

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Ariane Santos Borges

**Análise da variação sazonal do processo de
descoberta de vulnerabilidades de segurança**

Uberlândia, Brasil

2018

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Ariane Santos Borges

Análise da variação sazonal do processo de descoberta de vulnerabilidades de segurança

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Rodrigo Sanches Miani

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2018

Ariane Santos Borges

Análise da variação sazonal do processo de descoberta de vulnerabilidades de segurança

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 20 de dezembro de 2018:

Rodrigo Sanches Miani
Orientador

Maria Adriana Vidigal de Lima
Professora

Renan Gonçalves Cattelan
Professor

Uberlândia, Brasil
2018

*Aos meus pais José Antônio Silveira Borges e Adélia Maria dos Santos Borges e
avó Divina Luzia Alves dos Santos*

Agradecimentos

Agradeço primeiramente a Deus por ter me dado saúde, força e por ter me guiado durante toda a minha jornada, permitindo que eu chegasse até aqui e realizado este sonho.

Aos meus pais e avó pelo amor, incentivo e apoio que sempre me deram. Por sempre acreditarem em mim e não medirem esforços para que mesmo com tantas dificuldades eu pudesse ter uma boa formação, tanto pessoal quanto acadêmica.

Ao meu irmão Felipe pelo carinho, apoio e incentivo.

Aos meus amigos pelo incentivo e compreensão durante toda essa jornada.

Ao meu namorado Weuler que sempre esteve ao meu lado durante a realização deste trabalho. Obrigada pelos conselhos, apoio, paciência e companheirismo.

A todos os professores que contribuíram para a minha formação, desde a educação básica até a graduação, tanto pelos ensinamentos técnicos quanto pelos conselhos e conversas que levarei para toda minha vida.

Ao meu orientador, Professor Doutor Rodrigo Miani, pela paciência, prontidão e confiança. Obrigada por me auxiliar e guiar desde o projeto de Iniciação Científica até na execução deste trabalho.

Resumo

Esse trabalho utiliza conceitos de análise de dados para avaliar a ocorrência de vulnerabilidades descobertas em sistemas operacionais Windows, sistemas operacionais diferentes do Windows e aplicações Web nos últimos dez anos usando uma base de dados pública mantida pelo NIST (*National Institute of Standards and Technology*). Ao longo da pesquisa foi utilizado o cálculo de índice de sazonalidade e a função de autocorrelação para investigar se as tendências encontradas em um estudo anterior continuam as mesmas, além de analisar vulnerabilidades presentes em softwares que não foram contemplados em tal trabalho. Os resultados encontrados indicam duas tendências de sazonalidade: aumento de vulnerabilidades reportadas em junho para todos os sistemas operacionais investigados e diminuição do número de vulnerabilidades em janeiro em todos os softwares analisados.

Palavras-chave: Vulnerabilidade de Segurança, Segurança da Informação, Modelos de Descoberta de Vulnerabilidade, Sazonalidade em Sistemas de Software.

Lista de ilustrações

Figura 1 – Ciclo de vida de uma vulnerabilidade (XIAO et al., 2018)	15
Figura 2 – Vulnerabilidades reportadas pelo NVD em abril de 2012	16
Figura 3 – Vulnerabilidade CVE-2011-4042	17
Figura 4 – Parte de um arquivo XML para o sistema Solaris	22
Figura 5 – Vulnerabilidades acumulada por ano - Windows OSes	26
Figura 6 – Índices Sazonais - Windows OSes	27
Figura 7 – Função de autocorrelação - Windows OSes	27
Figura 8 – Box-plot para os sistemas Windows Vista, Windows 8, Windows 8.1 e Windows 10	28
Figura 9 – Vulnerabilidades acumulada por ano - Não Windows OSes	30
Figura 10 – Índice Sazonal - Não Windows OSes	31
Figura 11 – Função de autocorrelação - Não Windows OSes	31
Figura 12 – Box-plot para os sistemas Solaris, Red Hat e Ubuntu - Não Windows OSes	32
Figura 13 – Vulnerabilidades acumulada por ano - Web	34
Figura 14 – Índice sazonal - Web	35
Figura 15 – Função de autocorrelação - Web	35
Figura 16 – Box-plot para os sistemas IIS, IE e Firefox - Web	36
Figura 17 – Box-plot para o Chrome - Web	37

Lista de tabelas

Tabela 1 – Índices Sazonais de Descoberta de Vulnerabilidade - Windows OSes	26
Tabela 2 – Valores de ACF individuais para a Figura 7	28
Tabela 3 – Índices Sazonais de Descoberta de Vulnerabilidade - Não Windows OSes . .	30
Tabela 4 – Valores de ACF individuais para a Figura 11 - Não Windows OSes	32
Tabela 5 – Índices Sazonais de Descoberta de Vulnerabilidades - Web	34
Tabela 6 – Valores de ACF individuais para a Figura 15 - Web	36
Tabela 7 – Diferenças e Semelhanças entre os trabalhos	38

Lista de abreviaturas e siglas

ACF	Análise da Função de Autocorrelação
AIC	Critério de Informação Akaike
AML	Modelo Logístico Alhazmi-Malaiya
AT	Modelo Termodinâmico
ARIMA	Modelo Auto-Regressivo Integrado de Médias Móveis
CVE	Common Vulnerabilities and Exposures
HTTP	Hypertext Transfer Protocol
IE	Internet Explorer
IIS	Internet Information Services
JRE	Java Runtime Environment
JW	Modelo Weibull
LN	Modelo Linear
LP	Modelo Regressão de Poisson
SMB	Microsoft Server Message Block
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OS	Sistema Operacional
OSes	Sistemas Operacionais
OSVDB	Open Source Vulnerability Database
RE	Modelo Exponencial
RGB	Modelo de Crescimento de Confiabilidade
RQ	Modelo Quadrático
SO	Sistema Operacional
TI	Tecnologia da Informação

TIC	Tecnologia da Informação e Comunicação
VDM	Modelo de Descoberta de Vulnerabilidade
YF	Modelo Distribuição Normal Dobrada

Sumário

1	INTRODUÇÃO	11
1.1	Objetivos	12
1.2	Resultados esperados	12
1.3	Organização do Trabalho	12
2	FUNDAMENTAÇÃO TEÓRICA	14
2.1	Segurança da Informação	14
2.2	Vulnerabilidades	14
2.3	Trabalhos Relacionados	17
3	METODOLOGIA	20
3.1	Trabalho de Joh e Malaiya (2009)	20
3.2	Esta proposta	20
3.3	Coleta dos dados	21
4	RESULTADOS	23
4.1	Análise dos dados	23
4.1.1	Sistema Operacional Windows	25
4.1.2	Sistemas Operacionais diferentes do Windows	29
4.1.3	Servidores e Navegadores Web	32
4.2	Comparação entre os trabalhos	37
5	CONCLUSÃO	39
	REFERÊNCIAS	40

1 Introdução

Com o advento de novas tecnologias de informação e comunicação (TIC), tanto as empresas quanto as pessoas que estão inseridas nesse ambiente precisam entender que novas funcionalidades muitas vezes implicam em novos problemas de segurança. Uma forma usada para explorar novos problemas de segurança envolve encontrar vulnerabilidades em sistemas computacionais. Um exemplo comum é a exploração de vulnerabilidades em sistemas operacionais, como no sistema operacional Windows, em que é possível executar códigos maliciosos a partir de vulnerabilidades em protocolos como o *Microsoft Server Message Block* (SMB).

O uso de aplicativos móveis e aplicações Web torna estes sistemas vulneráveis a ataques cibernéticos. Os ataques cibernéticos estão aumentando a cada dia e as estatísticas apontam que mais de 70% dos sistemas computacionais têm vulnerabilidades que poderiam ser exploradas por um hacker ou, pior, já foram exploradas ([POSITIVE TECHNOLOGIES, 2018](#)). As perdas de dados devido a essa exploração são tipicamente de dois tipos: ou os dados são confidenciais para a organização ou são privados para um indivíduo. Independentemente da categoria, tais ataques resultam em perda de dinheiro ou reputação.

Dessa forma, gerenciar as falhas de segurança nos sistemas computacionais se torna uma tarefa essencial não somente para identificar as vulnerabilidades, mas também para criar uma base de conhecimento sobre o assunto e assim compreender possíveis padrões de comportamento. Diversos trabalhos mostram a importância da análise de vulnerabilidade para a compreensão dos riscos de segurança de uma organização ([GOEL; MEHTRE, 2015](#)) ([SINGH; SURENDER; PANKAJ, 2016](#)) ([SHAMELI-SENDI; AGHABABAEI-BARZEGAR; CHERIET, 2015](#)).

Dentro deste contexto, [Joh e Malaiya \(2009\)](#) propuseram um estudo sobre a variação sazonal no processo de descoberta de vulnerabilidade de segurança em softwares. A ideia dos autores foi investigar a base de dados de vulnerabilidades de segurança *National Vulnerability Database* (NVD) mantida pelo *National Institute of Standards and Technology* (NIST) entre os anos de 1995 até 2007 e descobrir, em qual período do ano, as taxas de descoberta de vulnerabilidade foram mais altas e detectar possíveis causas desse acontecimento. Seus resultados sugeriram um possível padrão de sazonalidade, o que significa que o mesmo deve ser levado em consideração em tomadas de decisões e em modelos de previsão de vulnerabilidades de segurança.

Dado que novas tecnologias, sistemas computacionais e, consequentemente, problemas de segurança surgiram desde o trabalho conduzido por [Joh e Malaiya \(2009\)](#), seria importante verificar se as tendências encontradas em seu trabalho ainda existem. Para isso, serão analisados dados de vulnerabilidades de segurança reportados a partir de 2008 para os seguintes

sistemas operacionais Windows, não-Windows e Web: Windows Vista, Windows 7, Windows 8, Windows 8.1, Windows 10, Solaris, Red Hat, Ubuntu, Mac OS X, IIS, IE, Chrome, Firefox e Apache. Além disso, serão usadas duas abordagens para encontrar os padrões de sazonalidade: cálculo do índice de sazonalidade e análise da função de autocorrelação (ACF) para cada um dos meses. Esse tipo de estudo é importante para compreender o impacto que os fatores temporais exercem sobre as questões de segurança da informação. Estudos anteriores sobre a sazonalidade encontrada em incidentes de segurança reportados em redes acadêmicas reforçam a importância desse tópico e podem ser vistos em (CONDON; HE; CUKIER, 2008) e (MIANI, 2013).

1.1 Objetivos

O objetivo geral desse trabalho é utilizar conceitos de análise de dados para avaliar a sazonalidade no processo de descoberta de vulnerabilidade em softwares nos últimos dez anos usando uma base de dados pública mantida pelo NIST. Para o estudo desse problema faz-se necessário uma preparação que envolve os seguintes conceitos: princípios básicos de segurança da informação (confidencialidade, integridade, autenticidade e disponibilidade) e passos que envolvem o ataque a um computador ou a uma rede de computadores.

O objetivo geral pode ser dividido nos seguintes objetivos específicos:

- Escolher os métodos estatísticos adequado para o problema;
- Identificar os softwares que exibem tendências de sazonalidade (no processo de descoberta de vulnerabilidades) entre os dados coletados;
- Traçar uma comparação dos resultados deste trabalho com os resultados do trabalho propostos em (JOH; MALAIYA, 2009).

1.2 Resultados esperados

Ao fim deste trabalho, espera-se identificar a existência de padrões sazonais no processo de descoberta de vulnerabilidades de software. Em linhas gerais dois tipos de resultados poderão ser encontrados: a confirmação dos padrões identificados por Joh e Malaiya (2009) ou a existência de novos padrões.

1.3 Organização do Trabalho

O trabalho está dividido da seguinte forma. O Capítulo 2 apresenta a fundamentação teórica: conceitos básicos e trabalhos relacionados. O Capítulo 3 apresenta uma seção sobre a

metodologia aplicada no trabalho replicado, uma seção sobre a metodologia proposta para este trabalho e uma seção referente à coleta dos dados. O Capítulo 4 é dividido em duas seções: uma seção referente à análise dos dados e outra para comparação entre o atual trabalho e o trabalho replicado. Conclusões e trabalhos futuros são apresentados no Capítulo 5.

2 Fundamentação Teórica

Este capítulo apresenta um levantamento bibliográfico sobre segurança da informação, vulnerabilidade de segurança e modelos de descoberta de vulnerabilidades (VDM), três conceitos que fundamentam o trabalho.

2.1 Segurança da Informação

Segurança da Informação é o conjunto de orientações, normas, procedimentos, políticas e demais ações que têm por objetivo proteger o recurso informação, possibilitando que o negócio da organização seja realizado e a sua missão seja alcançada (FONTES, 2017).

