May 2018

Statistical Analysis and Modeling of Cyber Security and Health Sciences

Nawa Raj Pokhrel
University of South Florida, nawaraj@mail.usf.edu

Statistical Analysis and Modeling of Cyber Security and Health Sciences

by

Nawa Raj Pokhrel

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Mathematics & Statistics
College of Arts and Sciences
University of South Florida

Major Professor: Chris P. Tsokos, Ph.D.
Kandethody M. Ramachandran, Ph.D.
Lu Lu, Ph.D.
Yuncheng You, Ph.D

Date of Approval:
May 17, 2018

Keywords: Software Vulnerability, ARIMA, Differential Equation, Operating System, SHEER

Dedication

To my parents Kaladhar and Tulasi.
To my brother Keshav and sister-in-law Goma.
To my wife Kabita.

Acknowledgments

First and foremost, my deepest gratitude to my major Professor Chris P. Tsokos, whose immeasurable advice, unflinching support, and continuous guidance are always my source of inspiration. Without his guidance this dissertation would not have been possible. Words are not enough to thank him.

I am also greatful to the members of my dissertation committee: Dr. Kanadethody M. Ramachandran, Dr. Lu Lu, and Dr. Yuncheng You for their support and advice throughout this process. I am thankful to Dr.Yicheng Tu for chairing the session. I would like to express my sincere thanks to Dr. Netra khanal and Dr. Keshav Pokhrel for their vital contribution in part of this dissertation.

I am indebted to Mahdi Goudarzi, Hansapani Sarasepa Rodrigo, Mohamed Abu Sheha, Freeh Alenezi, Emmanuel A. Appiah, Sulav Malla and all my fellow graduate friends for their constructive suggestions and critiques.

Finally, I would like to express my heartfelt thanks to my parents for manifold blessings and my wife Kabita for support and encouragement. I would not have been able to accomplish this feat without them.

Table of Contents

List of Tables

List of Figures

Abstract

Being in the era of information technology, importance and applicability of analytical statistical model an interdisciplinary setting in the modern statistics have increased significantly. Conceptually understanding the vulnerabilities in statistical perspective helps to develop the set of modern statistical models and bridges the gap between cybersecurity and abstract statistical /mathematical knowledge. In this dissertation, our primary goal is to develop series of the strong statistical model in software vulnerability in conjunction with Common Vulnerability Scoring System (CVSS) framework. In nutshell, the overall research lies at the intersection of statistical modeling, cybersecurity, and data mining. Furthermore, we generalize the model of software vulnerability to health science particularly in the stomach cancer data.

In the context of cybersecurity, we have applied the well-known Markovian process in the combination of CVSS framework to determine the overall network security risk. The developed model can be used to identify critical nodes in the host access graph where attackers may be most likely to focus. Based on that information, a network administrator can make appropriate, prioritized decisions for system patching. Further, a flexible risk ranking technique is described, where the decisions made by an attacker can be adjusted using a bias factor. The model can be generalized for use with complicated network environments.

We have further proposed a vulnerability analytic prediction model based on linear and nonlinear approaches via time series analysis. Using currently available data from National Vulnerability Database (NVD) this study develops and present sets of predictive model by utilizing Auto Regressive Moving Average (ARIMA), Artificial Neural Network (ANN), and Support Vector Machine (SVM) settings. The best model which provides the minimum error rate is selected for prediction of future vulnerabilities.

In addition, we purpose a new philosophy of software vulnerability life cycle. It says that vulnerability saturation is a local phenomenon, and it possesses an increasing cyclic behavior within

the software vulnerability life cycle. Based on the new philosophy of software vulnerability life cycle, we purpose new effective differential equation model to predict future software vulnerabilities by utilizing the vulnerability dataset of three major OS: Windows 7, Linux Kernel, and Mac OS X. The proposed analytical model is compared with existing models in terms of fitting and prediction accuracy.

Finally, the predictive model not only applicable to predict future vulnerability but it can be used in the various domain such as engineering, finance, business, health science, and among others. For instance, we extended the idea on health science; to predict the malignant tumor size of stomach cancer as a function of age based on the given historical data from Surveillance Epidemiology and End Results (SEER).

Chapter 1
Introduction

In today's world, there is a tremendous amount of demand to protect an individual computer or computer networks from unauthorized users. Almost all work in this domain is focused on the particular aspect of the problem ignoring the holistic perspective thus systems are robust to attack. The core logic behind the scene is, almost all cybersecurity problems are by nature interdisciplinary. Very little has been done in the scientific and research community to incorporate the cybersecurity problem in holistic perspective on interdisciplinary settings. We believe that statistics as a discipline could thrive in cross disciplinary platform. To strengthen this idea, we have breached the traditional academic boundaries and developed a series of statistical models to address the problem quantitatively in the field of cybersecurity. In a nutshell, the overall research lies at the intersection of statistical modeling, cybersecurity, and data mining. Furthermore, we have generalized the model developed in software vulnerability to health science particularly in stomach cancer data.

This dissertation consists of five complete chapters where the first four chapters are dedicated to present customized and standard models with their application in the given subject area. The methods we developed are not only applicable to cybersecurity domain but can be used in engineering products, health oriented systems, business and finance forecasting, among others. For instance, in Chapter 5, the developed model in software vulnerability is generalized in health science.

This chapter is organized as follows: Section 1.1 discusses our background and research problems. In Section 1.2, a model developed to compute overall security risk will be discussed. Section 1.3 will provide the predictive modeling of desktop Operating System using linear and non linear approach. Section 1.4 explores the differential equation model to predict the future vulnerability. Finally, the model we develop in software vulnerability is generalized in health system on Section 1.5.

1.1 Background and Research Problem

Cyber attacks are considered daunting challenges to individuals, small to medium to large scale companies/enterprises, educational institutions, governments, among others. Even if one uses sophisticated hardware, software, and high level of security measures, they are susceptible to attack. In today's world, none of the hardware, software, network architectures and devices are developed free of vulnerabilities. Vulnerability is simply the loophole that opens the door to exploit the sensitive resources to unauthorized users. Those who are directly involved in accessing the resources illegally are called hackers. In general depending on the complexity and size of the network, Information Technology (IT) manager(s)/system administrator(s) are responsible to protect any computer resources in an efficient way. There is always a tug of war between hackers and system administrators. To implement the proper plan, system administrators must know their own system weakness and behavior of the attackers. In a nutshell, to protect the resources, we should have a complete idea of our own weakness and behavior of the attackers. These two dimensions are the primary ingredients to protect the computer resources. This dissertation focuses on either of the dimension.

The overall cost and severity of the problem in cyberattacks are skyrocketing almost daily. A case in point is in 2017 National Vulnerability Database [1]; Flexera Software Vulnerability Report [2] and CVE details [3] websites recorded 14,712 new vulnerabilities which is almost 60 % higher than the previous year 2016. In the first four months of year 2018, a total of 4,393 vulnerabilities were added. According to Cisco, the number of denial of service attacks flood the system server's with irrelevant/junk traffic globally by 172 %. They also predict the overall growth of such type of activities by another two and a half times to 301 millions attacks by 2021. According to Cybersecurity Ventures Report [4], cybercrime the total cost in the world is around 6 trillion dollars annually by 2021. From the Yahoo hack almost 3 billion users were affected. In 2017 the Wanna Cry ransomware attack alone, spread over 150 countries and damages were around $4 billion. Not only the numbers of attacks but also the magnitude/severity of attacks is on the rise as well. Cyberattacks are progressively disastrous and trget the broader area even from politics to health sciences. Obama's administration proposed a $19 billion budget for cybersecurity. In the presidential campaign, Hillary's personal emails became front page news. In May 2017, Trump signed an executive order to improve the existing cybersecurity internal and external threats. Spending that amount of money from a high level indicates an acknowledgment to safeguards the nations digital assets

and to minimize the damages and losses.

In recent days, online services have become so popular with consumers and businesses alike. On the other hand, hacking is more easier than ever in a sense that more accessible technology is available to everyone. The internet makes it more convenient for world but at the same time it exposes their users to new threats.

One of the prominent concerns is the entire nature of the business process has completely changed in the digital age. In almost all cases, technology is not considered to be a supplemental aspect of business operation rather assets of the company solely depends on their computer networks as their core operations. For instance, the world's valuable companies from Facebook to Amazon to Uber to Ebay and several other companies' core business operations depend upon computer networks. This leads to a high level of investment to protect the core computer network to run the business efficiently and effectively.

It is worth mentioning here that applying scientific methodology and integration of knowledge of cybersecurity, statistical modeling, and data mining should be of major priority. Looking at the vulnerabilities in statistical perspective and analyzing the vulnerabilitiy data to predict the future behavior would play a crucial role in the computer security decision making process. Thus, our objective is to develop an innovative model in the vulnerability analysis and make a contribution for further development of this area.

1.2 Stochastic Model to Determine Overall Network Security Risk

The primary purpose of this Chapter is to develop the analytical statistical model to determine the overall network security risk using Markovian process in conjunction with Common Vulnerability Scoring System (CVSS) frame work. Unlike our approach which is used to determine the overall risk of the computer network, several studies tackled the network security problem but from different perspectives. A recent study [5] introduced simulation based study using Markovian process to develop security matrices.. Reference [6] employs the Markov property to rank each node in the security attack model. Cynthia and Laura [7] analyzed the risk to a specific network based on the attack paths that have high probability of successful attacks. Alternatively, reference [8] presented a quantitative approach to evaluate a risk based on system architecture and lists of the vulnerabilities associated with each component. Another similar recent work [9] presented a ranking scheme for the state of an attack graph that determines the probabilities of attackers reaching the states of the attack graph.

Reference [10] developed the stochastic model in conjunction with vulnerability life cycle. Conditional risk assessment technique was developed with respect to the known vulnerabilities. Similarly, [11] presented a model and methodology for security risk analysis of enterprise networks using probabilistic attack graph. The novelty of [12] this work comparing to the previous ones was that it adopted multi-agents technology in the risk assessment process and algorithm to compute the different risk indexes.

We purpose a novel model to determine overall network security risk based on exploitability and impact sub-score along with skills and expertise of the attackers. Utilizing firewall rules, vulnerabilities contained on each host, and services running on each host, an attack graph is developed called host access graph. Using Markovian random walk, risks are prioritized and the sum of the risk associated with each node makes the total security risk present in computer network.

