Revista Brasileira de Computação Aplicada, July, 2020

DOI: 10.5335/rbca.v12i2.10665
Vol. 12, Nº 2, pp. 122–133
Homepage: seer.upf.br/index.php/rbca/index

ISSN 2176-6649

ORIGINAL PAPER

An updated analysis of seasonal variations of the security vulnerability discovery process

Ariane Santos Borges[1], Paulo H. R. Gabriel [1], Rodrigo Sanches Miani [1]

[1]Faculty of Computing (FACOM), Federal University of Uberlandia (UFU)

*ariianeboorges@gmail.com; phrg@ufu.br; miani@ufu.br

Received: 2020-02-28. **Revised:** 2020-06-09. **Accepted:** 2020-06-30.

Abstract

Several factors may influence the security vulnerability discovery rates. The projection of these rates might help the development and the prioritization of software patches. Previous work studied the seasonal behaviors of the vulnerability discovery process for several operating systems and web-related software systems. We propose a replication study of an experiment conducted more than a decade ago to understand the changes in the dynamics of the security vulnerability discovery rates. In contrast to the findings from ten years ago, the investigated systems do not exhibit a year-end peak. Besides, the higher incidence during mid-year months for Microsoft operating systems was only noticed for the most recent Windows OSes: Windows 8.1 and Windows 10. These results highlight the relevance of reproducibility in scientific works. For cybersecurity studies, in particular, understanding the impact of specific findings over time might uncover unexpected trends and provide valuable insights.

Keywords: Cybersecurity; Security Vulnerability; Vulnerability Discovery Models

Resumo

Vários fatores podem influenciar as taxas de descoberta de vulnerabilidades de segurança. A projeção dessas taxas pode ajudar no desenvolvimento e na priorização de atualização de software. Trabalhos anteriores estudaram os comportamentos sazonais do processo de descoberta de vulnerabilidades para vários sistemas operacionais e sistemas de software relacionados à Web. O presente trabalho propõe a replicação de um experimento realizado há mais de uma década para entender as mudanças na dinâmica das taxas de descoberta de vulnerabilidades de segurança. Em contraste com os resultados encontrados anteriormente, os sistemas investigados não apresentam sazonalidade no fim do ano. Além disso, o aumento de vulnerabilidades no meio do ano para os sistemas operacionais Microsoft foi observada apenas para os sistemas Windows mais recentes: Windows 8.1 e Windows 10. Esses resultados destacam a importância da reprodutibilidade em trabalhos científicos. Na área de cibersegurança, em particular, a confirmação do impacto de resultados ao longo do tempo, pode servir para descobrir novas tendências e fornecer valiosos *insights* sobre o problema.

Palavras-Chave: Cibersegurança; Modelos de descoberta de vulnerabilidades; Vulnerabilidades de Segurança

1 Introduction

With the advent of new information and communication technologies (ICTs), both businesses and people in this environment need to understand that new features often imply new security issues. One way to exploit new security issues, and developing cyber-attacks involves finding vulnerabilities in computer systems. A typical example is exploiting vulnerabilities in protocols of operating systems such as *Microsoft Server Message Block* (SMB) from Microsoft Windows.

Cyber-attacks are increasing every day, and

statistics show that more than 70% of computer systems have vulnerabilities that could be exploited by an attacker (Positive Technologies, 2018). Data losses due to such vulnerabilities are typical of two types: either the data is confidential to the organization or is private to an individual. Regardless of the category, such attacks may result in loss of money or reputation.

Thus, managing security flaws in computer systems become an essential task not only to identify vulnerabilities but also to create a knowledge base on the subject and therefore, to understand possible patterns of behavior. Several references show the importance of vulnerability analysis for the understanding of the security risks of an organization (Goel and Mehtre, 2015, Shameli-Sendi et al., 2016, Singh et al., 2016).

In this context, Joh and Malaiya (2009) proposed a study on the seasonal variation in the process of security vulnerability discovery of software. The authors' idea was to investigate the *National Vulnerability Database* (NVD) between 1995 and 2007 and discover at what time of year vulnerability discovery rates were higher, detecting possible causes. Its results suggested a possible seasonality pattern, which means that it should be taken into account in decision-making and security vulnerability forecasting models.

Given that new technologies, computer systems, and consequently security issues have arisen since the work conducted by Joh and Malaiya (2009), it is essential to verify if the trends found in their work still exist. To do this, we analyzed security vulnerability data reported from 2008 to 2017 for the following systems: Windows Vista, Windows 7, Windows 8, Windows 8.1, Windows 10, Solaris, Red Hat, Ubuntu, Mac OS X, Internet Information Services (IIS), Internet Explorer (IE), Chrome, Firefox, and Apache. Additionally, two approaches were employed to find the patterns of seasonality: calculation of the seasonality index and analysis of the autocorrelation function (ACF) for each of the months. Since cyber-attacks are rapidly evolving, this type of study is important to understand the impact of temporal factors on information security issues. Previous studies investigate the behavior of security incidents and reinforce the importance of this research topic. They analyze factors such as how patterns in past movements are useful for forecasting incidents (Condon et al., 2008, Miani et al., 2015, Liu et al., 2015), the relationship between deployed security measures and security incidents frequency over time (Kuypers et al., 2016), and the prevalence of different sizes of data breaches over time (Liu et al., 2015).

The goal of this work is to use time-series analysis to evaluate the seasonality in the software vulnerability discovery process using a public database maintained by NIST. We want to i) identify software systems that exhibit seasonal trends (in the process of vulnerability discovery) among the data collected and ii) make a comparison of our results with the results of the previous in Joh and Malaiya (2009). In general terms, two types of results can be found: the confirmation of the patterns identified by Joh and Malaiya (2009) or the existence of new patterns.

The paper is organized as follows. Section 2 presents the theoretical foundation of security vulnerabilities. Section 3 discusses related work. Section 4 details the study that is being replicated (Joh and Malaiya, 2009) and also the proposed methodology. Section 5 presents the results. Conclusions and future work are provided in Section 6.

2 Security vulnerabilities

In the context of computer security, a vulnerability can be defined as a failure in a system that allows the realization and execution of an attack on a computer system (Aparecido and Bellezi, 2014). To exploit a vulnerability, an attacker must have at least one appropriate tool or technique that can connect to a system weakness, these tools or scripts are called *exploits* (Whitman and Mattord, 2012).

In general, vulnerabilities have a life cycle, as illustrated in Fig. 1 (Xiao et al., 2018). According to this figure, from the discovery of vulnerabilities, there is a race between developers/users and attackers: while developers try to release patches so that users can install and are no longer vulnerable, attackers attempt to exploit these vulnerabilities through automated tools before users install the patches.