Proteger a informação é responsabilidade de cada pessoa na organização em que coopera e cabe à organização orientar seus colaboradores em relação à proteção da informação. De acordo com Spanceski (2004), uma das formas de proteger a informação é conhecer os seguintes pilares da área da segurança da informação:

- Autenticidade: assegura que as informações vieram da fonte anunciada;
- Confidencialidade: protege as informações contra leituras ou cópias por pessoas não autorizadas;
- Integridade: garante que as informações não sofram alterações indevidas sem a permissão dos proprietários das informações;
- Disponibilidade: assegura que as informações estejam acessíveis e utilizáveis aos usuários sempre que necessários.

A importância da segurança cresce ainda mais quando os dados da organização são expostos não somente para os colaboradores mas para os usuários ou clientes finais. A falta de um planejamento em segurança pode acarretar problemas em cada um dos pilares descritos acima. Por exemplo, se um banco de dados de senhas não foi devidamente protegido contra leitura por pessoas não autorizadas, um eventual vazamento do mesmo pode representar um grande risco à confidencialidade dos envolvidos.

2.2 Vulnerabilidades

No contexto da segurança de computadores, uma vulnerabilidade pode ser definida como uma falha em um sistema que permite a realização e a concretização de um ataque em um

sistema computacional (APARECIDO; BELLEZI, 2014). Para explorar uma vulnerabilidade, um atacante deve ter pelo menos uma ferramenta ou técnica aplicável que possa conectar a uma fraqueza do sistema, essas ferramentas são chamadas de *exploits* (WHITMAN; MATTORD, 2011).

De maneira geral, vulnerabilidades possuem um ciclo de vida conforme a Figura 1, onde a partir da descoberta de vulnerabilidades há uma corrida entre desenvolvedores/usuários e atacantes. Enquanto os desenvolvedores tentam liberar atualizações (*patches*) para que os usuários possam instalar e não estarem mais vulneráveis, os atacantes tentam explorar essas vulnerabilidades por meio de ferramentas automatizadas antes que os usuários instalem as correções.

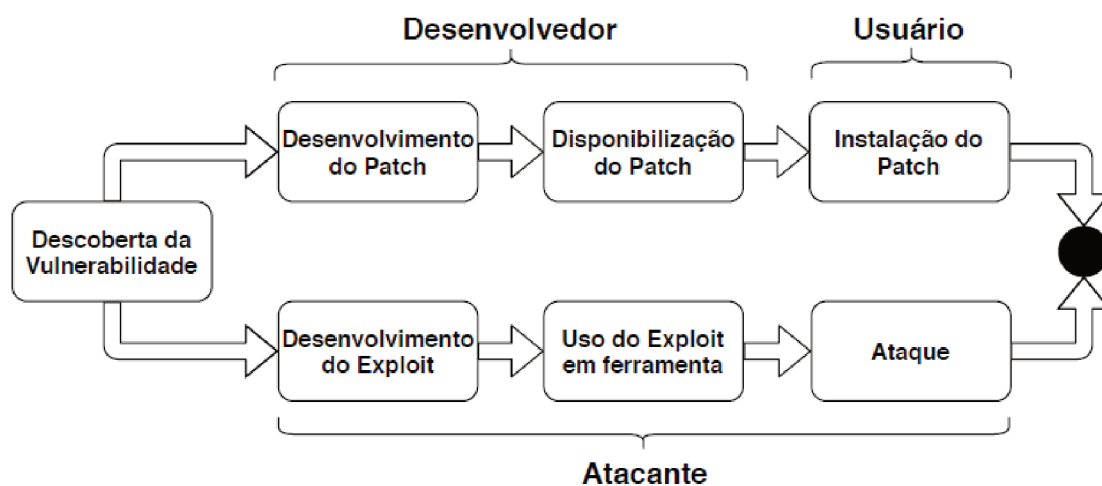


Figura 1 – Ciclo de vida de uma vulnerabilidade (XIAO et al., 2018)

Durante a fase de descoberta de vulnerabilidades, quando desenvolvedores ou atacantes descobrem falhas no sistemas, essas vulnerabilidades podem ser divulgadas ao público. Essa divulgação pode ocorrer ou em fóruns públicos ou através da liberação de uma atualização para a correção da vulnerabilidade.

Dados sobre vulnerabilidades de segurança em software geralmente são encontradas usando portais de busca especializados em armazenar e manter informações sobre vulnerabilidades e falhas de segurança, tais como as organizações NVD (NIST, 2004), Secunia (SECUNIA, 2009), US-CERT (CERT, 1991) e *Open Source Vulnerability Database* (OSVDB, 2002). Para este trabalho, assim como para a maioria dos estudos acadêmicos sobre o assunto, a coleta dos dados é feita com o auxílio do portal de busca NVD.

O NVD é uma referência na coleta de dados sobre vulnerabilidade e é sincronizado com o CVE (*Common Vulnerabilities and Exposures*) (CVE, 1985). Enquanto o CVE cadastra vulnerabilidades, o NVD categoriza e avalia os riscos delas.

O CVE é um banco de dados público em que todos os interessados podem obter acesso

a informações sobre vulnerabilidades. O principal gestor do CVE é o MITRE (*Massachusetts Institute of Technology's Digital Computer Laboratory*) e sua proposta não é somente divulgar informações sobre vulnerabilidades de segurança, e sim padronizar essas informações, com a ajuda do NVD (PEOTTA; GONDIM, 2006).

Um exemplo de vulnerabilidade reportada pelo NVD é mostrada na Figura 2. A plataforma do NVD disponibiliza todos os CVEs existentes desde 1988 até o ano atual. Para cada ano é possível escolher um mês e visualizar todas as vulnerabilidades divulgadas naquele período, mas não são todos os anos que tem os doze meses divulgados. Foi escolhido para esse exemplo o mês de abril do ano de 2012.

April 2012

Below is a list of CVEs for the selected month.

NOTE: The CVEs shown below have a **release date** in the year and month chosen. The CVE ID may show a year value that does not match the release date, however, the release date will fall within the chosen year and month.

228 entries found for April 2012

CVE-2011-4042	CVE-2011-4043	CVE-2011-4044	CVE-2011-4045	CVE-2011-4535	CVE-2011-5084
CVE-2011-5085	CVE-2012-0221	CVE-2012-0222	CVE-2012-0225	CVE-2012-0226	CVE-2012-0228
CVE-2012-0246	CVE-2012-0257	CVE-2012-0258	CVE-2012-1515	CVE-2008-7309	CVE-2008-7310
CVE-2008-7311	CVE-2011-3066	CVE-2011-3067	CVE-2011-3068	CVE-2011-3069	CVE-2011-3070
CVE-2011-3071	CVE-2011-3072	CVE-2011-3073	CVE-2011-3074	CVE-2011-3075	CVE-2011-3076
CVE-2011-3077	CVE-2011-5000	CVE-2012-0128	CVE-2012-0129	CVE-2012-0130	CVE-2012-0131
CVE-2012-0132	CVE-2012-0249	CVE-2012-0250	CVE-2012-0255	CVE-2012-0327	CVE-2012-1335
CVE-2012-1336	CVE-2012-1337	CVE-2012-1777	CVE-2012-1982	CVE-2012-2053	CVE-2012-2054
CVE-2012-2055	CVE-2012-0724	CVE-2012-0725	CVE-2012-1237	CVE-2012-1238	CVE-2012-1239
CVE-2012-1902	CVE-2011-3175	CVE-2011-3176	CVE-2011-4188	CVE-2012-0742	CVE-2012-2215
CVE-2012-0146	CVE-2012-0147	CVE-2012-0151	CVE-2012-0158	CVE-2012-0163	CVE-2012-0168
CVE-2012-0169	CVE-2012-0170	CVE-2012-0171	CVE-2012-0172	CVE-2012-0177	CVE-2012-0774
CVE-2012-0775	CVE-2012-0776	CVE-2012-0777	CVE-2012-1182	CVE-2012-0041	CVE-2012-0042
CVE-2012-0043	CVE-2012-0066	CVE-2012-0067	CVE-2012-0068	CVE-2012-1030	CVE-2012-1036
CVE-2012-1499	CVE-2012-1593	CVE-2012-1594	CVE-2012-1595	CVE-2012-1596	CVE-2012-1672
CVE-2012-1673	CVE-2012-1992	CVE-2012-2156	CVE-2012-2210	CVE-2012-2223	CVE-2012-2224
CVE-2012-2225	CVE-2011-3846	CVE-2012-0133	CVE-2012-1574	CVE-2012-2230	CVE-2010-4666
CVE-2011-1777	CVE-2011-1778	CVE-2011-1779	CVE-2011-4874	CVE-2011-4880	CVE-2011-4881
CVE-2011-4882	CVE-2011-4883	CVE-2012-0036	CVE-2012-1805	CVE-2012-1806	CVE-2012-1807
CVE-2012-1808	CVE-2012-1809	CVE-2012-1240	CVE-2012-1241	CVE-2011-2478	CVE-2012-0942
CVE-2012-1180	CVE-2012-1518	CVE-2012-1923	CVE-2012-1979	CVE-2012-1984	CVE-2012-1985
CVE-2012-2089	CVE-2012-2267	CVE-2012-2268	CVE-2011-4871	CVE-2011-5086	CVE-2011-5087
CVE-2011-5088	CVE-2011-5089	CVE-2012-0135	CVE-2012-0253	CVE-2012-0278	CVE-2012-0883
CVE-2012-1799	CVE-2012-1800	CVE-2012-1801	CVE-2012-1802	CVE-2012-1993	CVE-2012-0134
CVE-2012-2110	CVE-2012-2396	CVE-2012-0406	CVE-2012-0407	CVE-2012-2236	CVE-2012-2269
CVE-2012-2270	CVE-2012-2273	CVE-2012-2397	CVE-2012-2398	CVE-2012-1243	CVE-2012-2234
CVE-2012-2399	CVE-2012-2400	CVE-2012-2401	CVE-2012-2402	CVE-2012-2403	CVE-2012-2404
CVE-2012-0216	CVE-2012-0708	CVE-2012-0726	CVE-2012-0740	CVE-2012-0743	CVE-2012-0946
CVE-2012-1113	CVE-2012-1575	CVE-2012-2405	CVE-2012-2131	CVE-2012-0467	CVE-2012-0468
CVE-2012-0469	CVE-2012-0470	CVE-2012-0471	CVE-2012-0472	CVE-2012-0473	CVE-2012-0474

Figura 2 – Vulnerabilidades reportadas pelo NVD em abril de 2012

A Figura 2 mostra parte das 228 vulnerabilidades divulgadas para o ano de 2012 no mês de abril. Ao selecionar qualquer CVE, uma nova página é carregada com os detalhes da vulnerabilidade escolhida. Para este exemplo, o primeiro CVE-2011-4042 é apresentado na Figura 3.

🚩 CVE-2011-4042 Detail

Description

An unspecified ActiveX control in SVUIGrd.ocx in ARC Informatique PcVue 6.0 through 10.0, FrontVue, and PlantVue allows remote attackers to execute arbitrary code by using a crafted HTML document to obtain control of a function pointer.

Source: MITRE

Description Last Modified: 04/02/2012

Impact

CVSS v2.0 Severity and Metrics:

Base Score: 9.3 HIGH

Vector: (AV:N/AC:M/Au:N/C:C/I:C/A:C) (V2 legend)

Impact Subscore: 10.0

Exploitability Subscore: 8.6

QUICK INFO

CVE Dictionary Entry:

CVE-2011-4042

NVD Published Date:

04/02/2012

NVD Last Modified:

04/03/2012

Figura 3 – Vulnerabilidade CVE-2011-4042

A Figura 3 mostra alguns detalhes relacionados a vulnerabilidade CVE-2011-4042. Descrição (*Description na Figura*), a tabela de impacto (*Impact na Figura*) e a data de publicação da vulnerabilidade (*NVD Published Date na Figura*), são alguns dos campos apresentados em arquivos XML sendo identificados no cabeçalho principal do arquivo. Esse cabeçalho serve para facilitar na busca dos dados para a coleta.

2.3 Trabalhos Relacionados

Esta seção tem como objetivo descrever trabalhos relacionados ao tema principal desse trabalho: análise de descoberta de vulnerabilidade. Os trabalhos propostos por Massacci e Nguyen (2014), Alhazmi, Malaiya e Ray (2007), Alhazmi e Malaiya (2008) e Joh e Malaiya (2016) serão apresentados a seguir. Ao final, será discutido brevemente a relação destes trabalhos com a proposta deste trabalho.

Um modelo de descoberta de vulnerabilidade pode ser usado para avaliar e prever tendências futuras acerca de descoberta de vulnerabilidade de software. Ou seja, os modelos usam dados antigos de vulnerabilidades divulgadas (em geral fornecidos por bases de dados que catalogam vulnerabilidades de segurança, como a NVD) para tentar prever a ocorrência de novas vulnerabilidades no futuro.