1.2.1 Vulnerability Database

Our study on series of vulnerability analytical models in each chapter is based on available data. The most commonly used and best available data source is National Vulnerability Database (NVD) [1]. NVD, run by the U.S. Department of Commerce's National Institute of Standards and Technology, builds off the catalog of "Common Vulnerabilities and Exposures"(CVEs) maintained by non profit Mitre Corp. NVD is based on the Common Vulnerability Scoring System (CVSS) [13]. CVSS is the open framework that provides the quantitative scores representing the overall severity and risk of the known vulnerabilities. It is maintained by the **Forum of Incident Response Team** (FIRST) [14]. A CVSS score is on the scale of 0 to 10 and consists of three major metrics group: base, temporal and environmental as mentioned in Figure 1. Vulnerabilities with the base score range from 0-3.9 is considered **Low**, vulnerability, 4.0-6.9 as **Medium**, and 7.0-10 as **High**. The base score is computed using two sub-scores; $\mathbf{Exploitability\ sub-score}$ and $\mathbf{Impact\ sub-score}$ using standard expression mentioned in Figure. In Chapter 2, two sub-scores (Exploitability, Impact) are the fundamental quantitative value for our analysis. Furthermore, in Chapters 3 and 4, base score is employed. Further explanation of the listed equations on the schematic diagram is explained in the Appendix.

1.2.2 Introduction to Markov Chain

Mathematically, a Markov chain can be defined as a discrete stochastic process [32]. More specifically let S be a set of states (in the present study S is finite, we can think of it as nodes in host

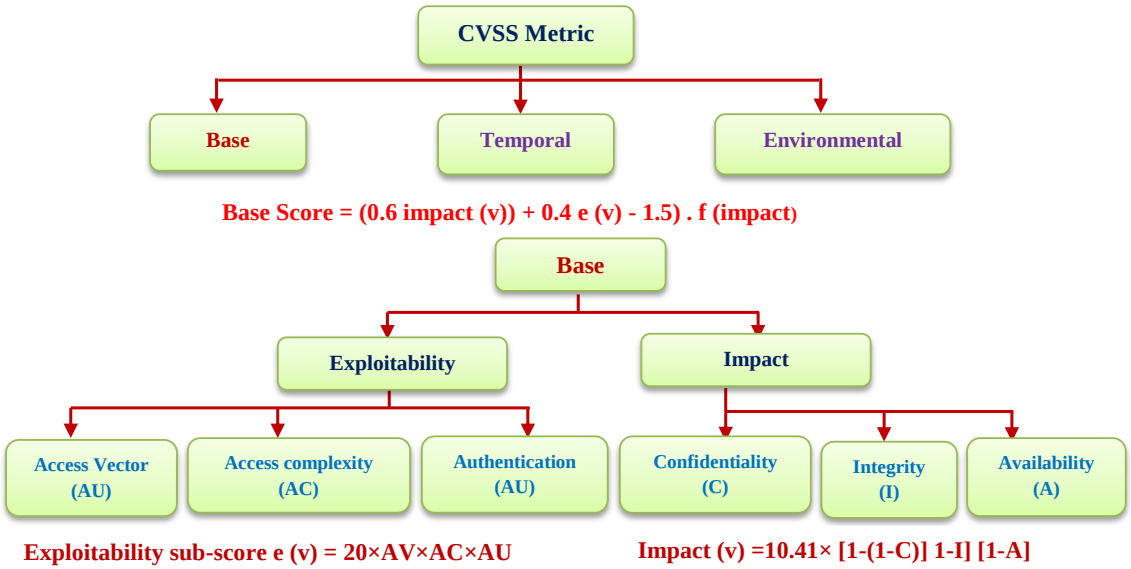

Figure 1.: Common vulnerability system for base metric calculation model

access graph). A Markov chain is a sequence of random variables X_0, X_1, X_2,.....,$X_n \in$ S that satisfies the "Markovian property", that is,

$$\text{P}[X_{n+1}=\text{y} \mid X_0=x_0, X_1=x_1 ..., X_n=x_n]=\text{P}[X_{n+1}=\text{y} \mid X_n=x_n]$$

We make the assumption that transition probabilities do not depend on time. The transition probability ($P_{x,y}$) defined as follows.

$$\text{P(x,y)}=\text{P}[X_{n+1}=\text{y} \mid X_n=x], \text{ for all n.}$$

The transition probability matrix of $P_{x,y}$ is N $\times$ N matrix whose (x,y) entry should satisfied the following properties.

$$0 \leq P_{x,y} \leq 1,\ 1 \leq x, y \leq N$$

and

$$\sum_{y=1}^{N} P_{x,y} = 1,\ 1 \leq x \leq N.$$

If any matrix satisfies the above two conditions then it is considered as transition matrix for the Markov Chain. In our analysis we have implemented discrete Markovian process.

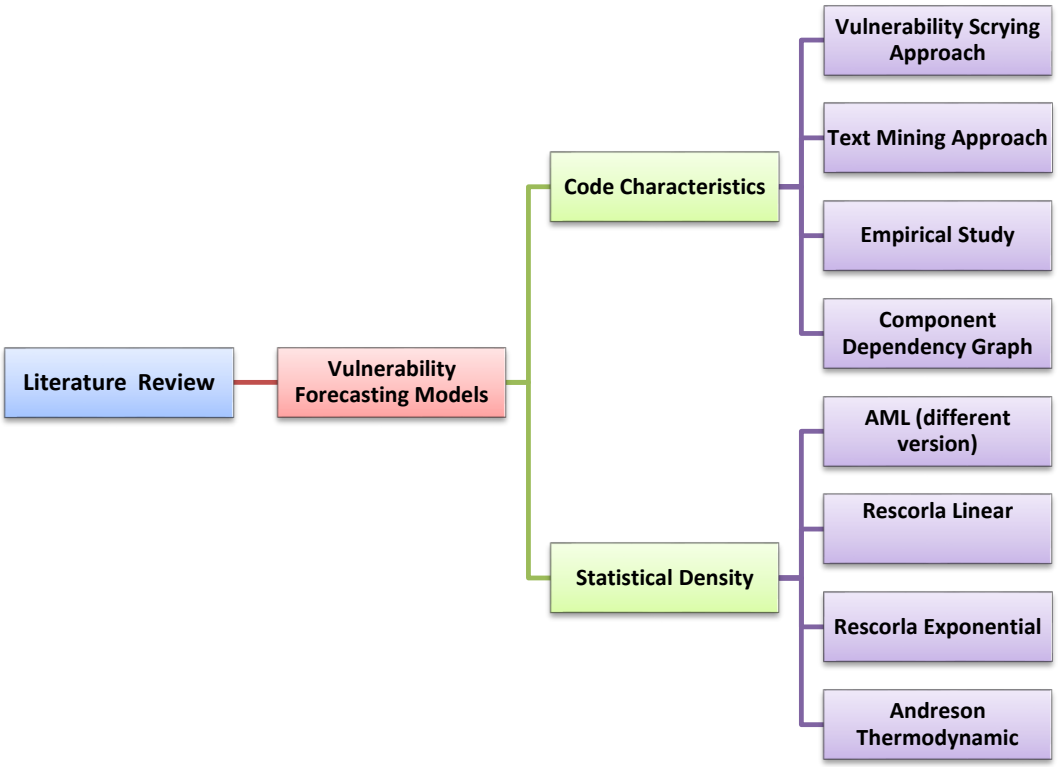

Figure 2.: Previous models on software vulnerability

1.3 Predictive Model for Desktop Operating System

The primary purpose of this chapter is to develop the time series predictive model to predict the future vulnerabilities using linear and non-linear techniques. In linear setting, we have employed Auto-Regressive Moving Average (ARIMA). On the other hand, Support Vector Machine (SVM) and Artificial Neural Networks (ANN) methods are implemented in a non-linear setting. We have selected three major Operating Systems, namely, Windows 7, Mac OS, and Linux kernel by using the listed vulnerabilities on National Vulnerability Database (NVD). As far as we know, this is the first study on the development of vulnerability prediction model using time series analysis by incorporating non-linear behavior. There are few existing models that have been developed in the scientific and research community with predefined assumptions and the given framework. Broadly speaking, previous research on this domain can be explained in schematic network diagram 2.

Code characteristics models rely on finding out the relationship between attributes of the code with its corresponding vulnerabilities. Rahimi and Zargham, [15] proposed a vulnerability scrying approach, that is, a vulnerability discovery prediction method based on code properties and quality. S. Riccardo et. al. [16] proposed a machine learning approach to predict which components of the

software applications contain security vulnerabilities using a text mining approach. This approach identified a series of vulnerable terms contained in its source code and was used to compute its frequency. Based on the frequency, they proceeded to forecast its future. Shin et. al. [17] performed an empirical study with traditional metrics of complexity, code churn, and faulty history using a large open source project to determine whether fault prediction models can be used for vulnerability prediction models . Nguyen and Tran, [18] proposed a component dependency graph to predict vulnerable components using machine learning methodology. All the mentioned approaches requires source code to build the models; source code of the OS is dynamic in nature and it is not available to the public in case of proprietary OS.

Statistical density based forecasting models use historical data to predict the future. To fulfill this objective, various kinds of models have been developed, mainly Alhazmi-Malaiya Logistic (AML) with different versions, [19] Poisson Log Arithmetic Model, [20], Rescorla Exponential Models [21], and Andersons Thermodynamics Model [22]. All the developed models are based on their underlying assumptions and defined framework and none of them considers the non linear behavior of the signal.

For code characteristics based models, we need a source code of the given software to develop statistical models. In reality, source code of the commercial OS is not available to the public. On the other hand, statistic density based models that have been developed are based on a series of underlying assumptions and criteria may or may not be applicable. Because of such limitations that exist on both categories, we should identify an alternative approach to forecast the future vulnerability of an OS by using time series analysis. Our model considers trend, level, and seasonality components if they exists. Similarly, to analyze the non-linear behavior of the number of vulnerabilities, we implemented ANN and SVM methodology.

1.4 A Predictive Software Vulnerability Model using Differential Equation

In this chapter, we have extended the vulnerability forecasting model to a more customized model using differential equations. Before developing the model, we looked deeply at the nature of vulnerability during the life cycle of the software because vulnerabilities are discovered throughout the entire life cycle of the software. We have developed the new philosophy of vulnerability life cycle and based on that idea, we have developed a new time based non linear differential equation model. Our proposed model is based on the fact that the vulnerability saturation is a local phenomenon, and it possesses an increasing cyclic behavior within the software vulnerability life cycle. The daily

vulnerability data is extracted from National Vulnerability Database (NVD) and is designed to obtain cumulative quarterly dataset. We purpose the strong statistical data driven model that best fits the available data, and projects future vulnerabilities along with current and future trends based on historical data. We even compare our model with the existing models and our model stands out best in terms of fitting and prediction capabilities.

1.5 Health Science : A Predictive Analytical Model for Stomach Cancer Data

In this chapter, we have generalized the differential equation model in stomach cancer data. The predictive model is not only applicable to software vulnerability but it can be used in various domains such as engineering, finance, business, health science, among others. For instance, we have implemented the idea on health science, to predict the malignant tumor size of stomach cancer as a function of age, based on the given historical data. Our analysis and modeling is based on the data obtained from Surveillance Epidemiology and End Results (SEER) program of the United States.