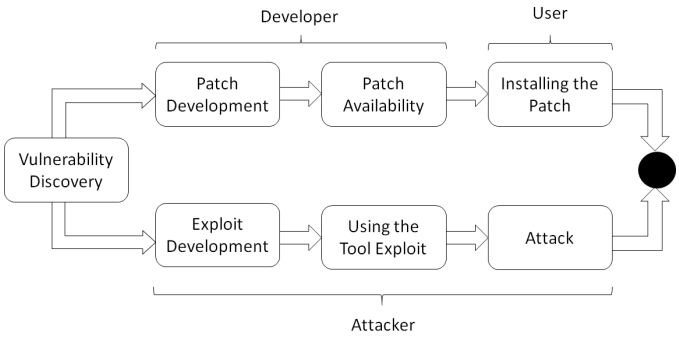


Figure 1: Vulnerability life cycle. Adapted from Xiao et al. (2018).

During the vulnerability discovery phase, when developers or attackers discover system failures, these vulnerabilities can be disclosed to the public. Such disclosure may occur either in public forums or through the release of an update for vulnerability correction (Xiao et al., 2018).

Data on software security vulnerabilities are often found using specialized search portals to store and maintain information about vulnerabilities and security holes, such as NVD (NIST, 2004), Secunia (Secunia, 2009), and US-CERT (CERT, 1991). We use the NVD portal as the main source of data, as done by previous work such as Joh and Malaiya (2009), Alhazmi et al. (2007), Roumani et al. (2015), Johnson et al. (2016), Han et al. (2017) and Anand et al. (2020).

3 Related Work

Several studies attempt to characterize computer security vulnerabilities. Since the goal of this work is to understand factors (seasonality) that might be used to forecast vulnerabilities, we focus on such type of study in this section.

One of the first works that investigate vulnerability prediction can be found in Alhazmi et al. (2007). The authors analyzed the number of vulnerabilities per unit of code size (density) from Windows and Red Had Linux. They found that values of vulnerability densities tend to fall within a range of values, similar to the defect density. They used this result to model the vulnerability discovery process using a logistic model.

Alhazmi and Malaiya (2008) describe the applicability and significance of several vulnerability discovery models for four operating systems (Windows XP, Windows 95, Red Hat Linux 6.2, and Red Hat Fedora). Vulnerability discovery models were examined using the Akaike information criterion (AIC) and chi-square test. The evaluation found that the AML model is generally better in the long run, with better performance for Windows 95, Red Hat Linux 6.2, and Red Hat Fedora.

Roumani et al. (2015) evaluate the use of time-series modeling to the vulnerability disclosure issue. They applied two techniques: autoregressive integrated moving average (ARIMA) and exponential smoothing to predict the number of vulnerabilities for five web browsers: Chrome, Firefox, Internet Explorer, Opera, and Safari. Their findings suggest that such modeling techniques can be useful for vulnerability prediction.

A recent study by Movahedi et al. (2019) compared the performance of time-series models and neural network models for predicting vulnerabilities. Authors found that neural network models outperform time-series models in all the cases in terms of prediction accuracy. Yasasin et al. (2020) uses several vulnerability modeling techniques (exponential smoothing, ARIMA, Croston, and neural network) but tackle a slightly different issue: predicting the number of post-release security vulnerabilities in subsequent periods of time. They found that the optimal forecasting methodology depends on the software and some techniques (ARIMA and Croston) outperforms exponential smoothing and neural network.

Regarding analyzing specific seasonal factors, Joh and Malaiya (2009) and Joh and Malaiya (2016) examined vulnerabilities disclosed in various software (Windows operating systems: Windows NT, Windows 95, Windows 98, Windows 2000, Windows XP and Windows 7; Mac OS X, Red Hat Linux Enterprise, AIX, Android and Chrome OS, and other systems: Apache, IIS, Internet Explorer, Firefox, Safari and Java (JRE)) to investigate possible annual variations in vulnerability discovery processes. They also examined the weekly frequency in the distribution of security updates (patches) and exploited vulnerabilities.

For all software groups examined, the authors found a higher vulnerability detection rate in certain months. In Microsoft products, they reported a higher incidence during the half-year periods. They also observed the periodical behavior of 7 days in the data of vulnerability scanning and confirmed that more activity occurs during the week than on the weekends. Specifically, vulnerability activity figures for Tuesday tended to be higher than the other days of the week. Results showed that periodicity needs to be considered for optimum allocation of resources and the evaluation of security risks.

The objective of this present study is to analyze the seasonality in two different moments: between 1995–2007 and between 2008–2017. The primary motivation for this is to evaluate the evolution of the behavior of security issues over the years.

4 Methodology

4.1 Previous work – Joh and Malaiya (2009)

Joh and Malaiya (2009) proposed a study on the seasonal variation in the process of vulnerability discovery of software security. The objective of the research was to find a seasonal pattern among the available databases of the studied software and to discover in which time of year the vulnerability rate tends to be higher and to detect possible causes of this event.

From the data collection, the authors analyzed the possible existence of seasonality using two statistical methods. The first was a seasonal index method measured with the chi-square test, which provides specific indices for each month, and the second was the autocorrelation function, which provides information for the correlated month. In this way, the authors were able to investigate the behavior of each of the systems.

The authors divided the operating systems into three categories: Windows, not Windows, and Web. The Windows operating systems were Windows NT, Windows XP, Windows 2000, and Windows Server 2003, non-Windows operating systems were SUN Solaris, Red Hat Linux, HP-UX and MAC OS X and Web applications were IIS, IE, Apache, and Firefox. The vulnerability database used by the authors were from 1995 to 2007.

Results suggest that, for the Windows category, the months of June and December had a high rate of vulnerability discovery, whereas February, March, April, and September had a low below-average vulnerability detection rate. Both non-Windows and Web categories showed that December had a high vulnerability discovery rate. According to the authors, this seasonality may be associated with the beginning of the semester in schools and the festive periods, such as Christmas and New Year, since people buy new computers with the operating systems described above.

4.2 Our Approach

Some of the operating systems used by Joh and Malaiya (2009) have been discontinued. For this reason, in this paper, we perform a data collection considering most recent Windows operating systems, such as

pub_date	type	name	seq	published	modified	severity	name3	vendor
15/08/2018	CVE	CVE-2008-0242	2008-0242	11/01/2008	28/09/2017	High	solaris	sun
15/08/2018	CVE	CVE-2008-0242	2008-0242	11/01/2008	28/09/2017	High	solaris	sun
15/08/2018	CVE	CVE-2008-0269	2008-0269	15/01/2008	28/09/2017	Medium	solaris	sun
15/08/2018	CVE	CVE-2008-0718	2008-0718	11/02/2008	28/09/2017	Medium	solaris	sun
15/08/2018	CVE	CVE-2008-0718	2008-0718	11/02/2008	28/09/2017	Medium	solaris	sun
15/08/2018	CVE	CVE-2008-0718	2008-0718	11/02/2008	28/09/2017	Medium	solaris	sun

Figure 2: Part of an XML file for the Solaris system

Windows Vista, Windows 7, Windows 8, Windows 8.1 and Windows 10, as well as GNU/Linux, Mac OS X. In addition, we consider the following software: Internet Explorer, Mozilla Firefox, Google Chrome, Microsoft IIS, and Apache HTTP Server. Besides, the database used corresponds to the years 2008 to 2017.