Dentre os diversos modelos estatísticos usados para analisar as características do processo de descoberta de vulnerabilidade, têm-se:

- Modelo Logístico Alhazmi-Malaiya (AML) e Modelo Linear (LN) proposto por Alhazmi e Malaiya (ALHAZMI; MALAIYA, 2006);
- Modelo Quadrático (RQ) e Modelo Exponencial (RE) proposto por (RESCORLA, 2005);
- Modelo Weibull (JW) proposto por Kim et al. (JOH; JINYOO; MALAIYA, 2008);

- Modelo Distribuição Normal Dobrada (YF) proposto por Younis et al. (YOUNIS; JOH; MALAIYA, 2011);
- Modelo Termodinâmico (AT) proposto por (ANDERSON, 2002);
- Modelo Regressão de Poisson (LP) proposto por Musa e Okumoto (MUSA; OKUMOTO, 1984).

Alguns modelos de crescimento de confiabilidade de software (RGB) também usam VDMs para estudar o processo de detecção de defeitos (OZMENT; SCHECHTER, 2006), (MUSA, 2004) em sistemas computacionais.

Massacci e Nguyen (2014) apresentam uma metodologia empírica que avalia sistematicamente o desempenho de VDMs em duas dimensões (qualidade e previsibilidade) e abordam todas as questões identificadas da metodologia tradicional, isto é, questões que distorciam estudos prévios no campo.

Para ilustrar a metodologia, os autores fizeram um experimento para avaliar oito VDMs (AML, AT, LN, JW, LP, RE, RQ e YF) em 30 lançamentos principais de quatro navegadores da Web: IE, Firefox, Chrome e Safari. Classificaram a idade da versão de um navegador em três períodos diferentes: juventude (dentro de 6 - 12 meses desde a data de lançamento), idade média (12 a 36 meses desde a data de liberação) e idade avançada (36 meses).

Os autores descobriram que se uma versão é relativamente nova, então o melhor modelo a ser usado é um modelo linear para estimar as vulnerabilidades nos próximos 3 - 6 meses. Para navegadores de meia-idade e idade avançada, o melhor é usar um modelo logístico em forma de "s". Em comparação com a metodologia tradicional, isso teria sido impossível.

Alhazmi, Malaiya e Ray (2007) investigam em seu artigo se é possível prever o número de vulnerabilidades que podem estar potencialmente presentes em um sistema de software, mas que ainda não foram encontradas. Os autores usaram vários sistemas operacionais como representantes de sistemas de software complexos, tais como Windows 95, Windows 98, Windows XP, Windows NT4, Windows 2000, Windows 2003, Red Hat 6.2 e Red Hat 7.1.

Além dos dados sobre vulnerabilidades descobertas nesses sistemas, os autores examinaram os resultados para determinar se a densidade de vulnerabilidades em um programa é uma medida útil tanto para os usuários quanto para os desenvolvedores. A densidade de vulnerabilidades pode ser usada pelos usuários para avaliar o risco ao considerar a compra de um novo sistema de software da mesma empresa e os desenvolvedores podem usar essa métrica para estimar quantidades de vulnerabilidades que provavelmente serão descobertas dentro de algum período de tempo futuro para decidir se adia a aplicação de patches examinando os riscos de modo a evitar a desestabilização dos sistemas de software.

Por fim, os autores consideraram a taxa de descoberta de vulnerabilidades para verificar se os modelos de descoberta de vulnerabilidades podem ser desenvolvidos para projetar tendên-

cias futuras. Os resultados revelaram que é possível modelar a descoberta de vulnerabilidades usando um modelo logístico (AML) que às vezes pode ser aproximado por um modelo linear.

[Alhazmi e Malaiya \(2008\)](#) descrevem a aplicabilidade e a significância dos parâmetros de seis modelos de descoberta de vulnerabilidade (AT, AML, LM, LP, RQ, RE) para quatro sistemas operacionais (Windows XP, Windows 95, Red Hat Linux 6.2 e Red Hat Fedora). Os modelos de descoberta de vulnerabilidade foram examinados usando o critério de informação de Akaike (AIC) e teste qui-quadrado. A avaliação descobriu que o modelo AML é geralmente melhor a longo prazo, com melhor desempenho para sistemas como o Windows 95, Red Hat Linux 6.2 e Red Hat Fedora.

Além do trabalho publicado em 2009, os autores Joh e Malayia publicaram um outro trabalho, com uma metodologia semelhante, em 2016 ([JOH; MALAIYA, 2016](#)). Este último trabalho condensa os dados entre 1995 e 2015. Os autores examinaram vulnerabilidades divulgadas em diversos softwares (Sistemas operacionais Windows: Windows NT, Windows 95, Windows 98, Windows 2000, Windows XP e Windows 7; Sistemas operacionais diferentes do Windows: iOS, MAC OS X, Red Hat Linux Enterprise, AIX, Android e Chrome OS; e outros sistemas: Apache, IIS, Internet Explorer, Firefox, Safari e Java (JRE)) para investigar possíveis variações anuais nos processos de descoberta de vulnerabilidades. Também examinaram a periodicidade semanal na distribuição de atualizações de segurança (patches) e exploração das vulnerabilidades.

Para todos os grupos de software examinados pelos autores, uma taxa de descoberta de vulnerabilidades mais alta é encontrada em determinados meses. Nos produtos Microsoft, uma incidência mais alta durante os períodos de meio ano é observada. Além disso, o comportamento periódico de 7 dias foi observado nos dados de varredura de vulnerabilidades; maior atividade durante a semana do que nos fins de semana foi confirmada. Especificamente, os valores da atividade de vulnerabilidade correspondentes a terça-feira tenderam a ser maiores do que os outros dias da semana. Os resultados mostraram que a periodicidade precisa ser considerada para a alocação ótima de recursos e para a avaliação dos riscos de segurança.

Apesar dos trabalhos [Massacci e Nguyen \(2014\)](#), [Alhazmi e Malaiya \(2006\)](#), [Alhazmi e Malaiya \(2008\)](#), [Joh e Malaiya \(2009\)](#) e [Joh e Malaiya \(2016\)](#) conduzirem estudos para modelar o processo de descoberta de vulnerabilidade de software, somente os dois últimos detalham a questão da sazonalidade. O objetivo do presente trabalho é comparar os resultados da análise da sazonalidade em dois diferentes momentos: entre 1995-2007 e entre 2008-2017. A principal motivação para isso é avaliar a evolução do comportamento de questões de segurança ao longo dos anos.

3 Metodologia

3.1 Trabalho de Joh e Malaiya (2009)

Os autores [Joh e Malaiya \(2009\)](#) propuseram um estudo sobre a variação sazonal no processo de descoberta de vulnerabilidade de segurança em softwares. O objetivo da pesquisa consistia em encontrar um padrão sazonal entre as bases de dados disponíveis dos softwares estudados e descobrir em qual época do ano, a taxa de vulnerabilidade era maior e detectar possíveis causas desse acontecimento.

A partir da coleta dos dados, os autores fizeram uma análise para descobrir uma possível existência de sazonalidade utilizando dois métodos estatísticos. O primeiro foi um método de índice sazonal medido com o teste qui-quadrado, no qual fornece índices específicos para cada mês, e o segundo foi usado a função de autocorrelação, que fornece informações do mês correlacionado. Dessa forma, os autores puderam investigar o comportamento de cada um dos sistemas.

Os autores dividiram os sistemas operacionais em três categorias: Windows, não Windows e Web. Os sistemas operacionais Windows foram Windows NT, Windows XP, Windows 2000 e Windows server 2003, os sistemas operacionais não Windows foram SUN Solaris, Red Hat Linux, HP-UX e MAC OS X e as aplicações Web foram IIS, IE, Apache e Firefox. A base de dados de vulnerabilidades utilizada pelos autores foram do ano de 1995 até 2007.

Os resultados encontrados pelos autores sugeriram que para a categoria Windows, os meses de junho e dezembro tiveram uma alta taxa de descoberta de vulnerabilidade, enquanto que os meses de fevereiro, março, abril e setembro tiveram baixa taxa de descoberta de vulnerabilidade abaixo da média. Ambas as categorias não Windows e Web mostraram que o mês de dezembro teve uma alta taxa de descoberta de vulnerabilidade. De acordo com os autores, essa sazonalidade no processo de descoberta de vulnerabilidade pode estar associada ao início de semestre nas escolas e os períodos festivos como Natal e Ano novo, pois são quando as pessoas compram novos computadores com algum dos sistemas operacionais descrito acima.

3.2 Esta proposta

Dado que alguns dos sistemas operacionais que foram utilizados pelos autores Hyun-Chul Joh e Yashwant K. Malaiya foram descontinuados, será feita uma coleta de dados referente aos sistemas operacionais Windows mais recentes como: Windows Vista, Windows 7, Windows 8, Windows 8.1 e Windows 10. Para os sistemas operacionais diferentes do Windows os escolhidos foram Solaris, Red Hat Linux, Ubuntu Linux e Mac OS X e para as aplicações Web

foram escolhidos IE, Firefox, Chrome, IIS e Apache. Além disso, a base de dados utilizada corresponde aos anos de 2008 até 2017.

A partir da coleta desses dados, a mesma técnica utilizada pelos autores será replicada neste trabalho, ou seja, os métodos de índice sazonal, teste qui-quadrado e a função de autocorrelação serão utilizados para encontrar padrões de repetição e suas periodicidades. Através desses métodos estatísticos, será possível inferir o comportamento descrito pelas vulnerabilidades por mês e ano.

Para melhor visualização dos dados calculados pelo índice sazonal e função de autocorrelação, serão construídos gráficos de série temporal para cada grupo de sistema e gráficos individuais para cada sistema, respectivamente. Para aqueles sistemas que em ambos os gráficos apresentarem possível sazonalidade será construído outro gráfico chamado box-plot para reforçar mais a análise de detecção de padrões de sazonalidade.

O método empregado para avaliar a sazonalidade do processo de descoberta de vulnerabilidades envolve os seguintes passos:

1. Coleta dos dados de vulnerabilidades para cada sistema desde o ano de 2008 até o ano de 2017;
2. Cálculo do índice de sazonalidade para cada sistema;
3. Construção de gráficos de série temporal para cada grupo de sistema de acordo com o cálculo do índice de sazonalidade;
4. Aplicação da função de autocorrelação para cada um dos meses para todos os sistemas;
5. Análise individual da função de autocorrelação para detecção de possível sazonalidade para cada sistema;
6. Construção de gráficos boxplot somente para os sistemas que apresentarem possibilidade de sazonalidade;
7. Fazer uma previsão para cada sistema identificado com possibilidade de sazonalidade usando componentes de sazonalidade.

3.3 Coleta dos dados

Os arquivos XML são disponibilizados pelo NVD em arquivos *zip*. Cada ano possui o seu próprio arquivo com as devidas informações. Portanto, ao extrair o arquivo XML, o próximo passo é importá-lo para uma tabela Excel para melhor visualização dos dados sobre as vulnerabilidades. O Excel foi escolhido para a importação dos dados devido às suas ferramentas de filtragem.

Uma amostra de como os dados são importados do arquivo XML é mostrada na Figura 4. No arquivo, encontra-se exatamente 35 campos de cabeçalho, porém somente 3 são importantes para a coleta dos dados. Os principais são os campos *published*, *name3* e *vendor*, que significam, respectivamente, a data de publicação da vulnerabilidade, o nome do sistema operacional em que foi encontrado vulnerabilidade e o nome da empresa responsável pelo sistema.

O campo *name3* apresenta todos os tipos de software conhecidos, por isso o segundo passo é filtrar esse campo de acordo com o tipo de software desejado, nesse caso o Solaris. De forma automática, todos os campos são atualizados com as respectivas informações daquele sistema. Depois de escolhido o sistema, a coluna *published* permite filtrar as vulnerabilidades por meses de divulgação para facilitar a contabilização das mesmas.

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

Figura 4 – Parte de um arquivo XML para o sistema Solaris

Outras informações como o nome da vulnerabilidade (*name*) reportada pelo CVE e a rigorosidade (*severity*) da vulnerabilidade não tem relevância para este trabalho, somente a data de publicação das vulnerabilidades.

Para o presente trabalho, foram usados os arquivos XML do ano de 2008 até 2017. Para alguns dos softwares escolhidos, os arquivos XML não apresentaram toda a informação necessária dos dez anos analisados, porém apresentaram informações de acordo com seu lançamento ou quando vulnerabilidades não foram encontradas.