Chapter 2

A Stochastic Predictive Model to Determine Overall Network Security Risk

In this chapter, we propose a stochastic model to determine the overall network security risk using markovian process in conjunction with Common Vulnerability Scoring System (CVSS). By constructing host access graph with the help of provided firewall rules, list of the nodes with its rank is prepared via ranking algorithm. Finally, summing up the risk associated with each node yields total risk present in the given network. One article is published [23] based on findings of this Chapter.

The rest of this study is organized as follows: Section 2.1 describes the overall introduction to the subject area. In Section 2.2, major components and statistical methodology employed will be discussed. Section 2.3 will provide the analytical framework of our study. Section 2.5 explores the implementation of our idea in specific network environment. Finally, contribution is presented in Section 2.6.

2.1 Introduction

Computer networks are undoubtedly vulnerable no matter what level of hardware, software or a combination of both types of security parameters are incorporated. As long as the network servers provide services on different host servers, they depend on the server software that may have security holes which makes them susceptible to malicious attacks. To detect and/or prevent the network accessible resources from suspicious attacks, various commercial Intrusion Detection Systems (IDSs) [24] /Prevention Systems are available in the market. These intrusion detection/prevention based tools provides some sort of a signal that alerts the network administrator and provides them a partial picture of the network [25]. One of the most important challenges on today's networks is to develop the mechanism to aggregate the security risk of all systems in a network to evaluate the overall security risk.

In order to evaluate the security risk of a large scale enterprise, an administrator must consider not only single vulnerability exploit but also the multi-stage and the multi-host vulnerability attack used

by the attackers. To incorporate this fact, an attack graph is built to find out the logical relationship between multiple exploits. However, when size and complexity of the network increases, two major problems occur. First, the attack graph grows exponentially when the size of the network and algorithm complexity increase. Secondly, comprehending the information conveyed by the graph becomes difficult. Therefore, the attack graph that addresses the issues mentioned earlier were chosen and we will explain further in the next section.

Very little has been done in scientific and research community to develop statistical model that quantify the overall network security risk. Most of the work focuses on qualitative and subjective aspect of networks without having formal statistical model. To get rid of this problem, we introduce the statistical model that uses Markov chains in conjunction with CVSS framework metrics to analyze risks associated with structures of various networks. The model can be used to identify critical nodes in the host access graph where attackers may be most likely to focus. Based on that information, a network administrator can make appropriate, prioritized decisions for system patching. Further, a flexible risk ranking technique is described, where the decisions made by an attacker can be adjusted using a bias factor. The model can be generalized for use with complicated network environments.

In the present study, we are proposing a stochastic model for the security risk evaluation for the entire network based on the $\mathbf{Exploitability\ sub-score}$ and $\mathbf{Impact\ sub-score}$. We are considering a realistic network topology having three host servers and each host consists of one vulnerability. Based on the network architecture and given firewall rules, a host access graph is constructed. From the host access graph one state transition probability matrix is computed by utilizing $\mathbf{Exploitability\ sub-score}$ and $\mathbf{Impact\ sub-score}$. By using the **Markovian** random walk, we can prioritize the risk associated with each node via ranking.

Finally, summing up the risk associated with all the nodes present in the network, we determine the overall network security risk. This quantitative value can be taken as a security metric to determine the risk of an entire network. Finally, the schematic network topology in our study represents a typical security system that is in operation. Thus, our proposed statistical model and methodology can be applied to a specific security system that is in place for a given company.

2.2 Background and Terminologies of Cybersecurity

In this section, we have defined some of the basic terminology related with cyber security. We also explain the basic idea of the Markov chain process that is implemented to develop the stochastic

model to achieve our objective. Figure 1 provides the schematic presentation of the CVSS framework, and vividly reveals the holistic idea to compute the base score along with **Exploitability sub − score** and **Impact sub − score**.These two sub-scores (Exploitability, Impact) are the fundamental quantitative value for our analysis.

2.2.1 Vulnerabilities

A vulnerability is a flaw that exists in computer resources or control that can be exploited by one or more threats. A software vulnerability [25] is an instance of an error in the specification, development, or configuration of software such that its execution can violate the security policy. Attackers normally use the known vulnerabilities which are listed publicly on National Vulnerability Database (NVD) to penetrate the system. Sometimes attackers may use a vulnerability that has not been disclosed publicly which is called zero day vulnerability. There is almost no defense against a zero day attack [26]. Zero day vulnerability remains unknown to vendors; thus information about the new vulnerabilities gives the attackers a free pass to attack any target host. The zero day attack has not been used in this study.

2.2.2 Markovian Properties and Attacking Process

A Markov chain is regarded as one of the best modeling techniques that has been used effectively in various fields such as reliability analysis [28], performance analysis, dependability analysis [29, 30], and cybersecurity analysis [31], among others. We will model the host access attack graph described in the previous subsection using a Markov chain with the real behavior of the attacker in conjunction with the Markovian properties.

The Markovian properties reveal the fact that the transitions between states are memoryless; transitioning to the next step depends only on the current state only and not on any of the other previous states. We can correlate this property with the attacker's behavior in a sense that an attacker needs to exploit several nodes before reaching the goal node. When the attacker starts attacking an immediate node to reach the goal node, there are many nodes available before reaching the goal node called **intermediate node**. When an attacker reaches any intermediate node, there is no memory of previous node. The attacker launches further attacks until the goal node is found. To advance the attack, the attacker should move from one intermediate node to another/several intermediate node/s. In the present study, we have assumed that selection of the best intermediate node depends on three parameters, namely **Exploitability sub − score**, **Impact sub − score**

and an individual skill of the attacker called **Bias factor**.

Without loss of generality, transition states are independent of time. Mathematically, there exists some transition probability matrix, P(x,y) such that

$$\mathrm{P(x,y)=P}[X_{n+1}=\mathrm{y} \mid X_n=x], \text{ for all n.}$$

We can create a new set of states S×[n], having a different set of states associated with each timestep. In the present study, P(x,y) represents the transition probability matrix. To simulate the Markov chain, a stochastic transition probability matrix P and the initial probability distribution is required. In the present study, initial risk associated with each nodes in the host access graph is considered as initial probability distribution which will be explained further in section 4. Once we have the stochastic matrix P and the initial risk, then utilizing the basic properties of Markovian process, we can determine the risk of the entire network.

2.2.3 Attack Graphs

Attackers usually penetrate any type of computer network via a chain of exploits where each exploit in the chain creates the foundation for upcoming exploits. A combination of such exploits make the chain called attack path; a collection of such attack paths develop the attack graph. An attack graph is a succinct representation of all paths through a system that ends in a state where an intruder has successfully achieved its goal [26]. There are many algorithms that have been developed in the scientific and research community to construct the attack graphs. However, it is very difficult to analyze the network via attack graph when a number of nodes and complexity of the network increase. As the scalability and complexity of the network increase exponentially, the computation cost to create the attack graph increases. As a result, it is difficult to interpret the attack graph precisely. On the other hand, most of the attack graphs are designed for a single target, and can not be used to evaluate the overall security of the networks with several targets. To address these striking problems Anming Xie, Zhuhua Cai, Cong Tang, Jianbin Hu, and Zhong Chen [27] developed a novel approach to generate and describe the attack graph. They developed a two layer attack graph, where the upper layer is a host access graph and the lower layer is composed of some host pair attack graphs. The lower level describes all the detailed attack scenarios between each host pair, and the upper level shows the direct network access relationship between each host pair by ignoring detailed information. In this study our stochastic model is based on upper layer attack graphs; that is, host access graphs have been utilized.

2.2.4 Common Vulnerability Scoring System(CVSS)

CVSS [3] is the open framework that provides the quantitative scores representing the overall severity and risk of the known vulnerabilities. It is maintained by the **Forum of Incident Response Team** (FIRST) [14]. A CVSS score is on the scale of 0 to 10 and consists of three major metrics group: base, temporal and environmental as mentioned in Figure 1. Vulnerabilities with the base score range from 0-3.9 is considered **Low** vulnerability, 4.0-6.9 as **Medium**, and 7.0-10 as **High**. The base score is computed using two sub-scores; $\mathbf{Exploitability\ sub-score}$ and $\mathbf{Impact\ sub-score}$ using standard expression mentioned in Figure 1. These two sub-scores are the fundamental quantitative value for our analysis.

2.3 Cyber Security Analytical Framework

The schematic network given below, Figure 1, shows a bird's eye view of our proposed cyber security model. The most important component of our model is the attack graph. It is constructed by considering the input as network topology, services running on each host, attack rules defined on firewalls, and vulnerabilities associated with each host running different services.

For simplicity limited numbers of nodes are present in our network illustration and we have developed a host access graph manually. However, as the size and complexity of the network increase, we can use any kind of attack graph generation tools [33] to construct the intended attack graph of interest. Nodes present on the attack graph represent the host. Each host runs different kinds of services and there may exist various vulnerabilities. CVSS assigns severity scores for each

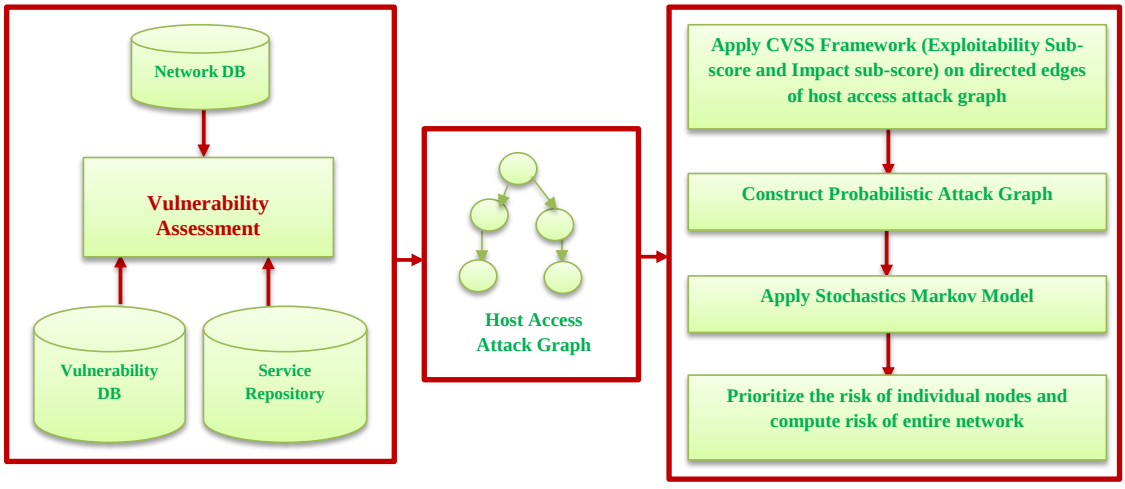

Figure 3.: Cyber security analytical framework

vulnerability. The score is computed based on the standard expressions. The standard expression depends on several matrices that provide a quantitative score to approximate ease and impact of exploit. In the present study, we have applied both scores to determine whether it is beneficial to move from one node to another node from the attacker's perspective.These two scores, that is, Exploitability sub-score and Impact sub-score, are combined to provide the basis of assigning the edges of attack graphs to represent the values of the probability distribution. This probability represents the possibility of a vulnerability to be exploited by an attacker. While implementing our stochastic model, the behavior of the attacker is another concern. In this study, we assume that the attacker will choose the vulnerability that maximizes the chances of succeeding in compromising the goal state. Due to any reason, if the attacker terminates attacking, then the attacker will move to the initial state. Finally, utilizing the properties of Markov chain, the risk of the individual node is computed. Nodes are prioritized based on computed risk. Then, we sum the risks of all the nodes that will give us the total security risk present in the network.