From these data, the same technique used in Joh and Malaiya (2009) was replicated, that is, the methods to extract seasonal indexes, chi-square test, and autocorrelation function. Through these statistical methods, it is possible to infer the behavior described by the vulnerabilities by month and year.

For better visualization of the data calculated by the seasonal index and autocorrelation function, we constructed a time series graphs for each system group and individual charts for each system, respectively. When both graphs present possible seasonality, we built another chart, called box-plot, to improve the the analysis.

The method used to evaluate the seasonality of the vulnerability discovery process involves the following steps:

i. Collection of vulnerability data for each system from the year 2008 to the year 2017;
ii. Calculation of the seasonality index for each system;
iii. Application of the autocorrelation function for each of the months for all systems;
iv. Individual analysis of the autocorrelation function to detect possible seasonality for each system;
v. Box-plot graphics construction only for systems that have the possibility of seasonality.

4.3 Data Collection

XML files are made available by NVD in *zip* format. Each year has its file with the necessary information. Therefore, when extracting the XML file, the next step is to import it into an Excel table for better visualization of the data about the vulnerabilities. Excel was chosen to import the data because of its filtering tools.

Fig. 2 shows a sample of how to import data from the XML file. In the file, there are precisely 35 header fields, but only three are essential for the data collection. The main ones are the *published*, *name3* and *vendor* fields, which respectively mean the vulnerability publication date, the operating system name where the vulnerability was found, and the company name responsible for the system.

The field *name3* displays all known software types, so the second step is to filter this field according to the type of software desired (in this case, Solaris). Automatically, all fields are updated with the respective information of that system. After choosing the system, the *published* column allows filtering of vulnerabilities by months of disclosure to facilitate accounting. Other information such as the name of the *(name)* vulnerability reported by CVE and the severity of the vulnerability is not relevant to this work, only the date of publication of the vulnerabilities.

For the present work, we used XML files from the year 2008 to 2017. For some of the chosen software, the XML files did not present all the necessary information of the ten years analyzed but presented information according to their launch or when vulnerabilities were not found. The following is a summary of the data collected:

- Windows 7: no vulnerabilities were found in the year 2008;
- Windows 8: there were no vulnerabilities found in the year 2008 until 2011;
- Windows 8.1: there were no vulnerabilities found in the year 2008 until 2012;
- Windows 10: no vulnerabilities were found between 2008 and 2014
- Solaris: we found vulnerabilities except in the year 2012;
- IIS: we found vulnerabilities except in the year 2011, 2014 and 2016;
- IE: we found vulnerabilities except in the year 2011;
- Apache: no vulnerabilities were found in 2008, 2011, 2015, 2016 and 2017;
- Firefox: there were no vulnerabilities found for the year 2017.

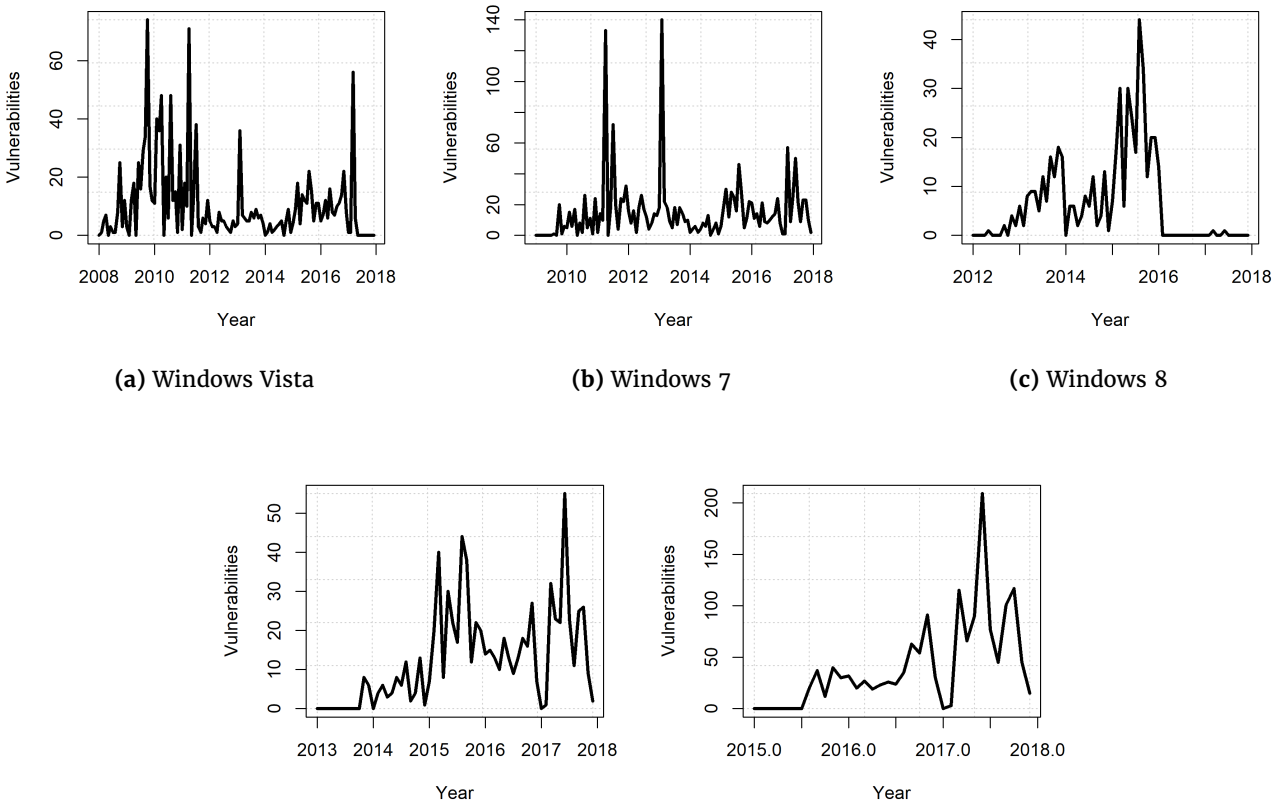
5 Results

This section presents the results of the seasonality analysis and compares it to the results from Joh and Malaiya (2009).