A seguir, um breve resumo dos dados coletados. Para o Windows 7 não foram encontradas vulnerabilidades no ano de 2008. Para o Windows 8 não foram encontrados vulnerabilidades no ano de 2008 até 2011. Para o Windows 8.1 não foram encontrados vulnerabilidades no ano de 2008 até 2012. Para o Windows 10 não foram encontrados vulnerabilidades no ano de 2008 até 2014. Para o Solaris foram encontrados vulnerabilidades, exceto no ano de 2012. Para o IIS foram encontrados vulnerabilidades, exceto no ano de 2011, 2014 e 2016. Para o IE foram encontrados vulnerabilidades, exceto no ano de 2011. Para o Apache não foram encontrados vulnerabilidades no ano de 2008, 2011, 2015, 2016 e 2017. Para o Firefox não foram encontrados vulnerabilidades para o ano de 2017.

4 Resultados

Neste capítulo serão mostrados os resultados da análise de sazonalidade e também será feita uma comparação com os resultados obtidos no ano de 2009 pelos autores Joh e Malaiya.

4.1 Análise dos dados

O índice de sazonalidade é uma medida muito utilizada para avaliar tendências sazonais e pode indicar o quanto a média de um período específico tende a estar acima (ou abaixo) do valor esperado (ARSHAM, 1994). Os valores mensais do índice sazonal são dados por:

$$s_i = \frac{d_i}{d} \quad (4.1)$$

onde, s_i é o índice sazonal para o i -ésimo mês, d_i é a média aritmética do i -ésimo mês e d é a média aritmética entre cada um dos d_i . Assim, se um índice sazonal apresentar valor 1 para um determinado mês isso indica que o valor esperado desse mês é 1/12 da média geral. Caso esse valor for maior que 1 isso indica que o valor esperado para esse mês é maior que a média geral e se for menor que 1 isso indica o contrário. Em outras palavras, para os resultados deste trabalho, se o índice sazonal for maior que 1 significa que há uma tendência de sazonalidade para o determinado mês.

Para determinar se os índices sazonais são estatisticamente significantes, pode ser aplicado o teste qui-quadrado (χ^2) para a hipóteses nula H_0 : *nenhuma sazonalidade presente*. Para ser estatisticamente significante, o valor calculado para χ_s^2 estático deve ser maior que o valor calculado para o χ_c^2 crítico com um p-valor suficientemente pequeno.

Outro método a ser aplicado para determinar se existe sazonalidade será a função de autocorrelação. A análise da autocorrelação fornece informações sobre a relação entre uma variável X medida no tempo t , com ela mesma medida em um tempo posterior $t + k$, onde k representa o intervalo de observação (*lags*). Seja $z_b, z_{b+1}, \dots, z_n$ os valores de uma série temporal, o coeficiente de autocorrelação r_k em um determinado intervalo de observação k é dado por (BOWERMAN; O'CONNELL, 1987):

$$r_k = \frac{\sum_{t=b}^{n-k} (z_t - \bar{z})(z_{t+k} - \bar{z})}{\sum_{t=b}^n (z_t - \bar{z})^2} \quad (4.2)$$

onde $\bar{z} = \frac{\sum_{t=b}^n z_t}{(n-b+1)}$ representa a média das observações. Valores de coeficiente de autocorrelação próximos de zero indicam ausência de sazonalidade em tal intervalo de observação,

enquanto que valores próximos de 1 mostram uma relação significativa associada a tal intervalo de observação.

O valor de r_k , aliado a análise do correlograma (gráfico com os valores de r_k dispostos em intervalos de defasagem) são usados para estabelecer critérios sobre uma série temporal. Três possíveis situações podem ocorrer (MIANI, 2013):

1. A série temporal é considerada aleatória ou estacionária se a maioria dos coeficientes de autocorrelação estão entre os intervalos de confiança e nenhum padrão foi detectado no correlograma;
2. A série temporal é considerada não estacionária “moderada” quando os valores dos coeficientes de autocorrelação diminuem lentamente conforme k aumenta, caracterizando um comportamento de tendência;
3. A série temporal é considerada não estacionária “forte” quando os valores dos coeficientes de correlação diminuem lentamente conforme k aumenta, mas de acordo com uma periodicidade, caracterizando um padrão sazonal.

A principal diferença entre as situações 2 e 3 é que uma série temporal não estacionária “moderada” não costuma exibir um comportamento periódico. Nesses tipos de dados, é comum encontrar componentes r_k isolados relacionados a uma tendência de crescimento (ou diminuição) ou ainda alguns tipos de padrões relacionados a sazonalidade.

Os resultados das análises para os grupos de sistemas de software usando o índice de sazonalidade, qui-quadrado e autocorrelação serão mostradas e explicadas nas próximas seções.

Com base em tais métodos, a seguinte sequência de passos será utilizada para identificar padrões de sazonalidade nas séries de dados:

1. Analisar a série temporal dos valores de vulnerabilidades descobertas por ano - é possível visualizar meses com um número maior ou menor de vulnerabilidades em todos os anos?
2. Calcular e analisar o índice de sazonalidade - identificar os meses em que o valor do índice é maior ou menor do que 1;
3. Calcular e analisar os coeficientes de autocorrelação e o correlograma - excluir os softwares em que a série é considerada como estacionária (ou seja, índices de sazonalidade maiores ou menores do que 1 encontrados anteriormente são potenciais outliers e não estão associados a padrões sazonais);
4. Investigar o box-plot das vulnerabilidades descobertas por ano e correlacionar com o índice de sazonalidade (a média de um determinado mês realmente é maior ou foi produto de uma outlier?)

4.1.1 Sistema Operacional Windows

Nesse tópico, serão examinados os sistemas operacionais Windows. A Figura 5 mostra a série temporal referente ao número de vulnerabilidades encontradas ao longo dos dez anos. Para esses sistemas operacionais o número total de vulnerabilidades foi de 1261 para Windows Vista (a), 1637 para o Windows 7 (b), 471 para Windows 8 (c), 759 para Windows 8.1 (d) e 1466 para Windows 10 (e).

As séries temporais para os sistemas Windows não fornecem nenhum padrão visível. É possível notar uma alta dispersão nos dados como uma alta concentração de vulnerabilidades em alguns anos - 2010 e 2011 para o Windows Vista e 2015 para o Windows 8, por exemplo - e baixa concentração de vulnerabilidades em outros - 2014 para o Windows Vista e a maioria da série para o Windows 7.

Com o auxílio do índice sazonal apresentado na Tabela 1 e na Figura 6 é possível notar que alguns meses possuem uma probabilidade maior de divulgação de vulnerabilidades do que os outros. Todos os sistemas apresentam baixo índice sazonal em janeiro e dezembro, e março apresenta alto índice sazonal em todos. Contudo, a fim de verificar se tais índices podem ser associados a padrões de sazonalidade testes adicionais devem ser feitos, como por exemplo, o de autocorrelação.

O correlograma e os coeficientes de autocorrelação apresentados na Figura 7 e na Tabela 2 mostram que o Windows 7 exibe um padrão estacionário (ausência de sazonalidade), diferentemente dos outros sistemas em que diversos coeficientes de autocorrelação são maiores do que 1 e estão fora do intervalo de confiança. O próximo passo da análise envolve a construção dos box-plots para os quatro sistemas que exibiram comportamento não-estacionário.

Com o auxílio dos índices de sazonalidade apresentado na Tabela 1 e do box-plot apresentado na Figura 8 é possível notar que o Windows Vista exibe o seguinte comportamento: poucas vulnerabilidades divulgadas em janeiro e um aumento de vulnerabilidade entre fevereiro e abril. Já o Windows 8 apresenta um decréscimo de vulnerabilidades em janeiro e abril mas aumento no segundo semestre (agosto-setembro). O Windows 8.1 apresenta poucas vulnerabilidades em janeiro e dezembro mas um aumento em junho. E o Windows 10 apresenta poucas vulnerabilidades em janeiro e dezembro mas um aumento junho.

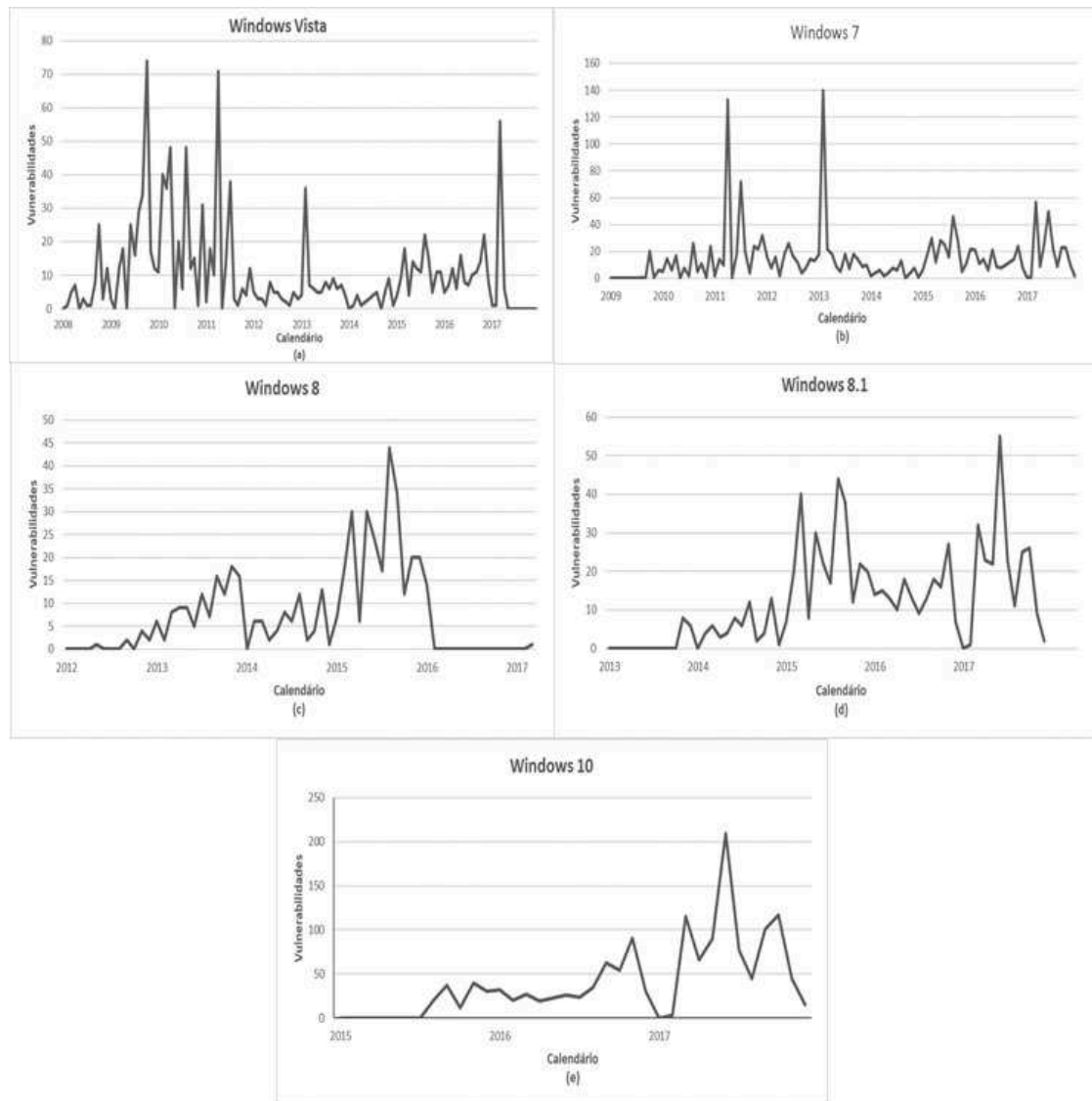


Figura 5 – Vulnerabilidades acumulada por ano - Windows OSES

Tabela 1 – Índices Sazonais de Descoberta de Vulnerabilidade - Windows OSES

	Windows Vista	Windows 7	Windows 8	Windows 8.1	Windows 10
Janeiro	0,3331	0,5205	0,6879	0,3320	0,2619
Fevereiro	1,1039	1,5467	0,6624	0,6324	0,1883
Março	1,5511	1,1802	1,1465	1,4387	1,1623
Abril	1,5987	1,4588	0,4331	0,6957	0,6958
Maio	0,4282	0,7770	1,1210	1,1700	0,9250
Junho	0,9040	1,0922	0,9682	1,5494	1,9236
Julho	0,9136	1,1802	0,8917	0,8696	0,8267
Agosto	1,2086	1,0556	1,6051	1,2648	0,8186
Setembro	0,8755	0,6891	1,3758	1,3123	1,6453
Outubro	1,4370	0,9016	0,7134	0,9170	1,4980
Novembro	0,7518	0,7330	1,4013	1,2490	1,4407
Dezembro	0,8945	0,8650	0,9936	0,5692	0,6139
χ^2_c	19,6751	19,6751	19,6751	19,6751	19,6751
χ^2_s	187,4800	144,8000	52,2870	103,5000	397,2600
p-value	2,20E-16	2,20E-16	2,42E-07	2,20E-16	2,20E-16

