5737	0.6502
May	0.7864	0.4800	0.8127	1.7734
Jun	0.9903	1.1200	0.7171	0.3842
Jul	1.0777	0.8533	0.7649	0.2365
Aug	1.2233	0.8533	0.8127	1.6552
Sep	0.8155	0.5333	1.1474	0.2365
Oct	1.1650	1.0133	1.1474	0.6502
Nov	0.7573	0.5867	0.8127	2.1872
Dec	2.0097	2.0267	2.5817	2.0690
χ_c^2	19.68	19.68	19.68	19.68
χ_s^2	49.3009	48.4400	63.6294	213.6256
p-value	8.356E-07	1.191E-06	1.949E-09	1.09E-39

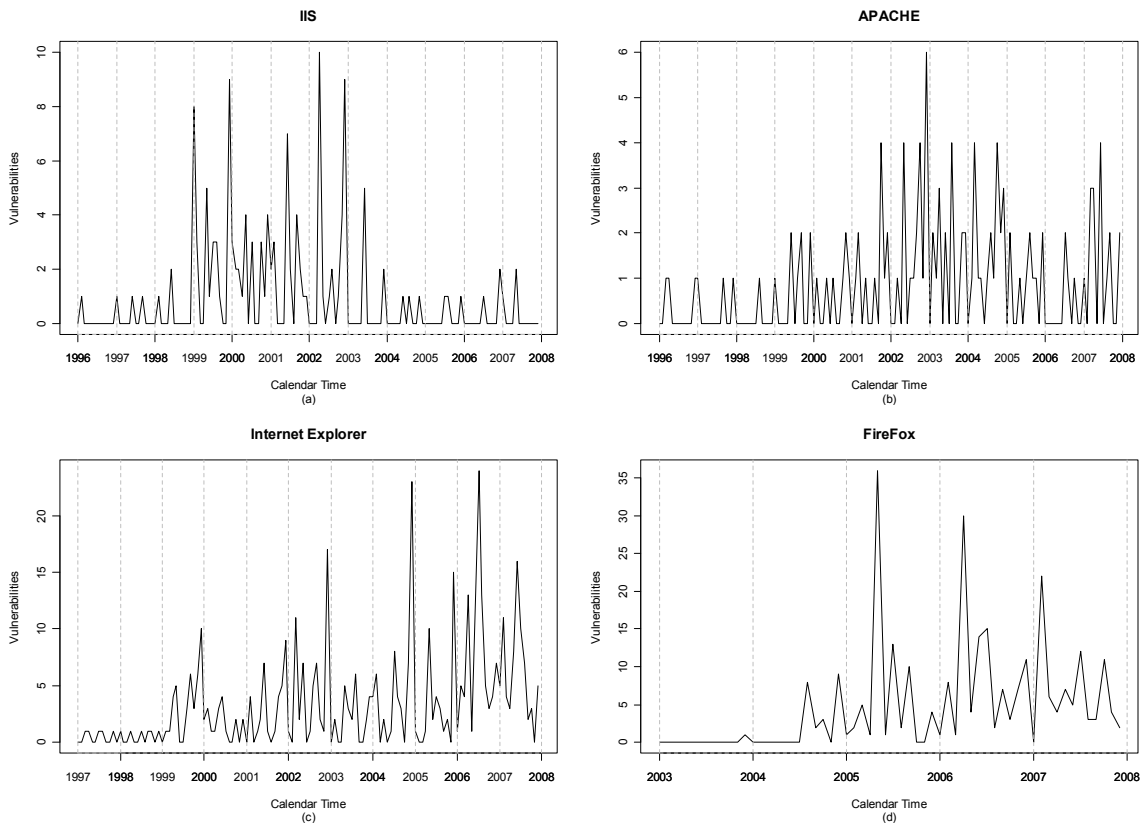

Figure 8. Vulnerabilities discovered for each month (Web S/B)

The mid-year peak may explain the peak in higher third quarter advisories for Microsoft products [29]. To evaluate the significance of non-uniformity of the distribution of the datasets, we conducted a chi-square test for the grand total of each month against the expected value (total vulnerabilities divided by 12). In Table 1, we can see that the Windows OSes yield extremely small p-values, thus we have a strong evidence of non-uniform distributions of vulnerability discovery rates [27], where the null hypothesis is that there is no seasonality in the dataset.