5.1 Data Analysis

The seasonality index is a measure widely used to evaluate seasonal trends and may indicate how much the average of a particular period tends to be above (or below) the expected value (Arsham, 1994). The monthly values of the seasonal index are given by Eq. (1):

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



(a) Windows Vista **(b)** Windows 7 **(c)** Windows 8

(d) Windows 8.1 **(e)** Windows 10

Figure 3: Accumulated vulnerabilities per year for the Windows OSes

where, s_i is the seasonal index for i^{th} month, d_i is the mean value of i^{th} month, and d is a grand average ([1]). Hence, for instance, 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 check whether the seasonal indexes are statistically significant, chi-square (χ^2) test for the null hypothesis H_0 has been calculated. To be statistically significant, χ^2 value (χ^2_s) must be greater than χ^2 critical value (χ^2_c) 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 Eq. (2) (Bowerman, 1987):

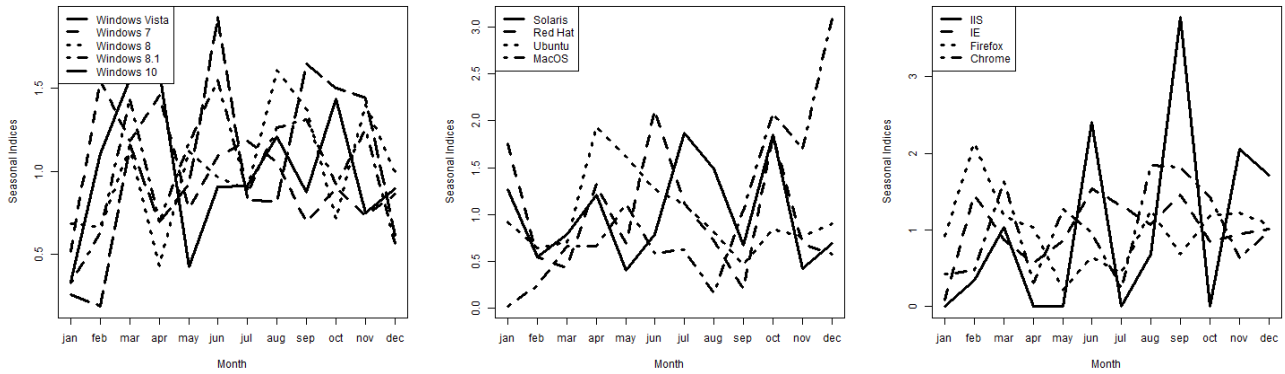
$$r_k = \frac{\sum_{t=b}^{n-k}(z_t - \bar{z})(z_{t+k} - \bar{z})}{\sum_{t=b}^{n}(z_t - \bar{z})} \qquad (2)$$

where $\bar{z} = \frac{\sum_{t=b}^{n} z_t}{(n-b+1)}$ represents the mean of the observations. Values of coefficient of autocorrelation close to zero indicate absence of seasonality in such an interval of observation, while values close to 1 show a significant relationship associated with such an interval of observation.

The value of r_k, along with the correlogram analysis (a graph with r_k values arranged in lag intervals) are used to establish criteria for a time series. According to Heckert et al. (2002) and Brockwell and Davis (2016), three possible situations can occur:

i. The time series is considered "random" or "stationary" if most of the autocorrelation coefficients are between the confidence intervals and no pattern was detected in the correlogram;
ii. The time series is considered "non-stationary" when the values of the autocorrelation coefficients decrease slowly as k increases, characterizing a trending behavior;
iii. The time series is considered "non-stationary strong" when the values of the correlation coefficients decrease slowly as k increases, but according to a periodicity (seasonal pattern).

[1] http://home.ubalt.edu/ntsbarsh/business-stat/stat-data/forecast.htm



(a) Windows OSes **(b)** Non-Windows OSes **(c)** Web

Figure 4: Seasonal indices

The main difference between situations 2 and 3 and that a non-stationary "moderate" time series does not usually exhibit periodic behavior. In these data types, it is common to find isolated r_k components related to a growing (or decreasing) trend or some types of patterns related to seasonality.

The results of the analysis for the groups of software using the index of seasonality, chi-square, and autocorrelation will be shown and explained in the next sections. Based on such methods, the following sequence of steps will be used to identify patterns of seasonality in the data series:

i. Analyze the time series of discovered vulnerability values per year – is it possible to view months with a greater or lesser number of vulnerabilities in all years?
ii. Compute and analyze the seasonality index – identify the months in which the index value is greater or less than one;
iii. Calculate and analyze the autocorrelation coefficients and the correlogram – exclude the software in which the series is considered stationary (that is, seasonally adjusted indices greater than or less than one previously found are potential outliers and are not associated with seasonal patterns);
iv. Investigate the box-plot of the discovered vulnerabilities per year and correlate with the seasonality index (the average of a given month is higher or was the product of an outlier?)

5.2 Windows Operating System

Fig. 3 shows the time series referring to the number of vulnerabilities found over the ten years. For these operating systems the total number of vulnerabilities was 1261 for Windows Vista (Fig. 3a), 1637 for Windows 7 (Fig. 3b), 471 for Windows 8 (Fig. 3c), 759 for Windows 8.1 (Fig. 3d), and 1466 for Windows 10 (Fig. 3e).

Time series for Windows systems provide no visible pattern. It is possible to notice a high dispersion in the data as a high concentration of vulnerabilities in some years – 2010 and 2011 for Windows Vista and 2015 for Windows 8, for example – and low concentration of vulnerabilities in others – 2014 for Windows Vista and the majority of the series for Windows 7.

With the help of the seasonal index presented in Table 1 and in Fig. 4a it is possible to note that some months have a higher probability of disclosure of vulnerabilities than the others. All systems have a low seasonal index in January and December, and March shows high seasonal index in all. However, in order to check whether such indices can be associated with seasonal patterns, additional tests should be done, such as, for example, calculating the autocorrelation coefficients.

The autocorrelation coefficients are shown in Table 2. It is possible to see that Windows 7 displays a stationary pattern (absence of seasonality), unlike other systems in which several coefficients autocorrelation coefficients are greater than one and are outside the confidence interval. The next step of the analysis involves the construction of box-plots for the four systems that exhibited non-stationary behavior.

With the help of the seasonal indexes shown in Table 1 and the box-plot shown in Fig. 5 it is possible to note that Windows Vista exhibits the following behavior: few vulnerabilities released in January and an increase in vulnerability between February and April. Windows 8 already has a decrease in vulnerabilities in January and April but an increase in the second half (August–September). Windows 8.1 has few vulnerabilities in January and December but an increase in June. Moreover, Windows 10 has fewer vulnerabilities in January and December while June exhibits an increase in such number.