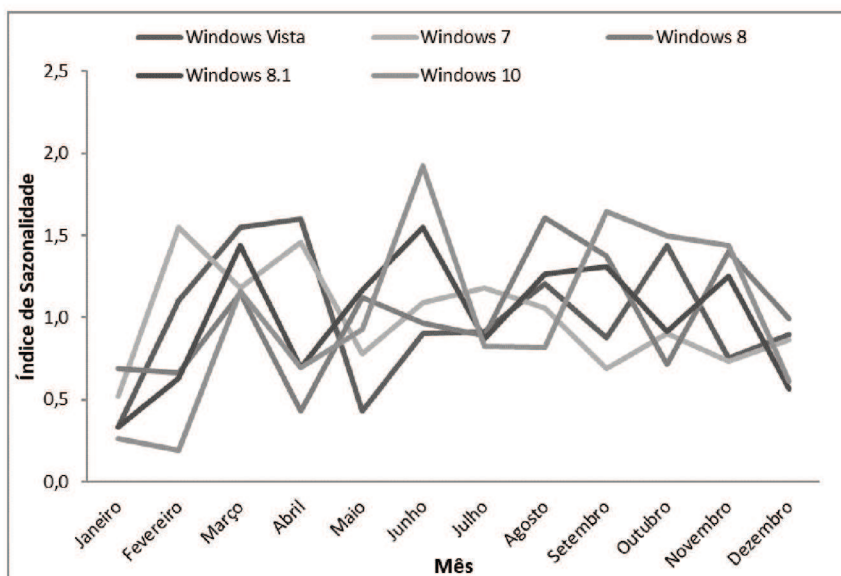


Figura 6 – Índices Sazonais - Windows OSes

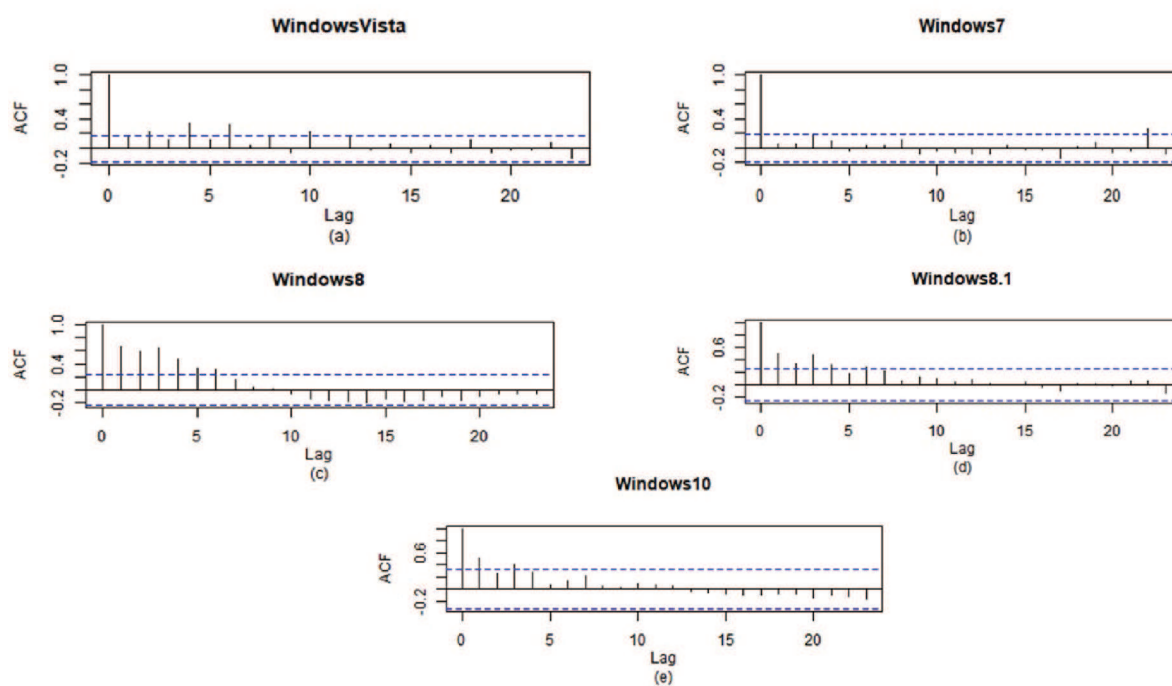


Figura 7 – Função de autocorrelação - Windows OSes

Tabela 2 – Valores de ACF individuais para a Figura 7

Windows Vista - 95% intervalo de confiança: (-0.1770343,0.1770343)						
1.0⁰	0.180¹	0.230²	0.107 ³	0.349⁴	0.119 ⁵	0.315⁶
0.030 ⁷	0.174 ⁸	-0.055 ⁹	0.219¹⁰	0.001 ¹¹	0.180¹²	-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% intervalo de confiança: (-0.1863929,0.1863929)						
1.0⁰	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	0.266²²	-0.088 ²³				
Windows 8 - 95% intervalo de confiança: (-0.2269618,0.2269618)						
1.0⁰	0.659¹	0.585²	0.642³	0.468⁴	0.330⁵	0.303⁶
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% intervalo de confiança: (-0.247765,0.247765)						
1.0⁰	0.497¹	0.332²	0.474³	0.314⁴	0.178 ⁵	0.275⁶
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% intervalo de confiança: (-0.3155171,0.3155171)						
1.0⁰	0.503¹	0.259 ²	0.406³	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				

Negrito representa os valores fora do intervalo de confiança e ^{Subscrito} representa as lags.

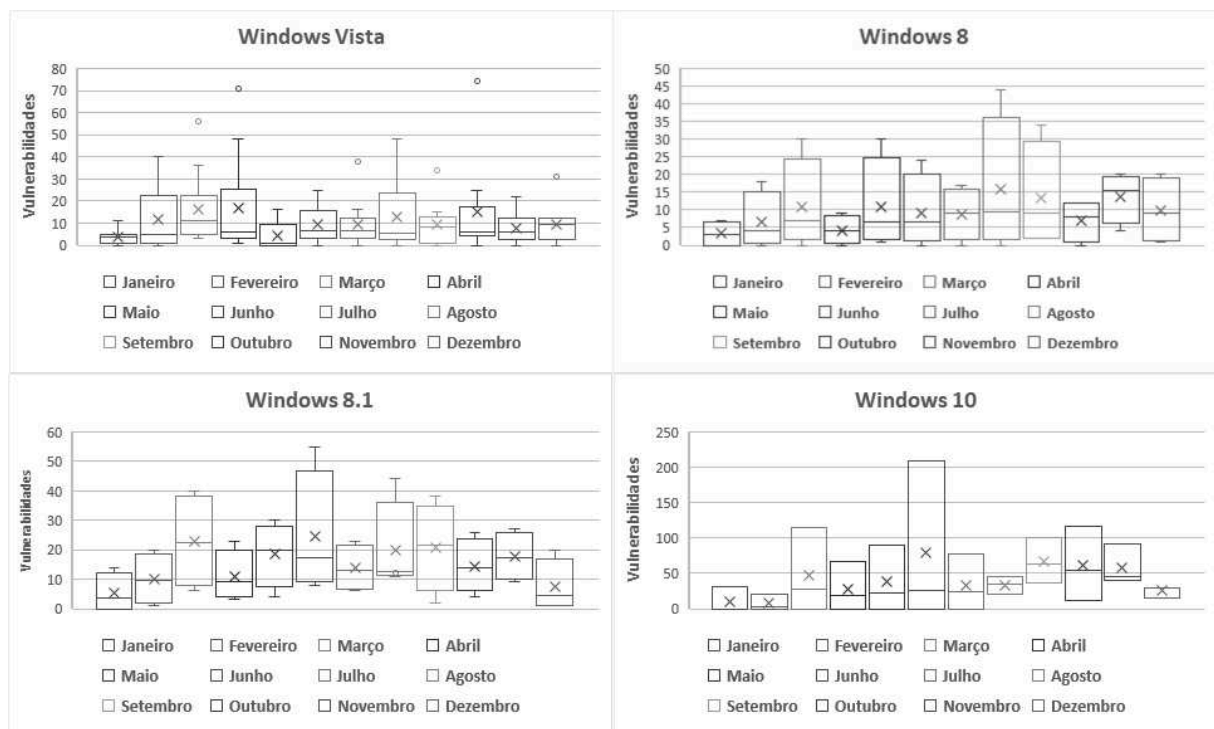


Figura 8 – Box-plot para os sistemas Windows Vista, Windows 8, Windows 8.1 e Windows 10

4.1.2 Sistemas Operacionais diferentes do Windows

Nesse tópico, serão examinados os sistemas operacionais diferentes do Windows. A Figura 9 apresenta o número de vulnerabilidades encontradas ao longo dos dez anos. Para esses sistemas, o número total de vulnerabilidades foram 756 para Solaris (a), 418 para Red Hat (b), 2166 para Ubuntu (c) e 9176 para Mac OS X (d).

As séries temporais para os sistemas diferentes do Windows não fornecem nenhum padrão visível. É possível notar uma alta dispersão nos dados como uma alta concentração de vulnerabilidades em alguns anos - 2008, 2009 e 2010 para o Solaris e 2016 para o Red Hat e Ubuntu, por exemplo - e baixa concentração de vulnerabilidades em outros - 2012 para o Solaris.

Com o auxílio do índice sazonal apresentado na Tabela 3 e na Figura 10 é possível notar que alguns meses possuem uma probabilidade maior de divulgação de vulnerabilidades do que os outros. Todos os sistemas apresentam baixo índice sazonal em fevereiro e março, mas nenhum apresenta alto índice sazonal em todos. Contudo, a fim de verificar se tais índices podem ser associados a padrões de sazonalidade, testes adicionais devem ser feitos, como por exemplo, o de autocorrelação.

O correlograma e os coeficientes de autocorrelação apresentados na Figura 11 e na Tabela 4 mostram que o Mac OS X exibe um padrão estacionário (ausência de sazonalidade), diferentemente dos outros sistemas em que diversos coeficientes de autocorrelação são maiores do que 1 e estão fora do intervalo de confiança. O próximo passo da análise envolve a construção dos box-plots para os três sistemas que exibiram comportamento não-estacionário.

Com o auxílio dos índices de sazonalidade apresentado na Tabela 3 e do box-plot apresentado na Figura 12 é possível notar que o Solaris exibe o seguinte comportamento: um aumento de vulnerabilidades divulgadas em janeiro e agosto e poucas vulnerabilidades divulgadas em fevereiro. Já o Red Hat apresenta um aumento de vulnerabilidades em junho e outubro e baixas vulnerabilidades divulgadas para o restante dos meses. O Ubuntu apresenta poucas vulnerabilidades em setembro, mas um aumento em abril e junho.