Autocorrelation function analysis. The autocorrelation function (ACF) in time series analysis is calculated by computing the correlation between a variable value and the successive values of the same variable after some time lags. In other words, ACF measures the linear relationship between time series observations separated by a lag of *k* time units [4, 28]. When an ACF value is located outside of defined confidence intervals at a lag *t,* there is a significant relationship associated with that time lag.

Figure 4 shows ACFs of the Windows OSes and Table 2 shows the corresponding ACF values with confidence intervals. In the figure, the upper and lower horizontal dotted lines represent 95% confidence intervals (figures 4, 6 and 10). Since the mid-year and year-end periods have the majority of big peaks in Figure 2, we expect that lags corresponding to about six months or its multiple would have their corresponding ACF values outside the confidence interval. In Figure 4(a) the lags for 0, 5, 6 and 11 months are outside of confidence interval; in other words, there are strong

Table 5: (Web S/B)
Vulnerability Discovery Seasonal Indexes

	IIS	Apache	IE	Firefox
Jan	1.3636	0.3000	0.4054	0.0857
Feb	0.9091	0.7000	0.8649	1.3714
Mar	0.1818	1.2000	0.5946	0.5143
Apr	1.0000	0.8000	0.7838	1.5000
May	1.0909	0.8000	1.1081	2.0143
Jun	1.5455	0.8000	1.2162	0.8571
Jul	1.0000	0.6000	1.4324	1.7143
Aug	0.6364	1.4000	1.1351	0.6429
Sep	0.5455	0.9000	0.7027	0.9429
Oct	0.5455	1.5000	0.5405	0.7286
Nov	0.6364	0.8000	0.7027	0.4714
Dec	2.5455	2.2000	2.5135	1.1571
χ_c^2	19.68	19.68	19.68	19.68
χ_s^2	46.0000	28.0000	130.4324	82.3142
p-value	3.23E-06	0.0032	1.42E-22	5.25E-13

autocorrelations with lags that are multiple of 6 confirming a seasonal pattern. In Figure 4(b) lags for 0, 5, 6, and 18 months, in Figure 4(c) for 0 and 5 months, in Figure 4(d) for 0, 2, 5 and 6 months are significantly different from zero of ACF which confirms a seasonal pattern. The same approach had been applied in [3, 4, 30] to prove seasonality in their datasets belonging to other fields of research. The partial autocorrelation function (PACF) is also can be used to detect trends and seasonality in a dataset [30]. We will now repeat the analysis for other software systems for comparison, as given in the next two subsections.

3.2 Non-Windows Operating Systems

We use the same analysis for the data for non-Windows OSes, namely, MAC OS X (1997-2007), HP-UX (1993-2007), SUN Solaris (1993-2007) and Red Hat Linux (1994-2007). They represent the entire versions (Figure 7) and MAC OS X includes server version also. Total cumulative numbers of vulnerabilities are 640, 258, 412, 503 for (a), (b), (c) and (d) respectively in the figure.

Seasonal index & Chi-square test. Table 4 and Figure 5 show seasonal indexes for each month of the four non-Windows OSes. Unlike the Windows OSs case, we do not find any consistent mid-year seasonal patterns among them. However, still some months tend to have more vulnerabilities than others. Also, all of them have higher incidences in December.

Autocorrelation function analysis. Figure 6 shows the ACF values for the Non-Windows OSes and Table 3 is for the corresponding ACF values. For Mac OS X, except lags number 16, 21, 22 and 23, all the lags are outside of the confidence interval. In HP-UX, only lags number 0, 1, 12 and 15 are outside of the confidence interval. For SUN Solaris, all the lags are outside of the confidence interval except lag number 7, 11, 13, 15, 18, 22 and 23. For Red Hat Linux, lags number 7, 13, 16, 18, 20, 22 and 23 are inside of the confidence interval.

3.3 Web Servers and Browsers

Here, data for web servers of Internet Information Services (IIS; 1996-2007) and Apache (1996-2007), and web browsers of Internet Explorer (IE; 1997-2007) and Firefox (2003-2007) are examined (Figure 8). Total cumulative numbers of vulnerabilities are 132, 129, 444, 280 for (a), (b), (c) and (d) respectively in the figure.