2.4 Model Representation

The central component of the proposed stochastic model exclusively depends on the host access graph mentioned in the previous section. Before delving into the modeling approach, let us formally explain the host access graph as shown in Figure 4.

In Figure 4 below, S_i, i=1, 2, 3,.....g are host nodes and S_g is a goal node. A node represents a host in the host access graph; thus, the number of nodes is equal to the number of hosts in the network. Similarly, directed edges between two nodes represent the access relationship between the corresponding two hosts so that there is only one directed edge from one node to another at most. Hence, there are no multiple edges in the graph, and our proposed model retains only the highest access achieved between the hosts, since higher levels of access to the destination host means more powerful attacks are achieved. A directed solid edge lines from host S_1 to host S_2 in Figure 4 represents the access available on S_2 from S_1. Similarly, dashed lines from host S_2 to host S_g states that there are other intermediate nodes present in between these nodes and the same explanation is applicable to other hosts.

Once the host access graph is constructed, then our basic foundation is developed for further analysis. To make this graph more applicable and realistic, we have modified it by adding one additional dummy node to represent the attacker. The attacker starts exploiting the immediate node by gaining a high level of privileges. In reality, even if an attacker is equipped with sophisticated

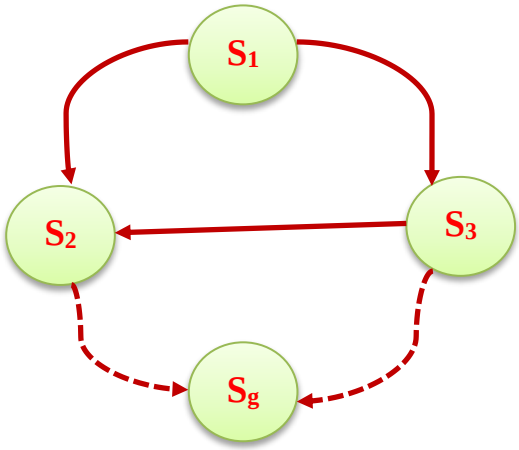

Figure 4.: An example of host access graph

tools and a high level of experience, there is no guarantee that he/she will reach the goal node. This may happen due to reaching a level of difficulty or being discovered by an intrusion response team or any type of unusual circumstance.

Whenever the attacker stops launching attacks at any point due to any reason, then he/she goes back to the initial state from where the attacking began. To incorporate this attack scenario, a dummy node A is introduced. For any node S_i, we define the edge (S_i, A). This is demonstrated by Figure 5 below, where a node A represents an attacker. There is a directed solid edge from every node to the attacker node A, this implies that when the attacker gives up exploiting the node further due to any reason, again he/she goes back to the initial state and proceeds to search for alternative options. Similarly, the meaning of dashed lines are similar to Figure 4 as mentioned before. In

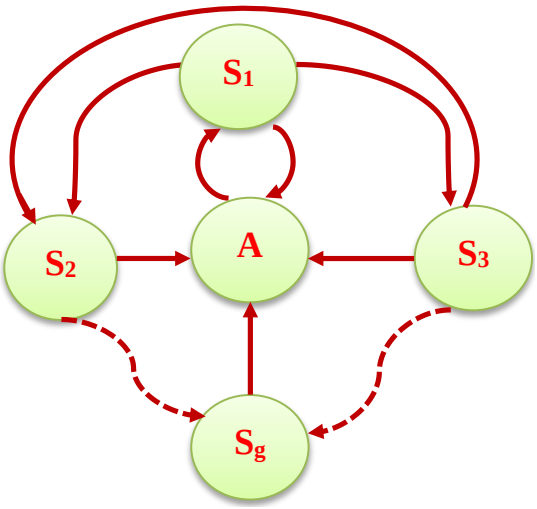

Figure 5.: Modified host access graph

our proposed model, the attacker starts attacking the immediate node and keeps on launching attacks until it reaches the goal node. One big question that arises here is **"what happens if the attacker is encountered with the multiple nodes to reach the goal node and on what basis the attacker decides to select the best node from the available alternatives."** We have assumed that the attacker's decision solely depends on two parameters. The first parameter is **Exploitability**; it is all about the level of complexity involved to attack the vulnerable node. The second parameter deals with **Impact** factor, which means how much impact can an attacker make when a vulnerable node is exploited. CVSS provides numerical scores scale of (0,10) where 0 signifies the most secure and 10 signifies the least secure of the mentioned parameters. These two parameters are conceptually expressed by,

$$\boldsymbol{ExploitabilityBenefit = f(Exploitability, Impact)} \tag{2.1}$$

In Equation 2.1, we have coined the new term **ExploitabilityBenefit**. It is defined as the function of **Exploitability** and **Impact** factor. Using these values an attacker determines the level of benefit to change from one to another node. To clarify this idea, let us take any two nodes from the host access graph as shown below, by Figure 6, where S_j and S_k are node j and node k, respectively with V_j and V_k being the corresponding vulnerabilities.

In Figure 6, there is a directed edge from node j to node k. An attacker makes the decision whether to move from node j to node k based on the **ExploitabilityBenefit** value. Moreover, making a final decision to move from one node to another node not only depends on **Exploitability** and **Impact** factor but also depends on the skills and expertise of the attacker. This is the subjective factor and varies from individual to individual.

In reality, it is an indispensable factor to make the attacking decision. We have represented this parameter in our model and is termed a **Bias** factor, **Bias** in a sense that its value varies from attacker to attacker. Incorporating all three mentioned parameters (**Exploitability**, **Impact**, **and Bias**), Equation 2.1 is further extended mathematically to,

$$\boldsymbol{a_{jk} = \beta Exp(v_k) + (1-\beta) Impact(v_k) \qquad 0 < \beta < 1} \tag{2.2}$$

In the above Equation 2.2, a_{jk} is the **ExploitabilityBenefit** score to move from the node j to node k. Similarly, $\mathbf{Exp(v_k)}$ is a function that measures the level of difficulty in exploiting the node

Figure 6.: A couple of nodes in host access graph

k. The quantitative value that determines the level of difficulty scale of 0 to 10 is provided by CVSS. On the other hand, **Impact**($\mathbf{v_k}$) is a function that measures the potential damages or losses that occur due to a successful exploitation of node k and its quantitative score is provided by CVSS. The possibility that a successful exploitation and damages/losses occur depends on experience and skills of the attacker. To incorporate this fact, we introduce a **Bias** factor β; its value ranges from 0 to 1 to indicate the level of experience and skills that the attacker possesses. When we combine **Exploitability** and **Impact** score with their corresponding **Bias** factor, a final weighted value is obtained to move from the node j to node k. To move the attacker from the initial node to the goal node, he/she needs to penetrate several intermediate nodes. Let us assume j is the initial node and g is the goal node and consists of three intermediate nodes namely k, l, and m. One possibility is that the attacker reaches the goal node by exploiting node j to node k, node k to node l, node l to node m, and finally node m to node g. To materialize this idea in mathematical notion, we need to construct the weighted adjacency matrix A as shown below.

$$\boldsymbol{A} = \begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0g} & \cdots & a_{0n} \\ a_{10} & 0 & \cdots & a_{1g} & \cdots & a_{1n} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ a_{n0} & a_{n1} & \cdots & a_{ng} & \cdots & 0 \end{bmatrix}$$

Each element of the adjacency matrix is computed using Equation 2.2. Diagonal values of the adjacency matrix are all zero because no cost is involved to move from the current node to the node itself. Elements of the matrix A are not normalized, thus, the non normalized values are converted into probabilities using Equation 2.3. This equation reveals the fact that in each step the attacker goes from node j to k with probability given by

$$p_{jk} = \frac{\boldsymbol{A}(j,k)}{\sum_l \boldsymbol{A}(j,l)} \tag{2.3}$$

Writing Equation 2.3 in matrix form we have,

$$\boldsymbol{P = DA} \tag{2.4}$$

Where, **A** is the weighted adjacency matrix. P is the transition matrix that provides the transition probability that the attacker moves from one state to another state and D is the diagonal matrix computed using Equation 2.5 below,

$$\boldsymbol{D_{jk}} = \begin{cases} \frac{\mathbf{1}}{\sum_l \boldsymbol{A(j,l)}} & \text{if j=k} \\ \mathbf{0} & \text{Otherwise} \end{cases} \tag{2.5}$$

Finally, we have constructed the transition matrix (using Equation 2.4) representing transitions probability that an attacker moves from one state to another state, that is, from state j to state k.

2.4.1 The Risk Based on Ranking

Consider an attacker starts attacking from the initial node to the goal node. The attacker must obtain a user level or root level of privilege on the intermediate node to advance the attack further to reach to the goal node. In reality the attacker should try to obtain the highest level of privilege. Host access graphs are created based on the philosophy of gaining high level of privilege. Nodes of the host access graph are treated as OR nodes, which can be satisfied if any of the child node is true. The risk analysis is based on the relative rank value for every node of the host access graph. R is the risk vector and its initial risk value is computed based on the number of hosts present in the host access graph. Suppose there exist N nodes in the host access graph; then simply set all the node ranks equal to 1/N. This initial risk is first injected by the starting node of an attacker. This risk value flows level by level until convergence. The complete risk ranking algorithm is described by the schematic diagram given below by Figure 7.