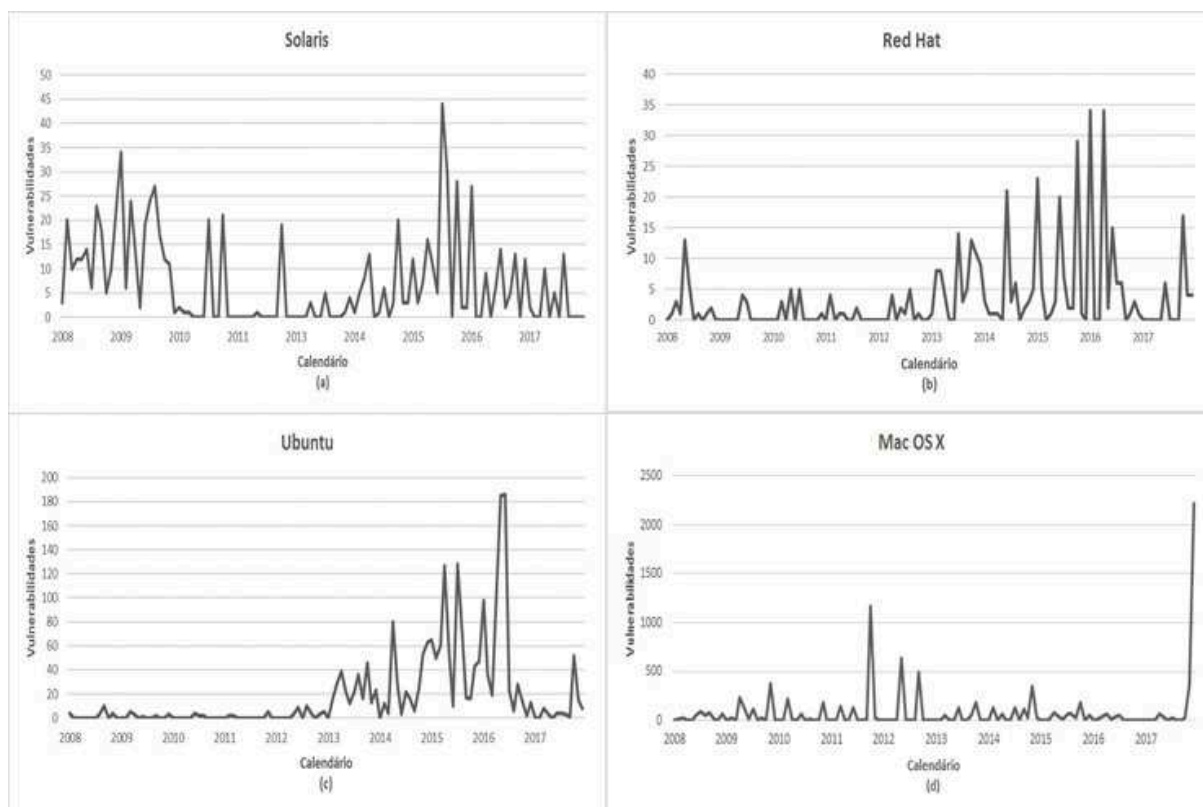


Figura 9 – Vulnerabilidades acumulada por ano - Não Windows OSes

Tabela 3 – Índices Sazonais de Descoberta de Vulnerabilidade - Não Windows OSes

	Solaris	RH Linux	Ubuntu	MAC OS X
Janeiro	1,2627	1,7512	1,0193	0,0183
Fevereiro	0,5456	0,5455	0,6372	0,2380
Março	0,7794	0,4306	0,6921	0,6578
Abril	1,2003	1,3206	1,9226	0,6591
Maio	0,4053	0,6890	1,6095	1,1155
Junho	0,7794	2,1244	1,2579	0,5885
Julho	1,8551	1,1196	1,1041	0,6212
Agosto	1,4809	0,7177	0,8075	0,1622
Setembro	0,7448	0,2010	0,4614	1,0488
Outubro	1,8395	1,8373	0,8404	2,0859
Novembro	0,4209	0,6890	0,7580	1,7014
Dezembro	0,6859	0,5742	0,8899	3,1033
X_c^2	19,6751	19,6751	19,6751	19,6751
X_s^2	192,1000	148,7600	359,0600	6808,0000
p-value	2,20E-16	2,20E-16	2,20E-16	2,20E-16

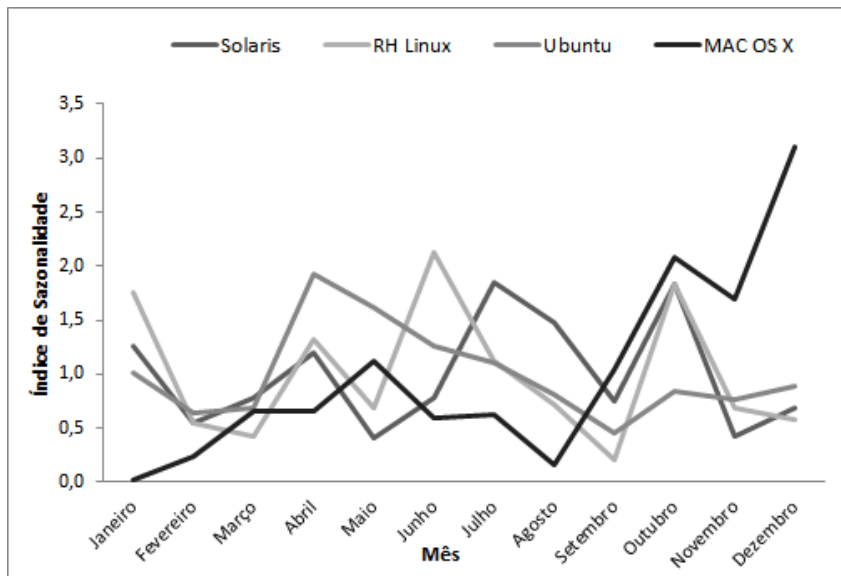


Figura 10 – Índice Sazonal - Não Windows OSes

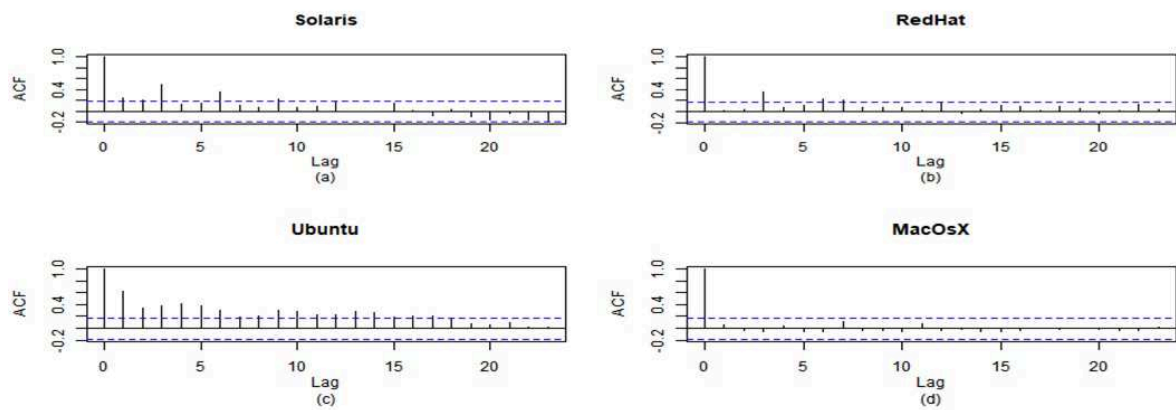


Figura 11 – Função de autocorrelação - Não Windows OSes

Tabela 4 – Valores de ACF individuais para a Figura 11 - Não Windows OSes

Solaris - 95% intervalo de confiança: (-0.1863929,0.1863929)						
1.0⁰	0.245¹	0.203²	0.480³	0.120 ⁴	0.152 ⁵	0.348⁶
0.116 ⁷	0.074 ⁸	0.227⁹	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% intervalo de confiança: (-0.1770343,0.1770343)						
1.0⁰	0.024 ¹	0.037 ²	0.364³	0.076 ⁴	0.119 ⁵	0.230⁶
0.205⁷	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% intervalo de confiança: (-0.1770343,0.1770343)						
1.0⁰	0.623¹	0.337²	0.370³	0.424⁴	0.369⁵	0.306⁶
0.196⁷	0.207⁸	0.299⁹	0.284¹⁰	0.234¹¹	0.233¹²	0.277¹³
0.264¹⁴	0.189¹⁵	0.202¹⁶	0.216¹⁷	0.148 ¹⁸	0.076 ¹⁹	0.062 ²⁰
0.097 ²¹	0.022 ²²	0.024 ²³				
Mac OS X - 95% intervalo de confiança: (-0.1770343,0.1770343)						
1.0⁰	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				

Negrito representa os valores fora do intervalo de confiança e *Sobrescrito* representa as lags.

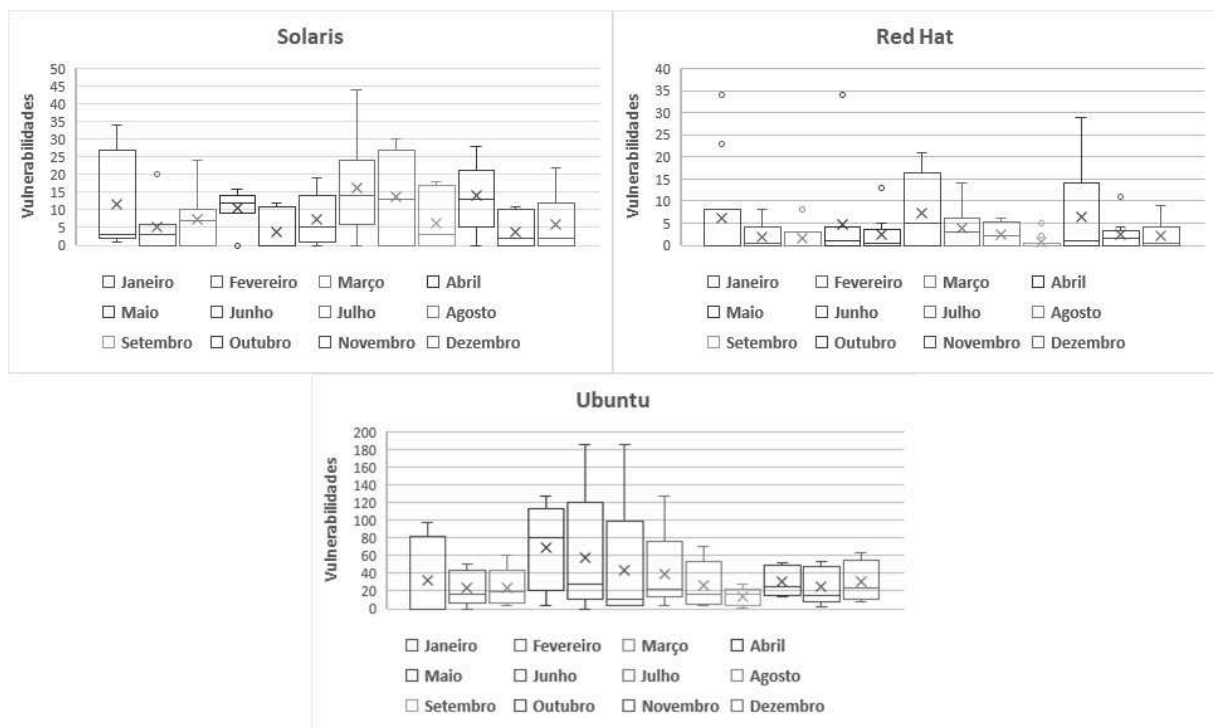


Figura 12 – Box-plot para os sistemas Solaris, Red Hat e Ubuntu - Não Windows OSes

4.1.3 Servidores e Navegadores Web

Nesse tópico analisaremos as aplicações Web. A Figura 13 apresenta o número de vulnerabilidades encontradas ao longo dos dez anos. Para esses sistemas, o número de vulnerabi-

lidades foram 35 para IIS (a), 2528 para IE (b), 60180 para Firefox (c) e 456021 para Chrome (d).

Ao coletar os dados referentes ao servidor Apache, o total acumulado de vulnerabilidades descobertas do ano de 2008 até o ano de 2017 foram de apenas 19 vulnerabilidades. Portanto, o servidor Apache não foi considerado nos cálculos de índice sazonal e de autocorrelação.

As séries temporais para as aplicações Web não fornecem nenhum padrão visível. É possível notar uma alta dispersão nos dados como uma alta concentração de vulnerabilidades em alguns anos - 2014 e 2015 para o IE e 2011 para o Chrome, por exemplo - e baixa concentração de vulnerabilidades em outros - 2012 a 2017 para o IIS.

Com o auxílio do índice sazonal apresentado na Tabela 5 e na Figura 14 é possível notar que alguns meses possuem uma probabilidade maior de divulgação de vulnerabilidades do que os outros. Todos os sistemas apresentam baixo índice sazonal em janeiro, mas nenhum apresenta alto índice sazonal em todos. Contudo, a fim de verificar se tais índices podem ser associados a padrões de sazonalidade testes adicionais devem ser feitos, como por exemplo, o de autocorrelação.

O correlograma e os coeficientes de autocorrelação apresentados na Figura 15 e na Tabela 6 mostram que todos os sistemas exibem um padrão não-estacionário, onde diversos coeficientes de autocorrelação são maiores do que 1 e estão fora do intervalo de confiança. O próximo passo da análise envolve a construção dos box-plots para os quatro sistemas que exibiram comportamento não-estacionário.

Com o auxílio dos índices de sazonalidade apresentado na Tabela 5 e do box-plot apresentado na Figura 16 é possível notar que o IIS exibe o seguinte comportamento: um aumento de vulnerabilidades divulgadas em julho e setembro e baixas vulnerabilidades para os demais meses. Já o IE apresenta um aumento de vulnerabilidades em fevereiro e junho e baixa vulnerabilidade divulgadas em janeiro. O Firefox apresenta poucas vulnerabilidades em janeiro e abril mas um aumento de vulnerabilidades divulgadas em fevereiro.

A Figura 17 apresenta dois gráficos de box-plot para o Chrome. Não foi possível identificar sazonalidade ao longo do ano devido aos valores de vulnerabilidade coletados para este sistema que são muito dispersos. Ao construir os dois gráficos para esta aplicação, um para entre 2009-2013 e outro entre 2014-2016, é possível identificar picos de sazonalidade.

No primeiro gráfico os meses de abril e julho apresentam baixa vulnerabilidade divulgada mas um aumento de vulnerabilidades em março, agosto e setembro. No segundo gráfico os meses de março, maio e agosto apresentam um aumento de vulnerabilidades divulgadas e baixas vulnerabilidades divulgadas para a maioria dos outros meses.