Seasonal index & Chi-square test. Table 5 and Figure 9 give the seasonal indexes for the web applications. IIS

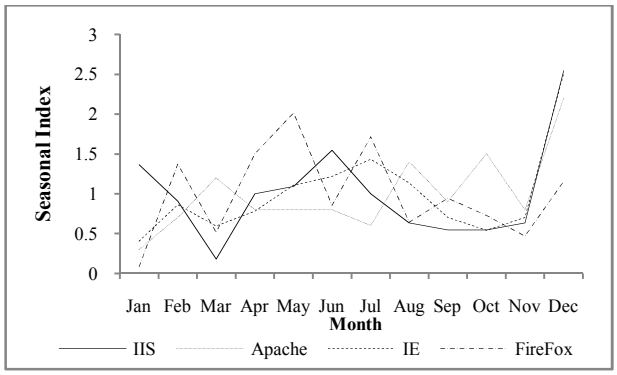

Figure 9. Autocorrelation function (Web S/W)

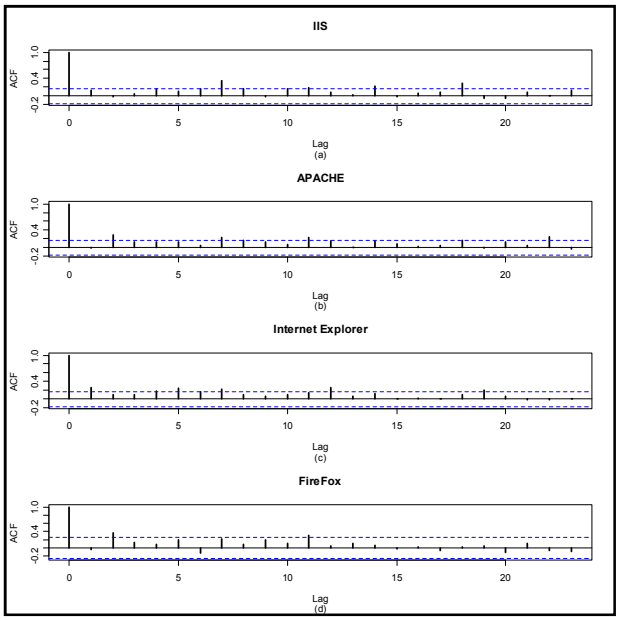

Figure 10. Autocorrelation function (Web S/W)

Table 6. Individual ACF Values for Figure 10

IIS ; 95% confidence interval = (-0.1633, 0.1633)							
1^{0}	0.134^{1}	-0.004^{2}	0.054^{3}	**0.178**^{4}	0.115^{5}	**0.178**^{6}	**0.359**^{7}
0.168^{8}	-0.02^{9}	**0.165**^{10}	**0.186**^{11}	0.097^{12}	0.033^{13}	**0.237**^{14}	-0.002^{15}
0.071^{16}	0.081^{17}	**0.288**^{18}	-0.046^{19}	-0.044^{20}	0.092^{21}	0.005^{22}	0.123^{23}
Apache ; 95% confidence interval = (-0.1705, 0.1705)							
1^{0}	-0.021^{1}	**0.287**^{2}	0.13^{3}	0.124^{4}	0.119^{5}	0.049^{6}	**0.231**^{7}
0.16^{18}	0.125^{9}	0.078^{10}	**0.222**^{11}	0.145^{12}	0.019^{13}	0.143^{14}	0.082^{15}
0.033^{16}	0.059^{17}	0.166^{18}	-0.016^{19}	0.138^{20}	0.047^{21}	**0.243**^{22}	-0.026^{23}
Internet Explorer ; 95% confidence interval = (-0.1633, 0.1633)							
1^{0}	**0.258**^{1}	0.11^{2}	0.112^{3}	**0.19**^{4}	**0.247**^{5}	**0.166**^{6}	**0.224**^{7}
0.1^{8}	0.058^{9}	0.1^{10}	0.153^{11}	**0.258**^{12}	0.069^{13}	0.115^{14}	-0.002^{15}
0.028^{16}	0^{17}	0.104^{18}	**0.198**^{19}	0.062^{20}	-0.019^{21}	-0.019^{22}	0.008^{23}
Firefox ; 95% confidence interval = (-0.2530, 0.2530)							
1^{0}	-0.044^{1}	**0.367**^{2}	0.131^{3}	0.085^{4}	0.183^{5}	-0.13^{6}	0.22^{7}
0.085^{8}	0.19^{9}	0.114^{10}	**0.308**^{11}	0.033^{12}	0.101^{13}	0.07^{14}	-0.027^{15}
0.017^{16}	-0.058^{7}	0.008^{18}	0.047^{19}	-0.107^{20}	0.103^{21}	-0.06^{22}	-0.091^{23}