The risk value of $\boldsymbol{r_k}$ for a node k depends upon the rank of its parents. The risk value of the node set by the initial node represents the starting node of the attacker. When the ranking process is started then intermediate risk value is computed via iteration. The intermediate value will flow level by level until a steady state is achieved. Mathematically, suppose $\boldsymbol{r_k}$ is the risk of node k given in the host access graph; then the risk of node k is computed using Equation 2.6, given by,

$$\boldsymbol{r_k} = \sum \boldsymbol{r_k p_{jk}} \tag{2.6}$$

Suppose, R = (r_1, r_2, r_3, ...,r_z) is the risk vector, where r_j is the rank of node j. Equation 2.6 is further extended to Equation 2.7 as shown below. The risk values are normalized, where $0 \leq r_k \leq 1$, for all j, and $\sum r_k = 1$. Thus, written in matrix form the risk vector R is given by R times the probability transition matrix P, that is,

$$R = RP \tag{2.7}$$

The value of R in Equation 2.7 is recursive and must be iteratively calculated until convergence, which is expressed by Equation 2.8, that is,

$$R^t = R^{t-1}P \tag{2.8}$$

The attacking process is based on the Markovian random walk, that is, an essential condition for the iterative computation to converge [34]. The probability distribution of risk analysis of the host access graph after the attacker follows one link is R_1=RP, where R is the risk vector and P is the one step transition probability matrix identified by Equation 2.4. Similarly, after two links the probability distribution is R_2=R_1P. Assuming this iteration converges to a steady state probability, then we have $R^t = R^{t-1}P$, where R^t is an eigenvector of P.

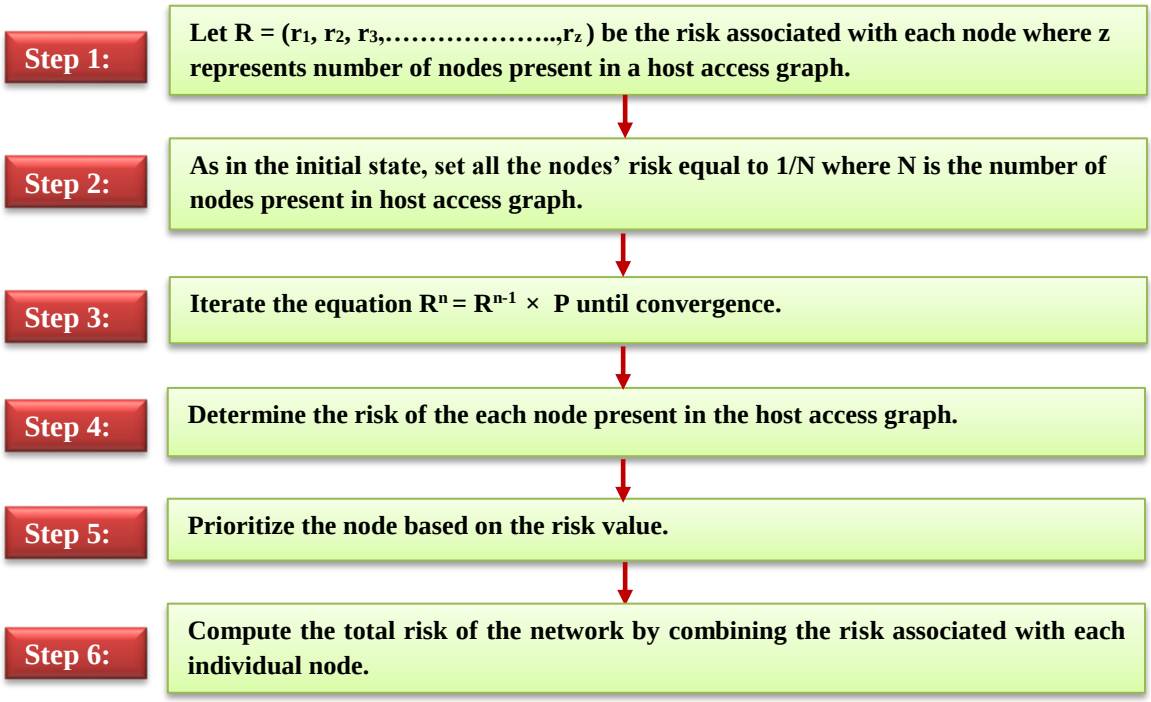

Figure 7.: Flow chart to compute risk of overall network

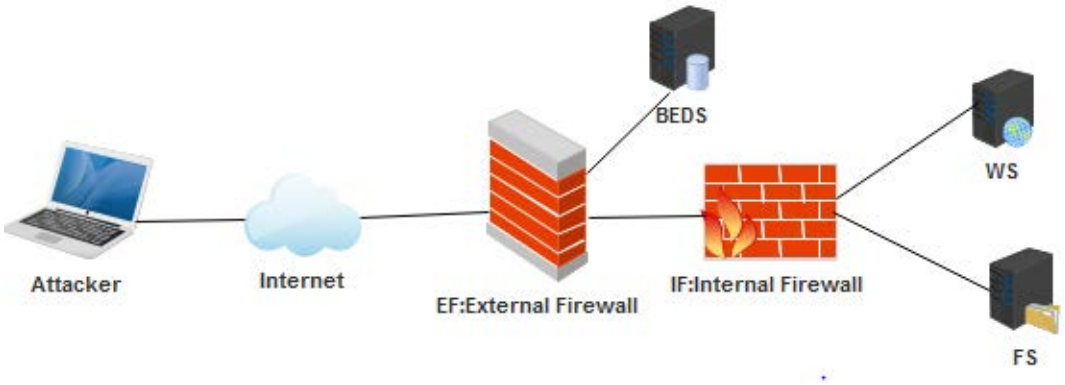

Figure 8.: Experimental topology

2.5 Network Environment:Illustration

To validate our proposed stochastic model, we have modified the network scenario [35, 36] to make it more realistic and practical as shown in Figure 8. In this network, there are three target hosts. These are publicly accessible **Web Server** (denoted by **WS**), a publicly accessible **File Server**(denoted by **FS**), and **Backend Database Server** (denoted by **BEDS**).

An attacker is located outside of the network. The packet transmission to the target host is controlled via two firewalls: **External Firewall(EF)** and **Internal Firewall(IF)**. EF allows any packet to be transmitted to WS and FS from outside of the network but no one can access the resources of BEDS from outside of the network directly. IF manages the transmission of the packet within the internal network.

The firewall rules are created to filter inbound and outbound traffic. A summary of firewall rules of the network scenario are shown in Table 1 below.

We have assumed that each of the target hosts consists of a single vulnerability. The attacker utilizes the vulnerability score to compromise the host. These are shown below by Table 2 along with its **Exploitability** sub-score and **Impact** sub-score taken from NVD.

Table 1: Firewall rules

Source	Destination	Service	Action
All	WS	http	Allow
All	WS	ftp	Allow
All	FS	ftp	Allow
WS	BEDS	oracle	Allow
FS	BEDS	ftp	Allow
All	All	All	Deny

Based on the experimental topology with its firewall rules and vulnerability associated with a respective host, we have generated a host access graph as shown below by Figure 9. To simplify the explanation, we have denoted the attacker, Web Server, File Server, and Backend Database Server as $\mathbf{M_0}$, $\mathbf{M_1}$, $\mathbf{M_2}$, and $\mathbf{M_3}$ respectively. The edges from all the nodes to the attacker node $\mathbf{M_0}$ are omitted to view the graph more clearly.

When the process illustrated by Equation 2.2 is applied on the host access graph of the experimental topology in the below Figure 9, we can obtain the weighted adjacency matrix as given by.

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{0} & \mathbf{8.2} & \mathbf{9.3} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{9.3} & \mathbf{8.2} \\ \mathbf{1} & \mathbf{8.2} & \mathbf{0} & \mathbf{8.2} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

Note that the diagonal elements of the above weighted adjacency matrix are all zero in a sense that practically no cost is involved to move from the current node to the same node. For the sake of simplicity, we have assumed the value of $\boldsymbol{\beta}$, the Biased factor to be 0.5. When the attacker stops attacking further due to any unusual circumstances, then it is certain that he/she will return to the initial node.

Table 2: Host vulnerabilities

Host	**Vulnerability**	**CVE-ID**	**Score**	**Impact Sub-Score**	**Exploitability Sub_Score**
WS	Apache Chunked Code	CVE-2002-0392	7.5	6.4	10
FS	Wuftpd Sockprintf	CVE-2003-1327	9.3	10	8.6
BEDS	Oracle Tns Listener	CVE-2012-1675	7.5	6.4	10

Hence, elements of the first column of the weighted adjacency matrix is 1, that is, weights of the edges from all host nodes to the attacker's node($\mathbf{M_0}$) is considered as 1, a sure event. The rest of all the elements of the weighted adjacency matrix are calculated using Equation 2.2. For example, the entry of the first row and second column is (0.5 $\times$ 10 + 0.5 $\times$ 6.4)= 8.2. This is the weighted value that the attacker uses to move from the node $\mathbf{M_0}$ to node $\mathbf{M_1}$. The same explanation is applicable for the rest of the elements of the weighted adjacency matrix. We have used the application software package "R" for all required calculations.

Once we have the weighted adjacency matrix A, then we need to convert its elements into respective probabilities; thus, it requires constructing a diagonal matrix. The entries of the main

diagonal are obtained by using Equation 2.5 as shown below.

$$D = \begin{bmatrix} \mathbf{0.05714} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0.05405} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0.05747} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

An element of the first row and the first column of the diagonal matrix is 1/(8.2+9.3) is equivalent to 0.05714. The same idea is used to compute the rest of all the elements. Utilizing weighted adjacency matrix and the diagonal matrix computed as shown above, we have obtained a transition matrix P below via Equation 2.4. Note that the extents of the first row second column is, 0.46857. It is the transition probability of the attacker moving from node $\mathbf{M_0}$ to node $\mathbf{M_1}$. As similar explanation is applicable to the rest of the entries of the transition matrix P as shown below. Now, the host access graph as shown by Figure 9 , consists of four nodes including the attacker's node. Based on our risk ranking algorithm explained in Figure 7, if we have four nodes then 1/4 = 0.25 is the initial risk of each node, hence the initial risk vector, R=(0.25, 0.25, 0.25, 0.25). Finally, when the initial risk(R) and transition probability(P) are iteratively multiplied using Equation 2.8, convergence is achieved by using the following values as listed in Table 3.