Comparando os dois gráficos, é possível perceber que a sazonalidade presente no primeiro gráfico, mudou completamente em relação ao segundo gráfico. Ou seja, os meses de

sazonalidade são parecidos, porém o número de vulnerabilidades divulgadas diminuíram significativamente, de 70 mil para 1400 vulnerabilidades.

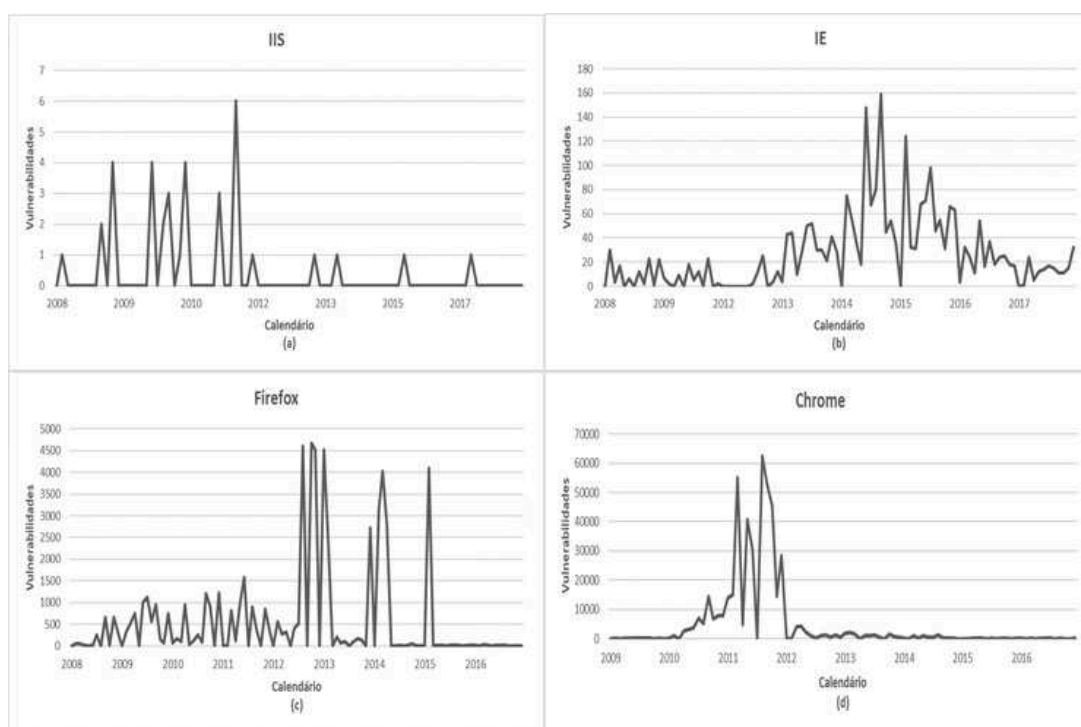


Figura 13 – Vulnerabilidades acumulada por ano - Web

Tabela 5 – Índices Sazonais de Descoberta de Vulnerabilidades - Web

	IIS	Chrome	IE	Firefox
Janeiro	0,0000	0,4195	0,0807	0,9220
Fevereiro	0,3429	0,4723	1,4573	2,1406
Março	1,0286	1,6300	0,8687	1,1687
Abril	0,0000	0,3099	0,5601	1,0351
Maio	0,0000	1,2653	0,8592	0,2154
Junho	2,4000	0,9620	1,5332	0,6526
Julho	0,0000	0,2399	1,3196	0,4407
Agosto	0,6857	1,8441	1,0680	1,2576
Setembro	3,7714	1,8143	1,4525	0,6837
Outubro	0,0000	1,4271	0,8497	1,1870
Novembro	2,0571	0,6378	0,9399	1,2241
Dezembro	1,7143	0,9777	1,0111	1,0724
X_c^2	19,6751	19,6751	19,6751	19,6751
X_s^2	49,0000	145470,0000	401,7500	13253,0000
p-value	9,46E-07	2,20E-16	2,20E-16	2,20E-16

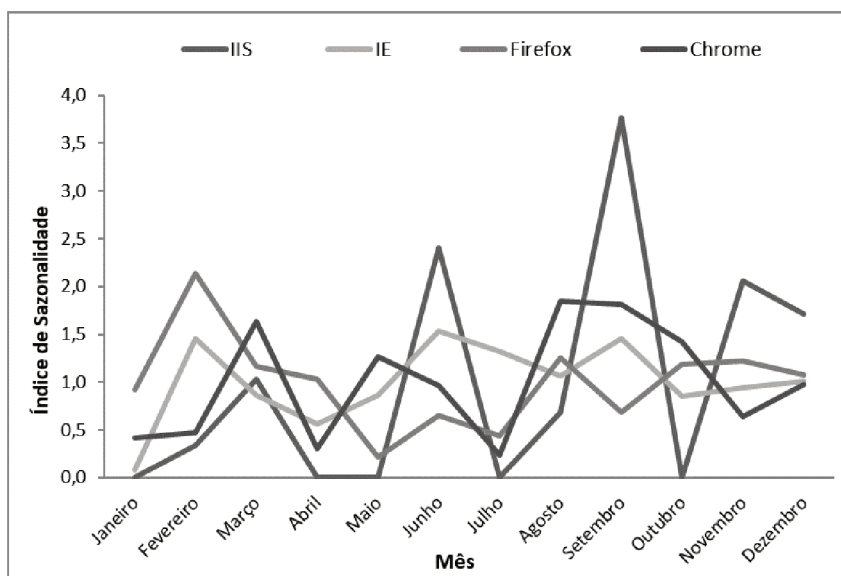


Figura 14 – Índice sazonal - Web

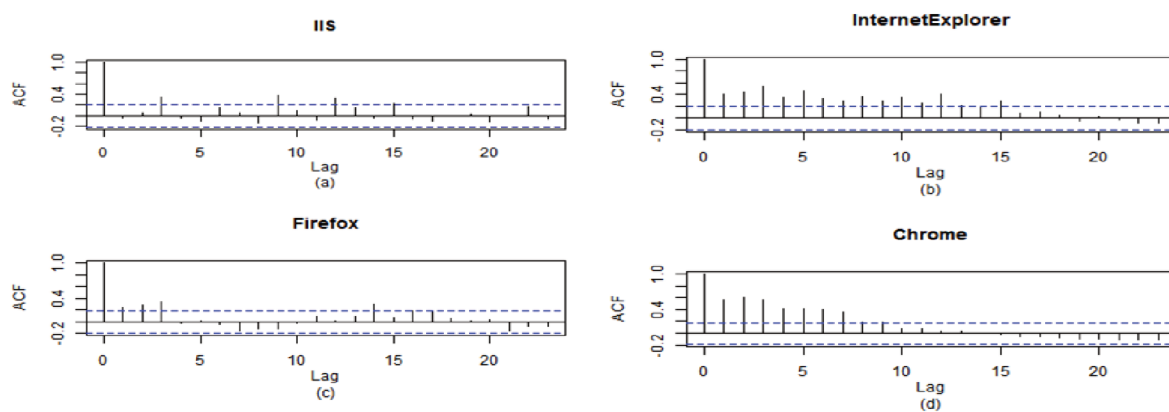


Figura 15 – Função de autocorrelação - Web

Tabela 6 – Valores de ACF individuais para a Figura 15 - Web

IIS - 95% intervalo de confiança: (-0.2106483,0.2106483)						
1.0⁰	-0.046 ¹	0.044 ²	0.345³	-0.057 ⁴	-0.108 ⁵	0.154 ⁶
0.055 ⁷	-0.152 ⁸	0.372⁹	0.091 ¹⁰	-0.080 ¹¹	0.319¹²	0.151 ¹³
-0.055 ¹⁴	0.216¹⁵	-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% intervalo de confiança: (-0.1974118,0.1974118)						
1.0⁰	0.413¹	0.449²	0.546³	0.350⁴	0.469⁵	0.320 ⁶
0.290⁷	0.359⁸	0.297⁹	0.347¹⁰	0.252¹¹	0.410¹²	0.210¹³
0.175 ¹⁴	0.296¹⁵	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% intervalo de confiança: (-0.1863923,0.1863929)						
1.0⁰	0.243¹	0.280²	0.342³	-0.021 ⁴	0.013 ⁵	-0.0360 ⁶
-0.151 ⁷	-0.112 ⁸	-0.127 ⁹	-0.031 ¹⁰	0.097 ¹¹	0.021 ¹²	0.088 ¹³
0.294¹⁴	0.060 ¹⁵	0.191¹⁶	0.182 ¹⁷	0.050 ¹⁸	0.008 ¹⁹	0.039 ²⁰
-0.166 ²¹	-0.085 ²²	-0.086 ²³				
Chrome - 95% intervalo de confiança: (-0.1770343,0.1770343)						
1.0⁰	0.564¹	0.605²	0.558³	0.409⁴	0.416⁵	0.397⁶
0.359⁷	0.191⁸	0.192⁹	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				

Negrito representa os valores fora do intervalo de confiança e *Sobrescrito* representa as lags.

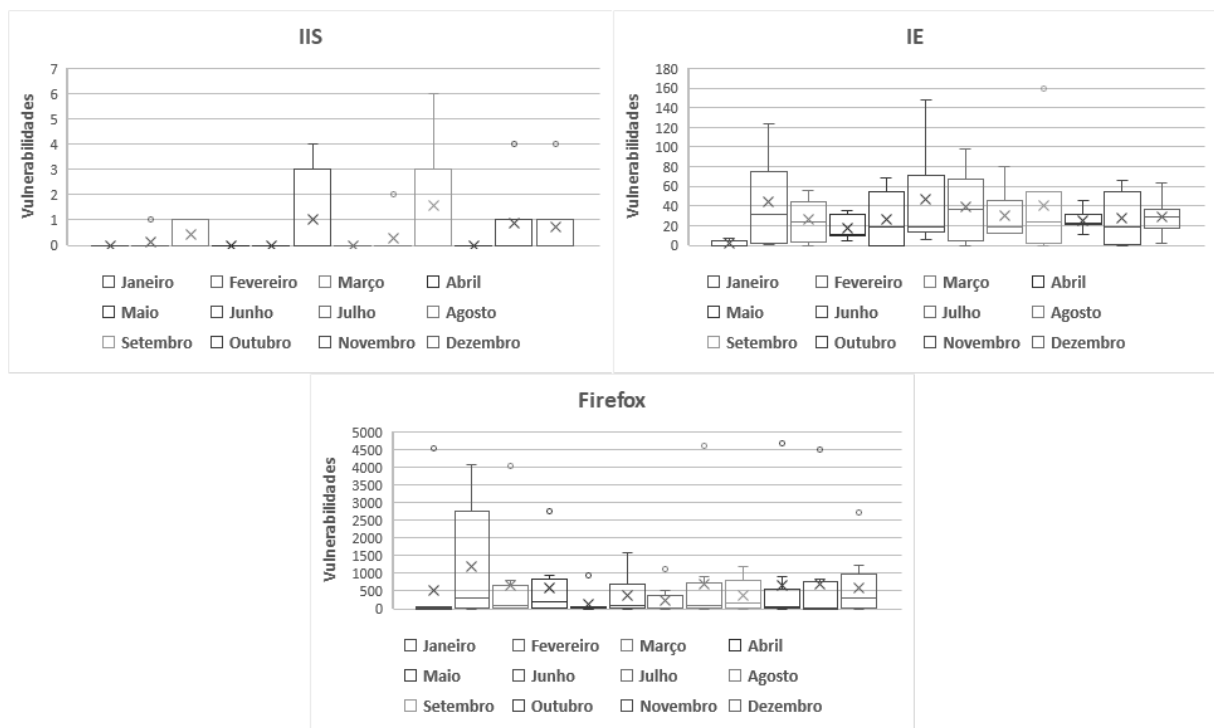


Figura 16 – Box-plot para os sistemas IIS, IE e Firefox - Web

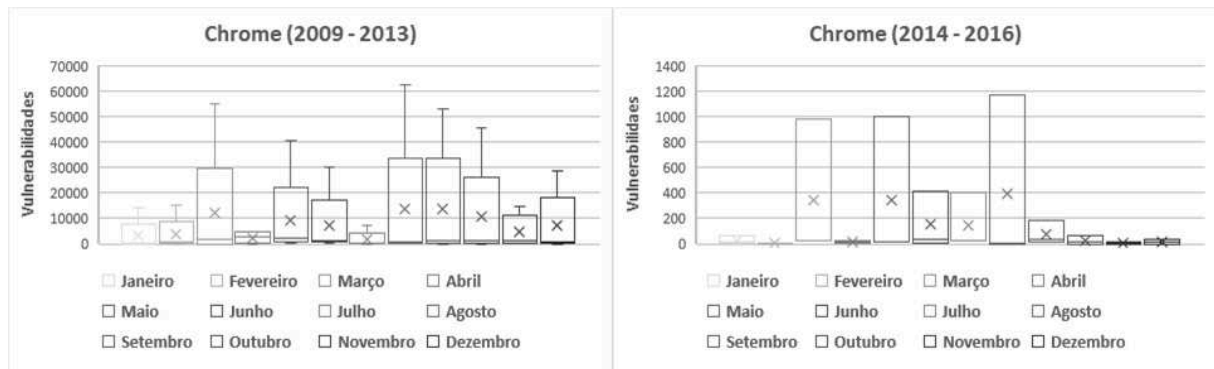


Figura 17 – Box-plot para o Chrome - Web

De todos os softwares analisados, somente os sistemas Windows 7 e Mac OS não apresentaram sazonalidade. O mês de janeiro apresentou ter menos divulgação de vulnerabilidades para os sistemas Windows e para as aplicações Web. Enquanto que para os sistemas diferentes do Windows, os meses de fevereiro e setembro apresentaram ter menos vulnerabilidades divulgadas. Em geral, para todos os sistemas, o mês de junho apresenta ter mais vulnerabilidades divulgadas.