Bold *fonts indicate outside of confidence intervals and* ^{Superscripts} *represent time lags*

and IE data displays a pattern similar to the Windows OSes. The mid-year and year-end periods tend to have more vulnerabilities than other months. Apache server shows a pattern nearly the opposite to IIS for every month except December. In December, all indexes are above the expected values. The small p-values (less than 0.05) in Table 5 confirms that the time series data is not uniformly distributed.

Autocorrelation function analysis. In Figure 10(a), lags for 0, 4, 6, 7, 8, 10, 11, 14 and 18 are outside of the confidence intervals. In Figure 10(b), lags for 0, 2, 7, 11 and 22, in Figure 10(c), lags for 0, 1, 4, 5, 6, 7, 12 and 19 in Figure 10(d), lags for 0, 2 and 11 are significantly different from zero of ACF. In the next section, we will attempt to identify the possible causes of the mid-year and year- end peaks.

4. Possible Factors Causing Seasonality

Rascorla [31] has suggested a possible cause for the year-end seasonality. He has suggested that a large number of vulnerabilities reported during the year-end may be a result of the end-of-year cleanup. However he has not discussed it in detail. It might be related with year-end report which needs to be completed before the end of the year for many organizations.

Further research is needed to answer why the vulnerability discovery in Microsoft products tends to peak in the mid-year months in addition to the year-end months. One possibility is that DEFCON [32], a major computer security related conference, happens in July or August. The potential conference participants might have a higher incentive [33] to find the vulnerabilities before the conference, to brag about, especially in popular Microsoft products. Figure 11 shows the number of occurrences of DEFCON and Black Hat [34] each calendar month; the two best well known conventions where vulnerabilities are announced. In the figure, (B) indicates Black hat, and (D) indicates DEFCON. The August-November period appears to be associated with release of a larger number of new Microsoft products. Figure 12 shows the release months for major versions of Windows OSs and Internet Explorer. The major versions of Windows and Internet Explorer tend to be released during June to November. This may be related to the starting of school semester and Christmas and New Year shopping seasons when many shoppers buy new computers with new operating systems known as IT seasonality. Authors in [20] also observed that occurrences of software security incidents increase during academic calendar. In December, emphasis may shift to identifying and handling vulnerabilities.

The reason for similar seasonality for the Windows operating systems, IIS and Internet Explorer could be due to the fact that IIS and Internet Explorer are distributed for only the Windows platform; vulnerabilities of the web servers and browsers may be correlated to the parent operating system platforms. According to the Symantec Internet Security Threat Report [1], more than half of the entire vulnerabilities in 2007 were web application related. In 2007, 64% of patch development time for Microsoft operating systems was for web browser and server.