5.3 Non-Windows Operating Systems

For non-windows operating systems, the total number of vulnerabilities was 756 for Solaris, 418 for Red Hat,

Table 1: Seasonal Vulnerability Discovery Indexes - Windows OSes

	WVista	W7	W8	W8.1	W10
Jan.	0.3331	0.5205	0.6879	0.3320	0.2619
Feb.	1.1039	1.5467	0.6624	0.6324	0.1883
Mar.	1.5511	1.1802	1.1465	1.4387	1.1623
Apr.	1.5987	1.4588	0.4331	0.6957	0.6958
May	0.4282	0.7770	1.1210	1.1700	0.9250
June	0.9040	1.0922	0.9682	1.5494	1.9236
July	0.9136	1.1802	0.8917	0.8696	0.8267
Aug.	1.2086	1.0556	1.6051	1.2648	0.8186
Sep.	0.8755	0.6891	1.3758	1.3123	1.6453
Oct.	1.4370	0.9016	0.7134	0.9170	1.4980
Nov.	0.7518	0.7330	1.4013	1.2490	1.4407
Dec.	0.8945	0.8650	0.9936	0.5692	0.6139
X_c^2	19.6751	19.6751	19.6751	19.6751	0.6578
X_s^2	187.48	144.8	52.287	103.5	397.26
p-value	2.20E-16	2.20E-16	2.20E-16	2.20E-16	2.20E-16

Table 2: Individual ACF values – Windows OSes

Windows Vista - 95% confidence interval: (-0.177,0.177)				
1.0^{0}	**0.180**^{1}	**0.230**^{2}	0.107^{3}	**0.349**^{4}
0.119^{5}	**0.315**^{6}	0.030^{7}	0.174^{8}	-0.055^{9}
0.219^{10}	0.001^{11}	**0.180**^{12}	-0.026^{13}	0.052^{14}
-0.058^{15}	0.043^{16}	-0.061^{17}	0.115^{18}	-0.050^{19}
-0.010^{20}	-0.021^{21}	0.086^{22}	-0.132^{23}	
Windows 7 - 95% confidence interval: (-0.1863,0.1863)				
1.0^{0}	0.058^{1}	0.044^{2}	0.183^{3}	0.083^{4}
-0.048^{5}	0.029^{6}	0.038^{7}	0.107^{8}	0.077^{9}
-0.048^{10}	-0.054^{11}	-0.083^{12}	-0.077^{13}	0.022^{14}
-0.026^{15}	-0.026^{16}	-0.133^{17}	0.011^{18}	0.07^{19}
-0.071^{20}	-0.042^{21}	**0.266**^{22}	-0.088^{23}	
Windows 8 - 95% confidence interval: (-0.2269,0.2269)				
1.0^{0}	**0.659**^{1}	**0.585**^{2}	**0.642**^{3}	**0.468**^{4}
0.330^{5}	**0.303**^{6}	0.159^{7}	0.032^{8}	0.017^{9}
-0.056^{10}	-0.145^{11}	-0.160^{12}	-0.187^{13}	-0.193^{14}
-0.150^{15}	-0.184^{16}	-0.152^{17}	-0.108^{18}	-0.160^{19}
-0.104^{20}	-0.062^{21}	-0.055^{22}	-0.071^{23}	
Windows 8.1 - 95% confidence interval: (-0.2477,0.2477)				
1.0^{0}	**0.497**^{1}	**0.332**^{2}	**0.474**^{3}	**0.314**^{4}
0.178^{5}	**0.275**^{6}	0.218^{7}	0.062^{8}	0.120^{9}
0.097^{10}	0.040^{11}	0.084^{12}	0.016^{13}	0.004^{14}
0.039^{15}	-0.040^{16}	-0.107^{17}	0.022^{18}	0.016^{19}
-0.018^{20}	0.057^{21}	0.058^{22}	-0.147^{23}	
Windows 10 - 95% confidence interval: (-0.3155,0.3155)				
1.0^{0}	**0.503**^{1}	0.259^{2}	**0.406**^{3}	0.288^{4}
0.063^{5}	0.130^{6}	0.223^{7}	0.043^{8}	0.033^{9}
0.083^{10}	0.071^{11}	0.040^{12}	-0.044^{13}	-0.070^{14}
-0.088^{15}	-0.097^{16}	-0.112^{17}	-0.074^{18}	-0.074^{19}
-0.145^{20}	-0.097^{21}	-0.123^{22}	-0.167^{23}	
Bold represents the values outside the confidence interval.				
The ^{superscript} represents the lags.				

2166 for Ubuntu and 9176 for Mac OS X. With the help of the seasonal index presented in Table 3 and in Fig. 4b it is possible to note that some months have a higher probability of disclosure of vulnerabilities than the others. All systems show low seasonal index in February and March, but none show high seasonal index at all.

The autocorrelation coefficients are shown in Table 4. Mac OS X exhibits a stationary pattern (absence of seasonality), unlike other systems in which several autocorrelation coefficients are greater than 1 and are outside the confidence interval. The next step of the analysis involves the construction of box-plots for the three systems that exhibited non-stationary behavior.

Using the seasonal indexes shown in Table 3 and the box-plot shown in Fig. 5 it is possible to note that Solaris displays the following behavior: an increase in vulnerabilities released in January and August and few vulnerabilities released in February. Red Hat shows an increase in vulnerabilities in June and October and low vulnerabilities released for the remaining months. Ubuntu has few vulnerabilities in September but an increase in April and June.

Table 3: Seasonal Vulnerability Discovery Indexes - No Windows OSes

	Solaris	RH Linux	Ubuntu	Mac OS X
Jan.	1.2627	1.7512	1.0193	0.0183
Feb.	0.5456	0.5455	0.6372	0.2380
Mar.	0.7794	0.4306	0.6921	0.6578
Apr.	1.2003	1.3206	1.9226	0.6591
May	0.4053	0.6890	1.6095	1.1155
June	0.7794	2.1244	1.2579	0.5885
July	1.8551	1.1196	1.1041	0.6212
Aug.	1.4809	0.7177	0.8075	0.1622
Sep.	0.7448	0.2010	0.4614	1.0488
Oct.	1.8395	1.8373	0.8404	2.0859
Nov.	0.4209	0.6890	0.7580	1.7014
Dec.	0.6859	0.5742	0.8899	3.1033
X_c^2	19.6751	19.6751	19.6751	19.6751
X_s^2	192.1	148.76	359.06	6808.0
p-value	2.20E-16	2.20E-16	2.20E-16	2.20E-16

5.4 Web Servers and Browsers

The number of vulnerabilities for Web Servers and Browsers was 35 for IIS, 2528 for IE, 60180 for Firefox,