In Table 3, four nodes $\mathbf{M_0}$, $\mathbf{M_1}$, $\mathbf{M_2}$, and $\mathbf{M_3}$ are listed with their respective risk in terms of numerical value. $\mathbf{M_0}$ is the dummy node added to the host access graph to represent the attacker and is connected to every other nodes in the graph. An assumption of our ranking algorithm, initial

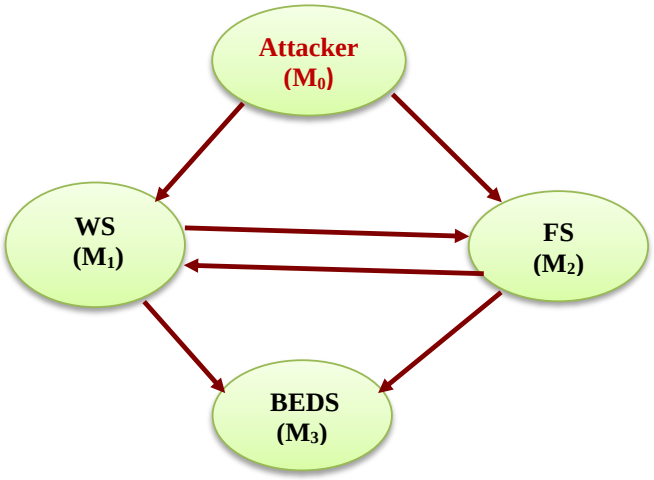

Figure 9.: Host access graph for experimental topology

risk of the all nodes are equal. Once the node is added to the graph and assigned the initial risk then it's final risk value is computed. As long as dummy node is considered we can ignore it's value from our analysis. Except $\mathbf{M_0}$, rest of the nodes are actual nodes present in our network.

$$P = \begin{bmatrix} \mathbf{0} & \mathbf{0.46857} & \mathbf{0.5314} & \mathbf{0} \\ \mathbf{0.0540} & \mathbf{0} & \mathbf{0.5027} & \mathbf{0.4432} \\ \mathbf{0.0575} & \mathbf{0.4712} & \mathbf{0} & \mathbf{0.4713} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

From the numerical value, we can conclude that node $\mathbf{M_2}$ is more risky than $\mathbf{M_1}$ and $\mathbf{M_3}$, so the vulnerability of the file server needs to be patched first before other nodes. The total sum of the risk associated with the node $\mathbf{M_1}$, $\mathbf{M_2}$, and $\mathbf{M_3}$ becomes 0.74. This value can be used as a security metric revealing the fact that this network is not very secure with respect to the given vulnerabilities and access relationship among the servers, and hence appropriate action must be implemented.

Table 3: Risk association with each node

Node	Risk
M_1	0.245
M_2	0.262
M_3	0.231

2.6 Contributions

In this chapter, we have developed a stochastic model to determine overall network security risk. The proposed model can be generalized to complicated network environment, however, the calculations are more complex but tractable. The proposed findings are based on a typical security system, however, the modeling aspect and scientific methodology is applicable to a specific security system that is in existence or it is required by a given enterprise. Our model findings can be useful for the system administrator to determine the critical nodes present in thenetwork where the attacker is most likely to visit. Here, we summarized our chapter contributions in terms of the following points:

1. We were able to develop analytic model to determne overall network security risk using Markovian process.

2. We have developed the risk ranking algorithm using a Markovian random walk to prioritoze the critical nodes based on quantative value.

3. The risk ranking algorithm that we implemented is very flexible in a sense that we can model the attacker in terms of skills and expertise by changing the Bias factor β in ExploitabilityBenefit computation.

4. Utilizing the outcome of the developed model network administrator can make the appropriate decision about system patching with priorities of their respective software.

5. Gaining in depth understanding of the risk and priority level of each host helps system administrator to implement strategic decisions like deployment of security products and to design network topologies.

6. The subject model can be generalized to be used for a larger, complicated network environment, however, the calculations are more complex but tractable.

Chapter 3

Time Series Predictive Modeling of Desktop Operating System Vulnerabilities

Unlike traditional ones, we propose a vulnerability analytic prediction model based on linear and non-linear approaches via time series analysis. We have developed the models based on Auto Regressive Moving Average (ARIMA), Artificial Neural Network (ANN), and Support Vector Machine (SVM) settings. The best model which provides the minimum error rate is selected for prediction of future vulnerabilities. Utilizing time series approach, this study has developed a predictive analytic model for three popular Desktop Operating Systems, namely, Windows 7, Mac OS X, and Linux Kernel by using their reported vulnerabilities on the National Vulnerability Database (NVD). Based on these reported vulnerabilities, we predict ahead their behavior so that the OS companies can make strategic and operational decisions like secure deployment of OS, facilitate backup provisioning, disaster recovery, diversity planning, maintenance scheduling, etc. One article is published [37] based on findings of this Chapter.

This chapter is organized as follows: Section 3.1 describes the overview of the subject area. Section 3.2 explains the literature review . Section 3.3, explores the dataset and explanation of major methods employed in our study. In Section 3.4 core analysis is presented. Finally in Section 3.5, we have pointed the list of the major contributions of this Chapter.

3.1 Introduction

A computer system is a collections of hardware and software components working together to perform a well defined objective as a unified whole entity. One of the core software component of the computer system is the Operating System (OS). An OS is a resource manager or a complex interactive software system. It enables the higher level application software to communicate with its hardware and memory. Vulnerabilities always exist on such software and causes tremendous security risks to software companies, developers, and individual users. Once an attacker compromised an Operating System via any vulnerability, this implies logically the whole computer system is in

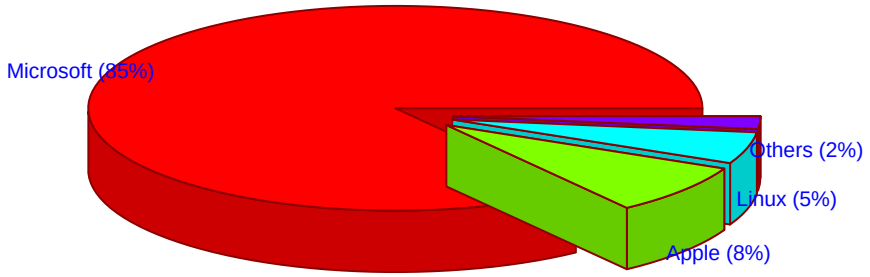

Figure 10.: Market share of desktop os based on netmarketshare data

control of the hacker. If the computer system itself is in control of unauthorized people, very significant consequences occur in tremendous financial loses, among other serious damages.

It is well known that the overall rate of the software vulnerabilities are extensively increasing [38]. According to the Secunia Vulnerability Review 2015, the number has increase to 55% in the past five years, and an 18% increase from 2013 to 2014 [39]. Similarly, Flexera Software reports that a 39% increase in the five year trend, and a 2% increase from 2014 to 2015 [40]. A Microsoft vulnerabilities study report in 2015 by Avecto Software Company reported that there is a significant uplift in the total number of vulnerabilities users are exposed to, rising 52% a year to year basis [41]. The documented facts and figures from the well established software institutions reveal the current increasing trend in number of software vulnerabilities that offer a significant problem to the industry. If these vulnerabilities are exploited and it is the objective of the hacker, we can realize a tremendous amount of damages and losses to software developers, government institutions, giant corporations, educational institutions, end users, and all possible stakeholders associated with this domain.

Some recent analytical study and modeling of general vulnerabilities can be found in [23, 42] . It is not possible to develop an OS software be free of vulnerabilities. On the contrary, we can have a precise estimation of vulnerabilities along with its trends, level, and seasonality based on the historical data. Once we have a better estimation of the number of vulnerabilities, as per our demand with respect to the calendar time, it would assist us to be well prepared to manage the forthcoming risks. At the same time, we can make diversity planning such as practical contingency plans, provisioning the backup capabilities, allocation of human and financial resources effectively and efficiently to achieve our mission, to be protected from the hackers.

Figure 10, gives a schematic view of the market share of Desktop OS worldwide, with Microsoft dominating the subject industry. In broad classification in-terms of the type of OS, two Operating

Systems exist in the market as described in Figure. A proprietary Operating System which in particularly conceptualizes, designs, and is sold by the specific company and do not share the source code to the public.

Microsoft and Apple are the two giant companies developing proprietary desktop Operating Systems. Similarly, Linux develops one of the non proprietary desktop Operating System referred as Linux kernel. According to Netmarketshare up to July 2016, [43] almost 85% of the market share of desktop Operating System is captured by the Microsoft company. Likewise, 8% of market share is captured by the Apple company and approximately 5% from Linux kernel which is graphically illustrated in Figure 10, above. To be more precise, out of 85% market coverage of all Microsoft's existing operating system, Windows 7 covers almost 48%. There is only one Operating System developed by the Apple company that is Mac OS X. On the other hand Linux develops Linux kernel and is considered as one of the oldest Operating System. This OS has minimum market coverage according to Netmarketshare. From the reported facts and popularity among the users if we aggregate the total market share of three desktop Operating System, they almost cover most of the market share in the Desktop Environment. Thus, it is appropriate to select Windows 7, Mac OS X, and Linux Kernel for our present study. These Operating Systems are the product of three industry leaders, Microsoft, Apple, and Linux.

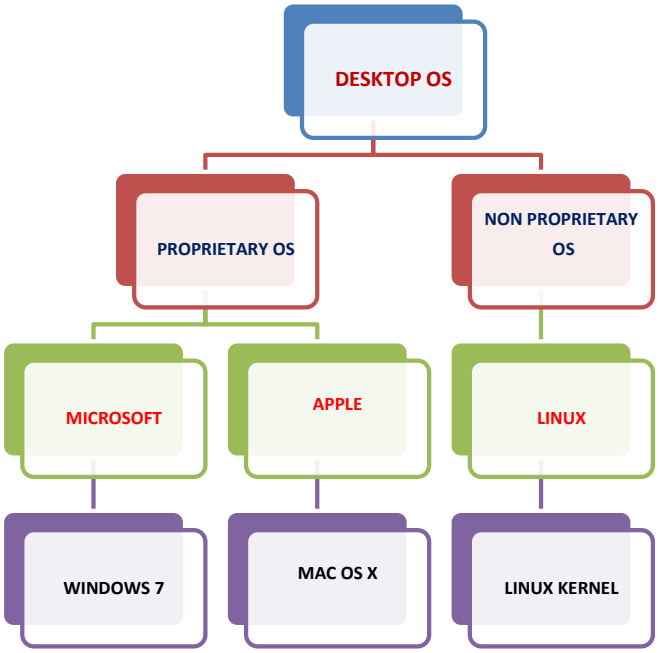

Figure 11.: Classification of desktop os

The schematic network of Desktop Operating Systems, given by Figure 11 above, displays a layout of the process that our analytic study will follow. In the present study, we have developed analytic vulnerability forecasting models using time series analysis via linear and non-linear approaches. The developed forecasting models completely capture the complicated linear and non-linear interrelationship between past data points and extrapolate those relationship into the future. We have implemented Autoregressive Integrated Moving Average (ARIMA) to incorporate the linear behavior of the signal in conjunction with trend, level and seasonality. To capture the non-linear characteristics of the signal, we are using Artificial Neural Network (ANN) and Support Vector Machine (SVM). Finally, we have compared the final outcomes of linear and non-linear models that best fit the actual data set. Based on the outcomes of the developed models, all the stakeholders associated with Operating System will find our predictive models of significant importance. As a software developer, one can evaluate and proceed to be confident with their strategic and operational policies. They can make the appropriate plans to allocate the human and financial resources effectively and efficiently. Moreover, they can make streamline patch decisions about OS and can utilize the outcomes for security testing procedure of the Operating System. Additionally, knowing the future vulnerabilities offer several benefit; one can identify the OS that are in need to be restricted to reduce its vulnerability, the predictive vulnerability score can be used for competitive market analysis, monitor the behavior of competing OS using the forecasted vulnerability etc. But most importantly this information is extremely important to the IT manager for his/her strategic planning to minimize the risk of chosen OS that will not be exploited. Finally, our results offer a unique marketing strategy for purchasing the best OS available in the market place that will have the best(smallest) future vulnerabilities.