Os sistemas Windows 8.1 e Windows 10 apresentaram um padrão de sazonalidade entre ambos no mês de junho, o mesmo vale para os sistemas Red Hat e Ubuntu. As aplicações IE e Firefox apresentaram um padrão de sazonalidade entre ambos no mês de fevereiro.

4.2 Comparação entre os trabalhos

Para os sistemas estudados, [Joh e Malaiya \(2009\)](#) descobriram que os meses de junho e dezembro, para os sistemas Windows, apresentavam uma alta taxa de descoberta de vulnerabilidade. Para os sistemas diferentes do Windows e as aplicações Web o mês de dezembro apresentava alta taxa de descoberta de vulnerabilidade. Contudo, o presente trabalho mostrou diferenças nesse sentido. Os sistemas Windows, os sistemas diferentes do Windows e as aplicações Web tiveram grande concentração de sazonalidade em junho. Isso mostra importância de realizar tais tipos de estudos usando dados mais recentes. Ou seja, comportamentos associados a questões de segurança da informação, em especial o estudo nesse trabalho, tendem a mudar ao longo do tempo.

A Tabela 7 ilustra as principais semelhanças e diferenças entre os trabalhos.

Tabela 7 – Diferenças e Semelhanças entre os trabalhos

Sistemas	Comparativo	Trabalho de Joh e Malaiya	Trabalho Atual
Sistemas Windows	Meses com alta concentração de vulnerabilidades	Junho e Dezembro	Junho
	Meses com baixa concentração de vulnerabilidades	Fevereiro, Março, Abril e Setembro	Janeiro
Sistemas diferentes do Windows	Meses com alta concentração de vulnerabilidades	Dezembro	Junho
	Meses com baixa concentração de vulnerabilidades	Fevereiro	Fevereiro e Setembro
Aplicações Web	Meses com alta concentração de vulnerabilidades	Dezembro	Fevereiro e Junho
	Meses com baixa concentração de vulnerabilidades	Setembro e Novembro	Janeiro

Outra questão importante está relacionada aos dados coletados. Em inúmeras situações o NVD retornou meses em que nenhuma vulnerabilidade foi divulgada. Sistemas como o Apache e IIS, por exemplo, obtiveram respectivamente 19 e 35 vulnerabilidades ao longo dos dez anos. Em trabalhos futuros seria interessante avaliar o que realmente aconteceu em tais meses como forma de fornecer contexto a análise dos resultados.

5 Conclusão

O objetivo do proposto trabalho foi encontrar possíveis sazonalidades em uma base de dados de dez anos coletados para os sistemas operacionais Windows, sistemas operacionais diferentes do Windows e aplicações Web afim de obter um comparativo dos resultados do presente trabalho com os resultados do trabalho proposto em (JOH; MALAIYA, 2009).

Em resumo, a análise dos dados das vulnerabilidades coletadas consistiu em cálculo do índice sazonal para cada sistema, aplicação da função de autocorrelação e construção de gráficos box-plot para os sistemas que apresentaram sazonalidade.

Quando comparado com o trabalho de Joh e Malaiya (2009) foi observado que a sazonalidade presente neste trabalho obteve mudanças. Os autores concluíram que os sistemas operacionais Windows obteve sazonalidade nos meses de junho e dezembro, já os sistemas operacionais diferentes do Windows e as aplicações Web obtiveram sazonalidade no mês de dezembro. Contudo, o presente trabalho conclui que todos os grupos de sistema obtiveram sazonalidade no mês de junho.

Ao longo do desenvolvimento deste trabalho, foi possível concluir que com apenas a metodologia mencionada nos capítulos anteriores não é possível comprovar que as sazonalidades encontradas e analisadas no capítulo de resultados são tendenciosas. Por isso, para trabalhos futuros, afim de obter uma previsão de tendências futuras de descoberta de vulnerabilidades é preciso utilizar um modelo de séries temporais chamado ARIMA (*Modelo Auto-Regressivo Integrado de Médias Móveis*). Esse modelo ajuda a prever pontos futuros na série e são aplicados em casos em que os dados mostram evidências de não-estacionariedade, como os dados coletados para este trabalho.

Referências

- ALHAZMI, O. H.; MALAIYA, Y. K. Prediction capabilities of vulnerability discovery models. *Proceedings - Annual Reliability and Maintainability Symposium*, n. Ref 2, p. 86–91, 2006. ISSN 0149144X. Citado 2 vezes nas páginas 17 e 19.
- ALHAZMI, O. H.; MALAIYA, Y. K. Application of vulnerability discovery models to major operating systems. *IEEE Transactions on Reliability*, v. 57, n. 1, p. 14–22, 2008. ISSN 00189529. Citado 2 vezes nas páginas 17 e 19.
- ALHAZMI, O. H.; MALAIYA, Y. K.; RAY, I. Measuring, analyzing and predicting security vulnerabilities in software systems. *Computers and Security*, v. 26, n. 3, p. 219–228, 2007. ISSN 01674048. Citado 2 vezes nas páginas 17 e 18.
- ANDERSON, R. Security in Open versus Closed Systems—The Dance of Boltzmann, Coase and Moore. *at Open Source Software Economics*, p. 127–142, 2002. Citado na página 18.
- APARECIDO, C. A. G. M.; BELLEZI, M. A. Análise de Vulnerabilidades com OpenVAS e Nessus. p. 34–44, 2014. Citado na página 15.
- ARSHAM, H. *Time Series Analysis for Business Forecasting*. 1994. Disponível em: <http://home.ubalt.edu/ntsbarsh/Business-stat/stat-data/Forecast.htm>. Citado na página 23.
- BOWERMAN, B. L.; O'CONNELL, R. T. *Time series forecasting: unified concepts and computer implementation*. Duxbury Press, 1987. ISBN 9780871500700. Disponível em: <https://books.google.com.br/books?id=h1QZAQAIAAJ>. Citado na página 23.
- CERT. *US-CERT | United States Computer Emergency Readiness Team*. 1991. Disponível em: <https://www.us-cert.gov/>. Citado na página 15.
- CONDON, E.; HE, A.; CUKIER, M. Analysis of Computer Security Incident Data Using Time Series Models. *2008 19th International Symposium on Software Reliability Engineering (ISSRE)*, p. 77–86, 2008. ISSN 1071-9458. Disponível em: <http://ieeexplore.ieee.org/document/4700312/>. Citado na página 12.
- CVE. *CVE - Common Vulnerabilities and Exposures (CVE)*. 1985. Disponível em: <https://cve.mitre.org/>. Citado na página 15.
- FONTES, E. L. G. *Segurança da Informação*. [S.l.]: Editora Saraiva, 2017. Citado na página 14.
- GOEL, J. N.; MEHTRE, B. M. Vulnerability Assessment & Penetration Testing as a Cyber Defence Technology. *Procedia Computer Science*, Elsevier Masson SAS, v. 57, p. 710–715, 2015. ISSN 18770509. Disponível em: <http://dx.doi.org/10.1016/j.procs.2015.07.458>. Citado na página 11.
- JOH, H.; JINYOO, K.; MALAIYA, Y. K. Vulnerability discovery modeling using weibull distribution. In: *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*. [S.l.: s.n.], 2008. p. 299–300. ISBN 9780769534053. ISSN 10719458. Citado na página 17.

- JOH, H.; MALAIYA, Y. K. Seasonal variation in the vulnerability discovery process. *Proceedings - 2nd International Conference on Software Testing, Verification, and Validation, ICST 2009*, p. 191–200, 2009. ISSN 2159-4848. Citado 6 vezes nas páginas 11, 12, 19, 20, 37 e 39.
- JOH, H.; MALAIYA, Y. K. Periodicity in software vulnerability discovery, patching and exploitation. *International Journal of Information Security*, Springer Berlin Heidelberg, 2016. ISSN 1615-5262. Disponível em: <<http://link.springer.com/10.1007/s10207-016-0345-x>>. Citado 2 vezes nas páginas 17 e 19.
- MASSACCI, F.; NGUYEN, V. H. An empirical methodology to evaluate vulnerability discovery models. *IEEE Transactions on Software Engineering*, v. 40, n. 12, p. 1147–1162, 2014. ISSN 00985589. Citado 3 vezes nas páginas 17, 18 e 19.
- MIANI, R. S. Um estudo sobre métricas e quantificação em segurança da informação. 2013. Citado 2 vezes nas páginas 12 e 24.
- MUSA, J. D. Software Reliability Engineering. v. 2, p. 368, 2004. Citado na página 18.
- MUSA, J. D.; OKUMOTO, K. A Logarithmic Poisson Execution Time Model for Software Reliability Measurement. In: *Proceedings of the 7th International Conference on Software Engineering*. Piscataway, NJ, USA: IEEE Press, 1984. (ICSE '84), p. 230–238. ISBN 0-8186-0528-6. Disponível em: <<http://dl.acm.org/citation.cfm?id=800054.801975>>. Citado na página 18.
- NIST. *NVD - Home*. 2004. Disponível em: <<https://nvd.nist.gov/>>. Citado na página 15.
- OSVDB. *OSVDB | Everything is Vulnerable*. 2002. Disponível em: <<https://blog.osvdb.org/>>. Citado na página 15.
- OZMENT, A.; SCHECHTER, S. E. Milk or Wine : Does Software Security Improve with Age? *15th USENIX Security Symposium (Security '06)*, p. 93–104, 2006. Citado na página 18.
- PEOTTA, L.; GONDIM, P. Análise de Risco em Ambientes Corporativos na Área de Tecnologia da Informação. *SEGeT – Simpósio de Excelência em Gestão e Tecnologia*, p. 1–12, 2006. Disponível em: <http://www.aedb.br/seget/artigos07/1049{_}risco.> Citado na página 16.
- POSITIVE TECHNOLOGIES. Vulnerabilities In Corporate Information Systems. 2018. Citado na página 11.
- RESCORLA, E. Is finding security holes a good idea? *IEEE Security and Privacy*, v. 3, n. 1, p. 14–19, 2005. ISSN 15407993. Citado na página 17.
- SECUNIA. *Computer Security Research - Secunia*. 2009. Disponível em: <<https://secuniaresearch.flexerasoftware.com/community/research/>>. Citado na página 15.
- SHAMELI-SENDI, A.; AGHABABAEI-BARZEGAR, R.; CHERIET, M. Taxonomy of Information Security Risk Assessment (ISRA). *Computers & Security*, v. 57, 2015. Citado na página 11.
- SINGH, H.; SURENDER, J.; PANKAJ, K. V. Penetration Testing: Analyzing the Security of the Network by Hacker's Mind. *Ijltemas*, V, n. V, p. 56–60, 2016. Citado na página 11.

SPANCESKI, F. R. Política De Segurança Da Informação – Desenvolvimento De Um Modelo Voltado Para Instituições De Ensino. 2004. Citado na página 14.

WHITMAN, M. E.; MATTORD, H. J. *Principles of information security*. [S.l.]: Cengage Learning, 2011. Citado na página 15.

XIAO, C. et al. From patching delays to infection symptoms: using risk profiles for an early discovery of vulnerabilities exploited in the wild. In: *27th USENIX Security Symposium (USENIX Security 18)*. [S.l.: s.n.], 2018. p. 903–918. Citado 2 vezes nas páginas 6 e 15.

YOUNIS, A.; JOH, H.; MALAIYA, Y. Modeling learningless vulnerability discovery using a folded distribution. *Proc. of SAM*, v. 11, p. 617–623, 2011. Disponível em: <http://www.lidi.info.unlp.edu.ar/WorldComp2011-Mirror/SAM8765.pdf>. Citado na página 18.