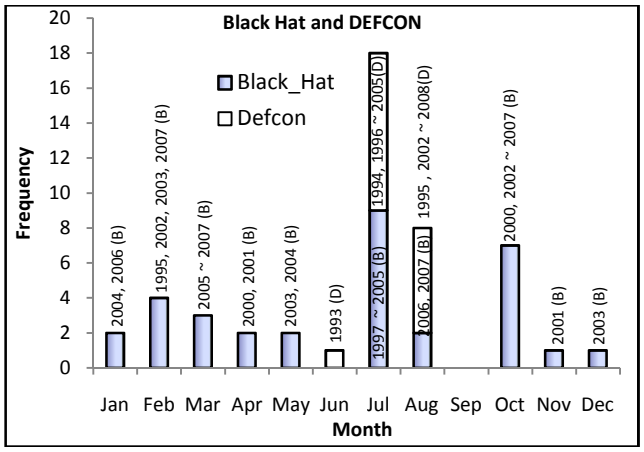

Figure 11. Black Hat and DEFCON conventions

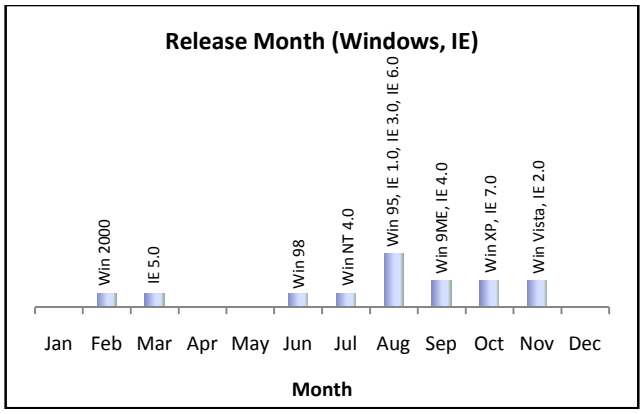

Figure 12. Windows and IE release month

5. Conclusion

Analysis of the vulnerability data using seasonal index and autocorrelation function approaches show that there is a significant seasonal pattern for major operating systems and popular web servers and browsers. For the all software systems examined, December has a high vulnerability discovery rate. A higher incidence during the mid-year periods is observed in Microsoft

products; Windows operating systems, IIS and Internet Explorer.

Further work is needed to develop methods for prediction of future vulnerability discovery trends using Box-Jenkins time series Model (ARIMA) which uses autocorrelation function, periodogram, spectral analysis and partial autocorrelation function analysis, and applying them into conjunction with the longer-term VDMs to improve the vulnerability discovery predictions.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments.

References

[1] Symantec. Symantec Global Internet Security Threat Report: Trends for July-December 07. Volume XIII, Published April 2008.

[2] S. Heston and R. Sadka, "Seasonality in the cross-section of stock returns," *Journal of Financial Economics*, Vol. 87 Issue 2, Feb 2008, pp. 418-445.

[3] N. Tran and D. A. Reed, "Automatic ARIMA Time Series Modeling for Adaptive I/O Prefetching," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 15, No. 2, Feb 2004, pp. 362-377.

[4] M. Rios, J. M. Garcia, J. A. Sanchez, and D. Perez, "A Statistical Analysis of the Seasonality in Pulmonary Tuberculosis," *European Journal of Epidemiology*, Vol. 16, No. 5, May 2000, pp. 483-488.

[5] A. Salehian, "ARIMA time series modeling for forecasting thermal rating of transmission lines," Transmission and Distribution Conference and Exposition, 2003 IEEE PES.

[6] J. Maes, S. Van Damme, P. Meire and F. Ollevier, "Statistical modeling of seasonal and environmental influences on the population dynamics of an estuarine fish community," *Marine Biology*, Vol. 145, No. 5, Oct 2004, pp. 1033-1042.

[7] J . Carrión-Baralt , C . Smith , E . Rossy-Fullana , R . Lewis-Fernández , K . Davis , and J . Silverman, "Seasonality effects on schizophrenic births in multiplex families in a tropical island," *Psychiatry Research*, Vol. 142 , Issue 1, 2006, pp 93 – 97.

[8] B. Jacobsen and N. Visaltanachoti, "The Halloween Effect in US Sectors," May 8, 2006. Available: http://ssrn.com/abstract=901088, Dec 28, 2008.

[9] N. Jegadeesh, "Evidence of Predictable Behavior of Security Returns," *The Journal of Finance*, Vol. 45, No. 3, Papers and Proceedings, Forty-ninth Annual Meeting, American Finance Association, Atlanta, Georgia, Dec 28-30, 1989 (Jul., 1990), pp. 881-898.

[10] H. Joh and Y. K. Malaiya, "Seasonality in Vulnerability Discovery in Major Software Systems," Fast Abstract, Proc. Int. Symp. Software Reliability Eng., Nov 2008, pp. 297-298.

[11] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing*, Third Edition, Prentice Hall PTR, 2003.