3.2 Related Research

During the past years, scientists and researchers have given tremendous amounts of time and effort to develop vulnerability forecasting models to predict the future vulnerabilities of OS taking into consideration of their historical behavior with reported data. One can characterize these proposed developed models into two categories.

a) Code Characteristics Based Models: These types of models are relying on finding out the relationship between attributes of the code with its corresponding vulnerabilities. Rahimi and Zargham, [15] proposed a vulnerability scrying approach, that is, a vulnerability discovery prediction method based on code properties and quality. S. Riccardo et. al. [16] proposed a machine

learning approach to predict which components of the software applications contain security vulnerabilities using a text mining approach. This approach identified a series of vulnerable terms contained in its source code and was used to compute its frequency. Based on the frequency, they proceeded to forecast its future. Shin et. al. [17] performed an empirical study with traditional metrics of complexity, code churn, and faulty history using a large open source project to determine whether fault prediction models can be used for vulnerability prediction models . Nguyen and Tran, [18] proposed a component dependency graph to predict vulnerable components using machine learning methodology . All the mentioned approaches requires source code to built the models, source code of the OS is dynamic in nature and it is not available to public in case of proprietary OS.

b) Statistical Density Based Models: In this category, vulnerability forecasting models are based on historical data of the Operating System. To fulfill this objective, various kinds of models have been developed, mainly Alhazmi-Malaiya Logistic(AML) with different versions, [19] Poissons Log Arithmetic Model, [20], Rescorals Exponential Models [21], and Andresons Thermoodynamics Model [22]. All the developed models are based on their underlying assumptions and defined framework and none of them considers the non linear behavior of the signal.

For code characteristics based models, we need a source code of the given software to develop statistical models. In reality, source code of the commercial OS is not available to the public. Each and every day new vulnerability comes into existence and we need to handle new vulnerabilities, thus the software should be continuously updated which implies that the source code changes with respect to the life cycle of the software. A company always updates its software so as to fulfill the demands of current or potential customers and hence we need to update the source code regularly. Thus, processes of making source code is always dynamic in nature. One of the prominent question that arises here is how can we forecast the future vulnerabilities by utilizing static source code?. Knowing very well that no OS is with zero or no vulnerability at all and this will continue in the future. On the other hand, considering statistic density based models, that have been developed are based on a series of underlying assumptions and criteria that may or may not be applicable. For instance, Rescola Linear Model(RL) attempts to fit vulnerability finding rates linearly with time but in reality situations are different for nonlinear behaviours. Because of such limitations that exist on both categories, we should identify an alternative approach to forecast the future vulnerability of an OS by using time series analysis. Our model considers trend, level, and seasonality components if they exists. Similarly, to analyze the non-linear behavior of the number of vulnerabilities, we

implemented ANN and SVM methodology.

3.3 Data and Methodology

We have directly extracted the vulnerability data from the National Vulnerability Database(NVD). It is the U.S. governments repository that integrates publicly available vulnerability resources and provides the common references to the industry resources. NVD is a product of the National Institute of Standards and Technology (NIST), Computer Security Division and is sponsored by the Department of Homeland Security's National Cyber Security Division. It contains reported vulnerabilities based on their Common Vulnerabilities and Exposures (CVE) identifier. The total number of vulnerabilities with respect to time in monthly basis is the fundamental quantitative values for our analysis and modeling. The schematic diagram in Figure 12 describes the holistic idea about the datasets and methods employed in our study.

We have collected the vulnerabilities for each Operating System, the earliest available data from NVD to December 2015 as training data, however, the whole one year, 2016 data is considered as testing data to validate our model. We summed the total vulnerabilities over a monthly period. Linear and non-linear time series methods are implementd to select the best model with minimum forecasting error for each OS.

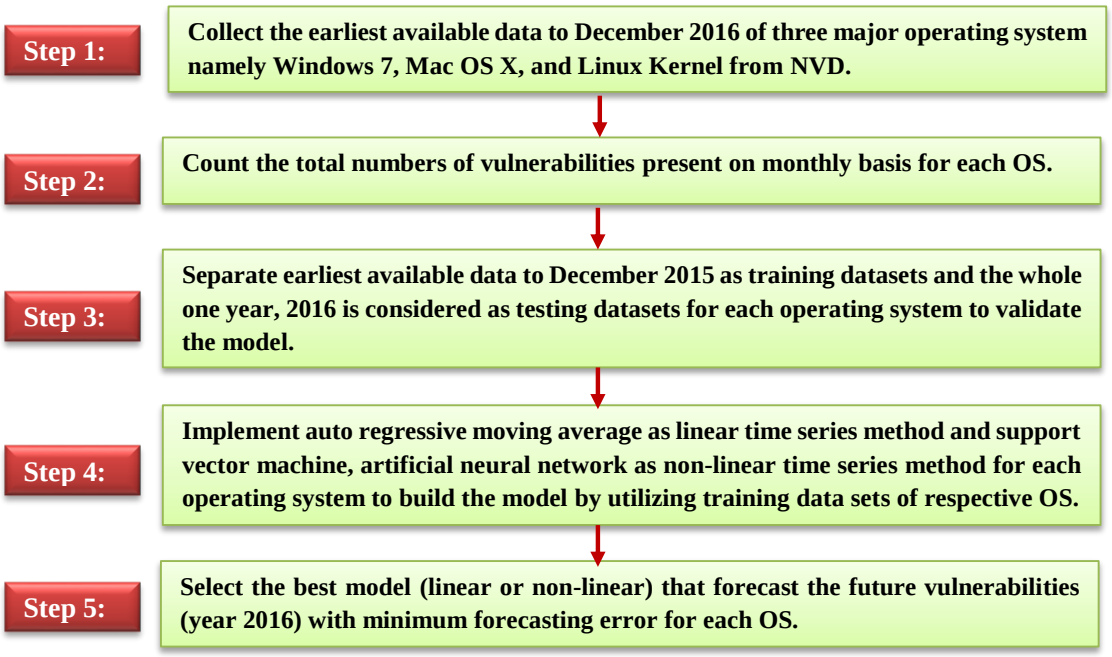

Figure 12.: Bird's eye view of data clollecton and method selection via flow chart

The following Table 4, shows the summary of descriptive analysis of the training data. It includes the total number of vulnerabilities on monthly basis, collection period, and monthly average. From the table below, we can conclude that average number of vulnerabilities was highest in Mac OS X followed by Linux kernel and Windows 7.

Table 4: Descriptive statistics of vulnerability datasets: mac os, windows 7, and linux kernel os

Operating System	**Collection Period**	**Total Vulnerabilities**	**Monthly Averages**
Mac OS X	Jan. 2002-Dec. 2015	1441	102.93
Windows 7	Jan. 2009-Dec. 2015	508	72.57
Linux Kernel	Jan. 2001-Dec. 2015	1292	86.13
Total	Jan. 2001-Dec. 2015	3241	261.68

Our analysis begins by investigating the total number of vulnerabilities accumulated by month for three OS, Figure 13, below reveals the overall variation in total number of vulnerabilities of Mac OS X. Inspecting Figure 13, the trend of the number of vulnerabilities of MAC OS X Operating System, it is clearly visible that initially the number of vulnerabilities are low and fairly stable, as a function of time. There is a high spike at the end of year 2015. In 2015 the number of vulnerabilities are almost four times greater than in previous years. One of the prominent reason is due to the rapid market share gains of Mac OS X which leads to growing attack surface for sensitive data. There are several malicious malware introduced in 2015, for instance, XcodeGhost which inserts malicious components in to the applications made with Xcode [44] (Apple's official tool for developing IOS and OS Apps).

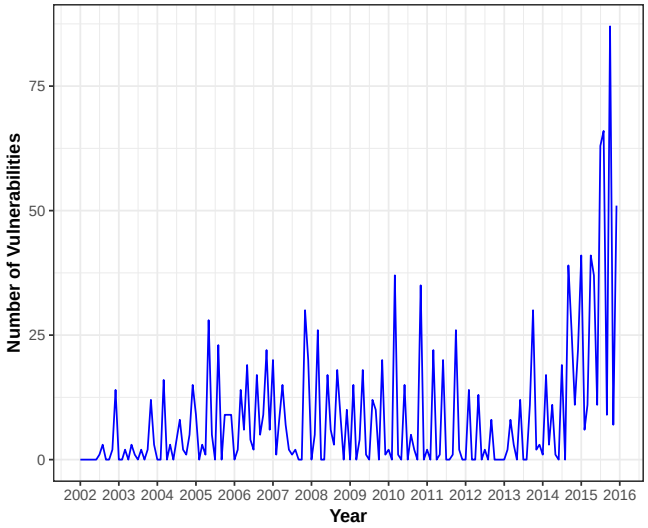

Figure 13.: Time series pattern of mac os x

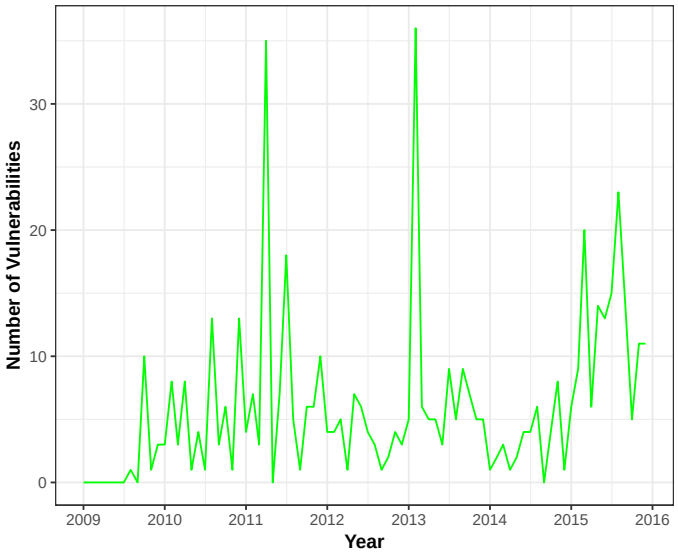

Figure 14.: Time series pattern of windows 7 os

Figure 14, shows the overall trend of the number of vulnerabilities of Windows 7 OS having a very nonlinear behavior. Initially number of vulnerabilities seems low but after a short period of time, we can see a significance increase and decrease of the number of vulnerabilities as a function of time.