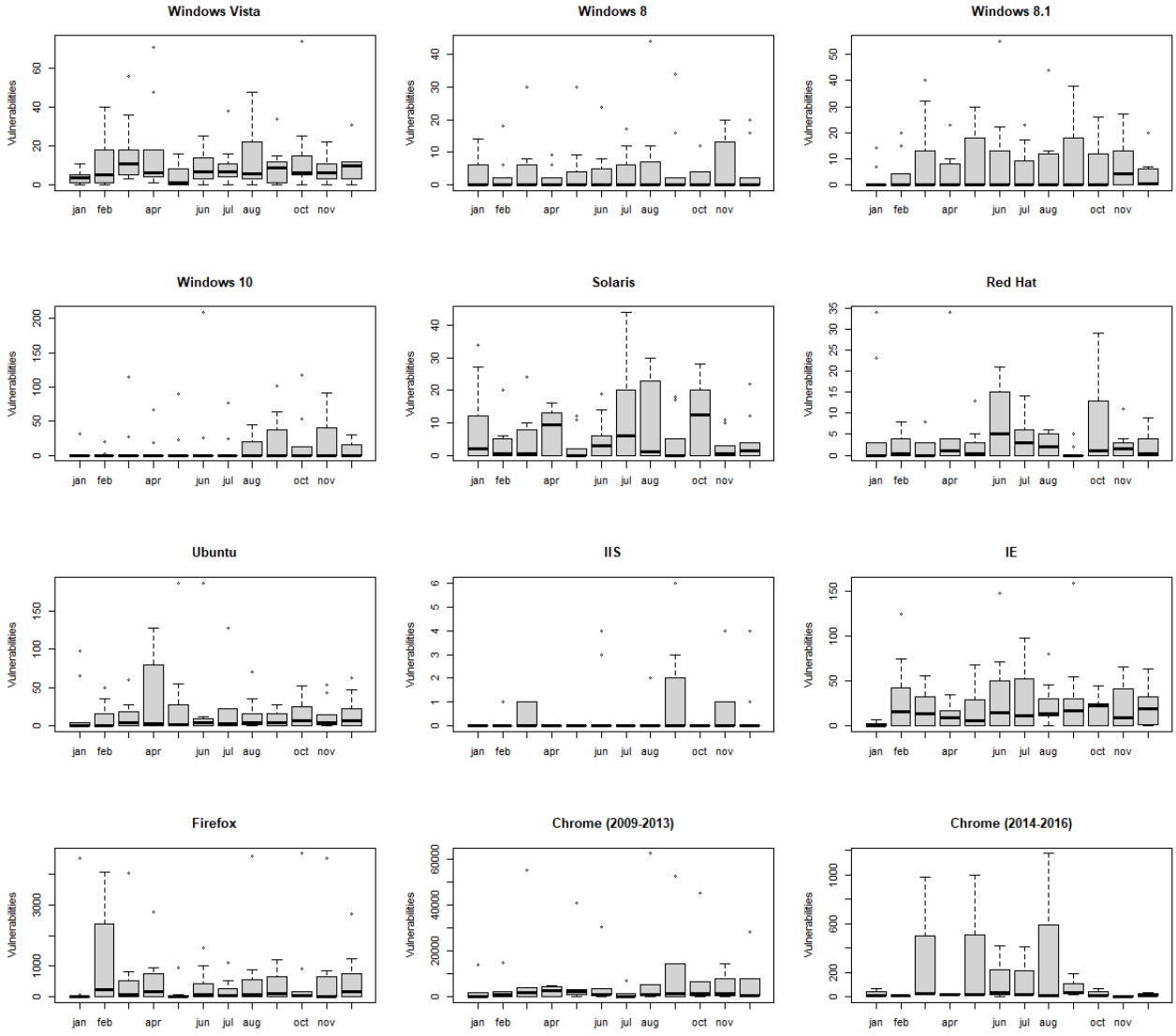


Figure 5: Box-plot for all systems

and 456021 for Chrome. Since the cumulative total of discovered vulnerabilities of the Apache Server from the year, 2008 through 2017 were only 19; such software was not considered in our analysis.

The seasonal index presented in Table 5 and in Fig. 4c show that all of the systems have low seasonal indexes in January, but none of them exhibit a high seasonal index. However, in order to check whether such indices can be associated with seasonal patterns, additional tests were conducted.

Table 6 show that all systems exhibit a non-stationary pattern where several coefficients of autocorrelation are higher than one and are out of range. The next step of the analysis involves the construction of box-plots for the four systems that exhibited non-stationary behavior.

The analysis of the box-plots (Fig. 5) and the seasonal indexes reveal that IIS exhibits the following behavior: an increase in vulnerabilities released in July and September and low vulnerabilities for the other months. The number of vulnerabilities associated with IE has increased in February and June and decreased in January. Firefox has fewer vulnerabilities in January and April but an increase in vulnerabilities in February.

Fig. 5 shows two box-plot graphics for Chrome. In the first graph, the months of April and July present a small number of vulnerabilities but an increase in March, August, and September. In the second graph, the months of March, May, and August show an increase in reported vulnerabilities and low vulnerabilities reported for most other months. By comparing the two graphs, it is possible to notice that the seasonality present in the first graph, has entirely changed concerning the second graph. This

Table 4: Individual ACF values – No Windows OSes

Solaris - 95% confidence interval: (-0.18639,0.18639)				
1.0^{0}	**0.245**^{1}	**0.203**^{2}	**0.480**^{3}	0.120^{4}
0.152^{5}	**0.348**^{6}	0.116^{7}	0.074^{8}	**0.227**^{9}
0.078^{10}	0.097^{11}	0.183^{12}	-0.010^{13}	-0.003^{14}
0.145^{15}	0.008^{16}	-0.086^{17}	0.040^{18}	-0.095^{19}
-0.164^{20}	-0.044^{21}	-0.151^{22}	-0.172^{23}	
Red Hat - 95% confidence interval: (-0.17703,0.17703)				
1.0^{0}	**0.024**^{1}	**0.037**^{2}	**0.364**^{3}	**0.076**^{4}
0.119^{5}	**0.230**^{6}	**0.205**^{7}	**0.073**^{8}	**0.080**^{9}
0.082^{10}	**0.013**^{11}	**0.164**^{12}	-0.027^{13}	0.048^{14}
0.108^{15}	0.091^{16}	0.028^{17}	0.101^{18}	0.060^{19}
-0.035^{20}	0.028^{21}	0.139^{22}	0.035^{23}	
Ubuntu - 95% confidence interval: (-0.17703,0.17703)				
1.0^{0}	**0.623**^{1}	**0.337**^{2}	**0.370**^{3}	**0.424**^{4}
0.369^{5}	**0.306**^{6}	**0.196**^{7}	**0.207**^{8}	**0.299**^{9}
-0.284^{10}	**0.234**^{11}	**0.233**^{12}	**0.277**^{13}	**0.264**^{14}
0.189^{15}	**0.202**^{16}	**0.216**^{17}	0.148^{18}	0.076^{19}
0.062^{20}	0.097^{21}	0.022^{22}	0.024^{23}	
Mac Os X - 95% confidence interval: (-0.17703,0.17703)				
1.0^{0}	0.058^{1}	-0.042^{2}	-0.053^{3}	0.037^{4}
-0.046^{5}	-0.048^{6}	0.106^{7}	-0.031^{8}	-0.037^{9}
-0.049^{10}	0.080^{11}	-0.045^{12}	-0.024^{13}	-0.047^{14}
-0.048^{15}	-0.035^{16}	-0.008^{17}	-0.018^{18}	-0.004^{19}
-0.011^{20}	-0.038^{21}	-0.039^{22}	0.026^{23}	
Bold represents the values outside the confidence interval.				
The ^{superscript} represents the lags.				