[12] O. H. Alhazmi and Y. K. Malaiya, "Modeling the Vulnerability Discovery Process," Proc. Int. Symp. Software Reliability Eng, Nov 2005, pp. 129-138.

[13] A. Ozment and S. E. Schechter "Milk or Wine: Does Software Security Improve with Age?", The 15th USENIX Security Symposium, Jul 31-Aug 4, 2006, pp. 93-104.

[14] A. Ozment, "Improving vulnerability discovery models," In Proceedings of the 2007 ACM Workshop on Quality of Protection (Alexandria, Virginia, USA, Oct 29 - 29, 2007). ACM, New York, NY, pp. 6-11.

[15] O. H. Alhazmi and Y. K. Malaiya, "Application of Vulnerability Discovery Models to Major Operating Systems," *IEEE Trans. Reliability*, Mar 2008, pp. 14-22.

[16] H. Joh, J. Kim and Y. K. Malaiya, "Vulnerability Discovery Modeling using Weibull Distribution," Fast Abstract, Proc. Int. Symp. Software Reliability Eng., Nov 2008, pp. 299-300.

[17] S-W. Woo, O. H. Alhazmi and Y. K. Malaiya, "An Analysis of the Vulnerability Discovery Process in Web Browsers", Proc. 10th IASTED Int. Conf. on Software Engineering and Applications, Nov 2006.

[18] S-W. Woo, O. H. Alhazmi and Y. K. Malaiya, "Assessing Vulnerabilities in Apache and IIS HTTP Servers", Proc. IEEE Int. Symp. on Dependable, Autonomic and Secure Computing (DASC'06), Sep-Oct 2006, pp. 103-110.

[19] J. Kim, Y. K. Malaiya and I. Ray, "Vulnerability Discovery in Multi-Version Software Systems," Proc. 10th IEEE Int. Symp. on High Assurance System Engineering (HASE), Dallas, Nov 2007, pp. 141-148.

[20] E. Condon, A. He and M. Cukier, "Analysis of Computer Security Incident Data Using Time Series Models," Proc. Int. Symp. Software Reliability Eng., Nov 2008, pp. 77-86.

[21] Forum of Incident Response and Security Teams (FIRST). Common Vulnerability Scoring System (CVSS). Available: http://www. first.org/cvss/ cvss-guide.html, Apr 10, 2008.

[22] National Institute of Standards and Technology. National Vulnerability Database. Available: http://nvd. nist.gov/download.cfm, Mar 31, 2008.

[23] Open Source Vulnerability Database (OSVDB). Available: http://osvdb.org, Apr 10, 2008.

[24] Department of Homeland Security. US-CERT Vulnerability Notes Database. Available: http://www.kb.cert.org/vuls, Apr 10, 2008.

[25] Secunia. Available: http://secunia.com, May 3, 2008.

[26] Common Vulnerability Scoring System, Available: http://cve.mitre.org , Dec 20, 2008.

[27] H. Arsham. Time-Critical Decision Making for Business Administration. Available: http:// home.ubalt. edu/ntsbarsh/Business-stat/stat-data/Forecast. htm#rseas onindx, Mar 31, 2008.

[28] B. L. Bowerman and R. T. O'connell, *Time Series Forecsting: Unified concepts and computer implementation,* 2nd Ed., Boston: *Duxbury Press,* 1987.

[29] A. Jaquith, *Security Metrics: Replacing Fear, Uncertainty, and Doubt*, Addison-Wesley Professional, 2007.

[30] A. Senter. A Summary of Forecasting Methods. Available:http://userwww.sfsu.edu/~efc/classes/biol710/timeseries/timeseries1.htm, Mar 31, 2008.
[31] E. Rescorla, "Is Finding Security Holes a Good Idea?" *IEEE Security and Privacy*, Vol. 3, No. 1. Jan-Feb 2005, pp. 14-19.
[32] DEFCON. DEF CON Communications, Inc. Available: https://www.defcon.org/main.html, Mar 31, 2008.
[33] A. Arora and R. Telang, "Economics of Software Vulnerability Disclosure," *IEEE Security and Privacy*, Jan 2005, pp. 20-25.
[34] Black Hat. Black Hat ™. Available: http://www. blackhat.com, Mar 31, 2008.