Table 6: Individual ACF values – Web

IIS - 95% confidence interval: (-0.2106,0.2106)				
1.0^{0}	-0.046^{1}	0.044^{2}	**0.345**^{3}	-0.057^{4}
-0.108^{5}	0.154^{6}	0.055^{7}	-0.152^{8}	**0.372**^{9}
0.091^{10}	-0.080^{11}	**0.319**^{12}	0.151^{13}	-0.055^{14}
0.216^{15}	-0.069^{16}	-0.109^{17}	-0.007^{18}	0.021^{19}
-0.119^{20}	-0.011^{21}	0.167^{22}	-0.065^{23}	
IE - 95% confidence interval: (-0.1974,0.1974)				
1.0^{0}	**0.413**^{1}	**0.449**^{2}	**0.546**^{3}	**0.350**^{4}
0.469^{5}	**0.320**^{6}	**0.290**^{7}	**0.359**^{8}	**0.297**^{9}
0.347^{10}	**0.252**^{11}	**0.410**^{12}	**0.210**^{13}	0.175^{14}
0.296^{15}	0.078^{16}	0.100^{17}	0.044^{18}	-0.046^{19}
0.012^{20}	-0.034^{21}	-0.102^{22}	-0.089^{23}	
Firefox - 95% confidence interval: (-0.1863,0.1863)				
1.0^{0}	**0.243**^{1}	**0.280**^{2}	**0.342**^{3}	-0.021^{4}
0.013^{5}	-0.036^{6}	-0.151^{7}	-0.112^{8}	-0.127^{9}
-0.031^{10}	0.097^{11}	0.021^{12}	0.088^{13}	**0.294**^{14}
0.060^{15}	**0.191**^{16}	0.182^{17}	0.050^{18}	0.008^{19}
0.039^{20}	-0.166^{21}	-0.085^{22}	-0.086^{23}	
Chrome - 95% confidence interval: (-0.177,0.177)				
1.0^{0}	**0.564**^{1}	**0.605**^{2}	**0.558**^{3}	**0.409**^{4}
0.416^{5}	**0.397**^{6}	**0.359**^{7}	**0.191**^{8}	**0.192**^{9}
0.073^{10}	0.082^{11}	0.043^{12}	0.041^{13}	-0.008^{14}
-0.012^{15}	-0.050^{16}	-0.061^{17}	-0.076^{18}	-0.093^{19}
-0.097^{20}	-0.104^{21}	-0.104^{22}	-0.111^{23}	
Bold represents the values outside the confidence interval.				
The ^{superscript} represents the lags.				

fact reinforces the importance of studying security events using different time windows.

Table 5: Seasonal Vulnerability Discovery Indexes - Web

	IIS	Chrome	IE	Firefox
Jan.	0.0000	0.4195	0.0807	0.9220
Feb.	0.3429	0.4723	1.4573	2.1406
Mar.	1.0286	1.6300	0.8687	1.1687
Apr.	0.0000	0.3099	0.5601	1.0351
May	0.0000	1.2653	0.8592	0.2154
June	2.4000	0.9620	1.5332	0.6526
July	0.0000	0.2399	1.3196	0.4407
Aug.	0.6857	1.8441	1.0680	1.2576
Sep.	3.7714	1.8143	1.4525	0.6837
Oct.	0.0000	1.4271	0.8497	1.1870
Nov.	2.0571	0.6378	0.9399	1.2241
Dec.	1.7143	0.9777	1.0111	1.0724
X_c^2	19.6751	19.6751	19.6751	19.6751
X_s^2	49.0	145470.0	401.75	13253.0
p-value	9.46E-07	2.20E-16	2.20E-16	2.20E-16

Of all the software analyzed, only Windows 7 and Mac OS systems did not present any seasonality pattern. January was the month with less incidence of vulnerabilities for Windows systems and Web applications. For non-windows OSes, February and September have a less incidence of vulnerabilities. For most of the studied systems, June is the month with a higher incidence of vulnerabilities.

5.5 Discussion and comparison with Joh and Malaiya (2009)

Table 7 illustrates the main similarities and differences between papers. For the systems studied, Joh and Malaiya (2009) found that June and December, for the Windows systems, had a high rate of vulnerability discovery. For systems other than Windows and Web applications, December showed a high incidence of vulnerability discovery rate. However, our results showed that this pattern has changed over the years. For example, Windows systems, non-Windows systems, and Web applications have a higher incidence of vulnerabilities in June. In other words, the year-end peak for these systems found by Joh and Malaiya (2009) does not exist anymore. Besides, it would be important to study the reasons that affect such changes.

An important issue found during the analysis is related to the collected data. In many situations, NVD has returned months in which no vulnerability was disclosed. Systems such as Apache and IIS, for example, obtained 19 and 35 vulnerabilities respectively over the ten years. In future work, it would be interesting to evaluate what actually happened in those months as a way of providing context for the analysis of the results. Our work shows the importance of performing such type of studies using updated data. That is, behaviors associated with information security issues, such as the modeling of security vulnerabilities, might change over time. This result emphasizes the importance of conducting cybersecurity replication studies in order to confirm or clarify specific outcomes. The usable privacy and security community and references (Coopamootoo

Table 7: Differences and similarities between the papers

Systems	Comparative	Joh and Malaiya (2009) - 1995 to 2007	Our paper - 2008 to 2017
Windows System	Months with a higher incidence of vulnerabilities	June and December	June
	Months with a lower incidence of vulnerabilities	February, March, April, and September	January
Different Windows System	Months with a higher incidence of vulnerabilities	December	June
	Months with a lower incidence of vulnerabilities	February	February and September
Web Applications	Months with a higher incidence of vulnerabilities	December	February and June
	Months with a lower incidence of vulnerabilities	September and November	January

and Groß, 2016, Herley and Van Oorschot, 2017) discuss the impact of moving security research forward in a more scientific fashion, and this involves replicating or extending relevant previously published studies and experiments.

6 Conclusion

The purpose of this paper was to find possible seasonalities in a vulnerability dataset of ten years composed of Windows operating systems, non-Windows operating systems, and Web applications. In summary, from the collected dataset of vulnerabilities, we compute the seasonal index for each system, applied the autocorrelation function, and constructed box-plot graphs for the systems that presented seasonality. Next, we compare our results with a previous work (Joh and Malaiya, 2009).

By comparing both results, we observed that the seasonality reported in the updated dataset (our paper) has changed. Joh and Malaiya (2009) concluded that Windows operating systems had seasonality in the months of June and December, while other operating systems and Web applications obtained seasonality in December. However, this study indicates that all of these system groups obtained seasonality in June. This result reinforces the relevance of replicating cyber-security studies in order to understand the impact of some findings over time.

For future work, we would like to explore the seasonality factor in order to forecast the future behavior of vulnerability disclosures. For this, we might use several time-series modeling techniques such as moving average, exponential smoothing, and ARIMA (*Auto-Regressive Integrated Moving Average*) models. For instance, the ARIMA model can be applied in cases where the data show evidence of non-stationarity, such as the data collected for this work. Evaluating the forecast capabilities of machine learning models is another important research topic. Finally, it would be interesting to consider other relevant software, for example, IoT applications.

References

Alhazmi, O. H. and Malaiya, Y. K. (2008). Application of vulnerability discovery models to major operating systems, *IEEE Transactions on Reliability* **57**(1): 14–22. http://dx.doi.org/10.1109/tr.2008.916872.

Alhazmi, O. H., Malaiya, Y. K. and Ray, I. (2007). Measuring, analyzing and predicting security vulnerabilities in software systems, *Computers & Security* **26**(3): 219–228. http://dx.doi.org/10.1016/j.cose.2006.10.002.

Anand, A., Bhatt, N. and Alhazmi, O. H. (2020). Modeling software vulnerability discovery process inculcating the impact of reporters, *Information Systems Frontiers* pp. 1–14. http://dx.doi.org/10.1007/s10796-020-10004-9.

Aparecido, C. A. G. M. and Bellezi, M. A. (2014). Análise de vulnerabilidades com OpenVAS e Nessus, *Tecnologias, Infraestrutura e Software* **3**(1): 34–44.

Arsham, H. (1994). Time series analysis for business forecasting. Available at http://home.ubalt.edu/ntsbarsh/Business-stat/stat-data/Forecast.htm.

Bowerman, B. L. (1987). *Time series forecasting: Unified concepts and computer implementation*, Duxbury Press, Boston, MA.

Brockwell, P. J. and Davis, R. A. (2016). *Introduction to time series and forecasting*, Springer, Switzerland.

CERT (1991). US-CERT | United States Computer Emergency Readiness Team. Available at https://www.us-cert.gov/.

Condon, E., He, A. and Cukier, M. (2008). Analysis of computer security incident data using time series models, *19th International Symposium on Software Reliability Engineering*, IEEE, pp. 77–86. http://dx.doi.org/10.1109/issre.2008.39.

Coopamootoo, K. P. L. and Groß, T. (2016). Evidence-based methods for privacy and identity management, *Privacy and Identity Management. Facing up to Next Steps*, Vol. 498 of *Advances in Information and Communication Technology book series*, Springer International Publishing, pp. 105–121. http://dx.doi.org/10.1007/978-3-319-55783-0_9.

Goel, J. N. and Mehtre, B. M. (2015). Vulnerability assessment & penetration testing as a cyber defence technology, *Procedia Computer Science* **57**: 710–715. http://dx.doi.org/10.1016/j.procs.2015.07.458.

Han, Z., Li, X., Xing, Z., Liu, H. and Feng, Z. (2017). Learning to predict severity of software vulnerability using only vulnerability description, *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, IEEE, pp. 125–136. http://dx.doi.org/10.1109/icsme.2017.52.

Heckert, N. A., Filliben, J. J., Croarkin, C. M., Hembree, B., Guthrie, W. F., Tobias, P. and Prinz, J. (2002). Handbook 151: Nist/sematech e-handbook of statistical methods, *Technical report*, National Institute of Standards and Technology, Gaithersburg, MD. Available at https://www.nist.gov/publications/handbook-151-nistsematech-e-handbook-statistical-methods.

Herley, C. and Van Oorschot, P. C. (2017). SoK: Science, security and the elusive goal of security as a scientific pursuit, *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, pp. 99–120. http://dx.doi.org/10.1109/sp.2017.38.

Joh, H. and Malaiya, Y. K. (2009). Seasonal variation in the vulnerability discovery process, *2009 International Conference on Software Testing Verification and Validation*, IEEE, pp. 191–200. http://dx.doi.org/10.1109/icst.2009.9.

Joh, H. and Malaiya, Y. K. (2016). Periodicity in software vulnerability discovery, patching and exploitation, *International Journal of Information Security* **16**(6): 673–690. http://dx.doi.org/10.1007/s10207-016-0345-x.

Johnson, P., Gorton, D., Lagerström, R. and Ekstedt, M. (2016). Time between vulnerability disclosures: A measure of software product vulnerability, *Computers & Security* **62**: 278–295. http://dx.doi.org/10.1016/j.cose.2016.08.004.

Kuypers, M., Paté-Cornell, E. and Maillart, T. (2016). An empirical analysis of cyber security incidents at a large organization, *Technical report*, Stanford University, Stanford, CA. Available at https://fsi.stanford.edu/publication/empirical-analysis-cyber-security-incidents-large-organization.

Liu, Y., Sarabi, A., Zhang, J., Naghizadeh, P., Karir, M., Bailey, M. and Liu, M. (2015). Cloudy with a chance of breach: Forecasting cyber security incidents, *24th USENIX Conference on Security Symposium*, USENIX Association, USA, pp. 1009–1024. http://dx.doi.org/10.5555/2831143.2831207.

Miani, R. S., Zarpelao, B. B., Sobesto, B. and Cukier, M. (2015). A practical experience on evaluating intrusion prevention system event data as indicators of security issues, *34th Symposium on Reliable Distributed Systems (SRDS)*, IEEE, pp. 296–305. http://dx.doi.org/10.1109/srds.2015.17.

Movahedi, Y., Cukier, M. and Gashi, I. (2019). Vulnerability prediction capability: A comparison between vulnerability discovery models and neural network models, *Computers & Security* **87**: 101596. http://dx.doi.org/10.1016/j.cose.2019.101596.

NIST (2004). NVD - Home. available at https://nvd.nist.gov/.

Positive Technologies (2018). Vulnerabilities in corporate information systems 2018, *Technical report*, Positive Technologies, London, UK. Available at https://www.ptsecurity.com/ww-en/analytics/corporate-vulnerabilities-2018/.

Roumani, Y., Nwankpa, J. K. and Roumani, Y. F. (2015). Time series modeling of vulnerabilities, *Computers & Security* **51**: 32–40. http://dx.doi.org/10.1016/j.cose.2015.03.003.

Secunia (2009). Computer security research – secunia. Available at https://secuniaresearch.flexerasoftware.com/commu-nity/research.

Shameli-Sendi, A., Aghababaei-Barzegar, R. and Cheriet, M. (2016). Taxonomy of information security risk assessment (ISRA), *Computers & Security* **57**: 14–30. http://dx.doi.org/10.1016/j.cose.2015.11.001.

Singh, H., Surender, J. and Pankaj, K. V. (2016). Penetration testing: Analyzing the security of the network by hacker's mind, *International Journal of Latest Technology in Engineering,Management & Applied Science* **5**(5): 56–60.

Whitman, M. E. and Mattord, H. J. (2012). *Principles of information security*, 4 edn, Course Technology, Boston, MA.

Xiao, C., Sarabi, A., Liu, Y., Li, B., Liu, M. and Dumitras, T. (2018). From patching delays to infection symptoms: Using risk profiles for an early discovery of vulnerabilities exploited in the wild, *27th USENIX Security Symposium (USENIX Security 18)*, USENIX Association, Baltimore, MD, pp. 903–918.

Yasasin, E., Prester, J., Wagner, G. and Schryen, G. (2020). Forecasting IT security vulnerabilities — An empirical analysis, *Computers & Security* **88**: 101610. http://dx.doi.org/10.1016/j.cose.2019.101610.