There is a very sharp increase in the number of vulnerabilities in 2012, 2014 and 2015 for windows 7. "Secunia Vulnerability Review 2014", reported that the majority of the vulnerabilities on Windows 7 comes from the non-Microsoft software like Google Chrome, Adobe Flash Player and others rather than the major defect in OS itself. To incorporate the sharp random fluctuations of the number of vulnerabilities in each year, we initially believe that non linear time series methods is the suitable method to build the analytical forecasting model. If the IT manager had a good forecast of the large number of vulnerabilities, the subject of our study, he/she would have taken appropriate action to address this critical issue.

The overall trend of the number of vulnerabilities of Linux Kernel OS is demonstrated by Figure 15, from 2001 to 2015. Even though there is an increasing and decreasing trend, it would be fair to say that there is a significant variation in the number of vulnerabilities especially in 2014 and 2015, with more than double the number of vulnerabilities for the previous year. Years 2014 and 2015 were difficult years for Linux OS in terms of security perspective, for example "Heartbleed" is the severe vulnerability detected in OpenSSL that left large number of cryptographic keys and private data from important sites and services in the Internet that were open to the hackers. Similarly, Shellshock is the vulnerability that is dominantly used in Linux OS command line Shell, also called

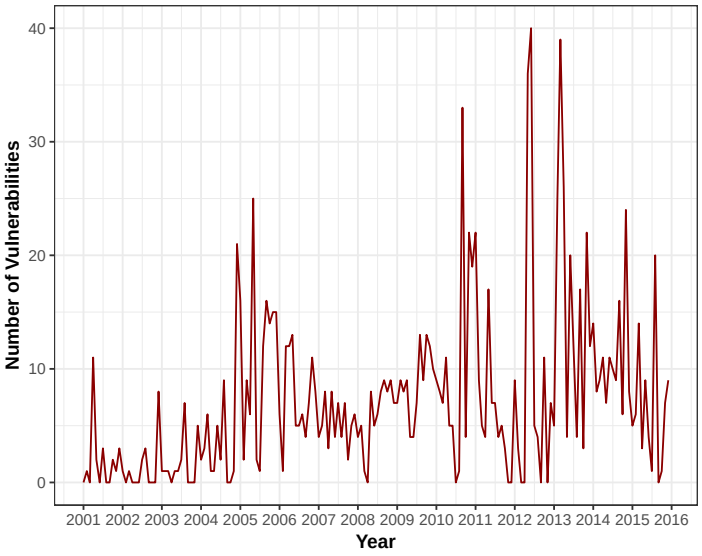

Figure 15.: Time series pattern of linux kernel os

Bash or GNU Bourne Again Shell left the door open for an hacker to lunch malicious attack.

All three graphs mentioned above, Figure, provides a pictorial view comparison of the number of vulnerabilities of the three OS, MAC OS X, Windows 7 OS, and Linux Kernel OS from respective calender time, on monthly basis. It does not seem obvious seasonality components exist but random fluctuations have a significant influence on each case. It is clear that each of the OS has some sort of increasing or decreasing trend for a specific period of time and all of sudden some spikes come and changes the behavior of the signal. To incorporate all the mentioned facts, we have employed linear and non linear techniques to build the best analytic forecasting model. The following section provides a brief explanation of the techniques employed in this study.

3.3.1 Autoregressive Integrated Moving Average Process (ARIMA) Model

Autoregressive Integrated Moving Average(ARIMA) models, commonly used for linear models for univariate time series analysis. To construct the ARIMA model to forecast the vulnerabilities requires three steps. Before going to the first step, it is necessary to check if the vulnerability data is stationary, this implies that the number of vulnerabilities associated with each operating system shows no trend over monthly observations. We have implemented Dicky-Fuller and Philips-Perron [45] unit root test to examine the stationarity of the ARIMA models. Whenever stationarity of the ARIMA models is established, the **first steps** to build the ARIMA model requires to identify the appropriate structure and order of the model. The ARIMA model includes autoregressive

terms(AR), moving average terms(MA), and differencing operations. An ARIMA model structure is represented by (p, d, q) where p is AR process, d is number of differences(filtering) and q is MA process. An AR(n) specifies number of immediately preceding vulnerabilities in the series that are used to forecast vulnerabilities at present. For example, AR(1) means the number of vulnerabilities at time t=1 rely on the numbers of vulnerabilities at t-1. Differencing term d is the degree of differencing (the number of times the vulnerability data have had past value subtracted) required to achieve the stationarity condition of the process. The MA(n) shows that present vulnerabilities have a relationship with past vulnerabilities, white noise error terms or random shocks. The random shocks are assumed to be independent and come form the same probability distribution. For example, MA(1), AR(1) means that the number of vulnerabilities at time t=1 relies on numbers of past prediction errors at t-1. The general equation of the model ARIMA (p, d, q) is given by:

$$\mathbf{y_t = c + (\alpha_1 y_{t-1} + ... + \alpha_p y_{t-p}) + (\beta_1 \varepsilon_{t-1} + ... + \beta_q \varepsilon_{t-q}) + \varepsilon_t} \qquad (3.1)$$

where,

$\boldsymbol{y_t}$ = differenced in series

c = a constant

$\boldsymbol{\alpha, \beta}$ =coefficients or weights

p = order of the AR term

q = order of MA term

$\boldsymbol{e_t}$ = residuals at time t

The **second step** is to construct the ARIMA model is to identify the number of parameters that are necessary to be included in the model which is a function of the order of the model. Furthermore we need to obtain estimates of the parameter that drive the model. We have implemented a graphical and statistical approach to find out the parameters used to forecast the vulnerabilities. For the graphical method, autocorrelation function (ACF) and partial autocorrelation function (PACF) is implemented. On the other hand, estimation of the required parameters requires complicated iteration procedure using maximum likelihood or non linear least square estimation methods. The **final step** of ARIMA is a diagnostic checking and forecasting vulnerabilities of the OS. The complete model fitting process is based on the law of principle of parsimony where the best possible model is the simplest with respect to accurately forecast the vulnerability of a given OS.

Utilizing the model building procedure of ARIMA model namely model formulation, model estimation, and model checking or model verification, we have developed three models for Mac OS X, Windows 7, and Linux as shown by the following equation 2, 3, and 4 respectively:

Mac OS X(ARIMA(1,1,3)):

$$\mathbf{y_t = 0.0203 - 0.8190y_{t-1} - 0.3626e_{t-1} - 0.8124e_{t-2} + 0.4432e_{t-3} + e_t} \tag{3.2}$$

Windows 7(ARIMA(2,1,1)):

$$\mathbf{y_t = 0.0197 - 0.1956y_{t-1} - 0.3350y_{t-2} - 0.8533e_{t-1} + e_t} \tag{3.3}$$

Linux Kernel(ARIMA(2,0,3)):

$$\begin{aligned} \boldsymbol{y_t} &= \mathbf{1.3367 + 0.0217}\boldsymbol{y_{t-1}} + \mathbf{0.7517}\boldsymbol{y_{t-2}} - \mathbf{0.0648}\boldsymbol{e_{t-1}} - \mathbf{0.6713}\boldsymbol{e_{t-1}} \\ &\quad + \mathbf{0.3317}\boldsymbol{e_{t-1}} + \boldsymbol{e_t} \end{aligned} \tag{3.4}$$

The abovementioned models are used to predict the future vulnerabilities via linear approach.

3.3.2 Artificial Neural Network

Artificial Neural Networks (ANN) is one of the useful and popular method, which have been used in forecasting using time series data. A wide variety of applications can be found in market predictions, meteorological and network traffic forecasting, [46, 47, 48, 49], where most of them have used feed-forward ANN models in a sliding window format over the input sequence. The major advantage of neural networks is that they are data driven and does not require restrictive assumptions about the form of the basic model.

$$\mathbf{y_t = f(w_{01}^{(2)} + w_{kj}^{(2)} \sum_{k=1}^{H} g(w_{0k}^{(1)} + \sum_{l=1}^{P} w_{lk}^{(1)} y_{t-l}))}, \tag{3.5}$$

Any feed forward ANN model consists of three or more layers called input, hidden, and output. The operational structure of the ANN model for the subject study are demonstrated below by Figure , and the final outcome of the ANN with one hidden node would be expressed analytically in Equation 3.5; where $\boldsymbol{y_t}$ is the total number of vulnerabilities reported in month t, p is the number

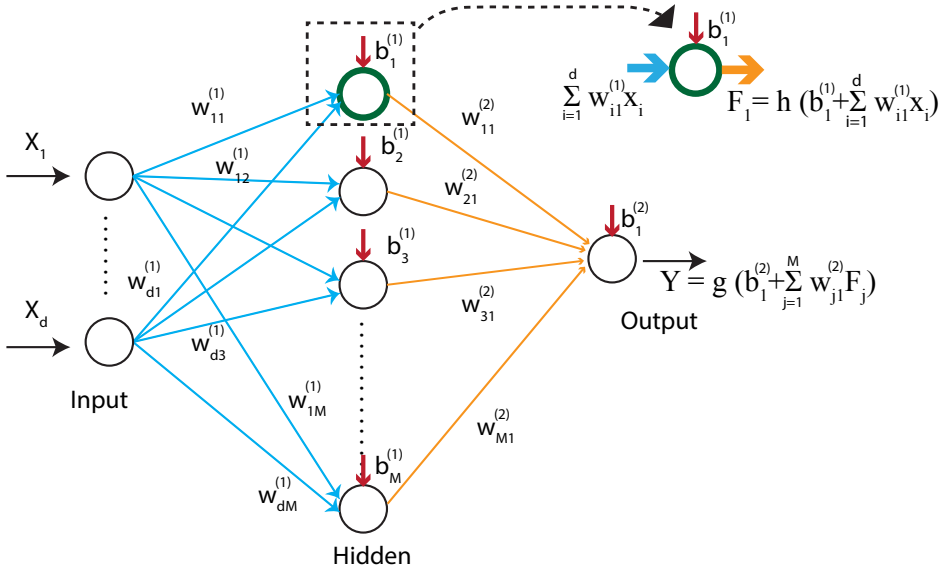

Figure 16.: The architecture of the ANN model used for os

of lags (number of vulnerabilities reported in the past p months) and the H is the number of hidden nodes, g and f are the activation functions associated with the hidden and the output nodes. In order to have a better generalization with the ANN model, we need to develop new procedures. Here, in our analysis, we have used different number of lags and select the model with minimum Mean Absolute Error. In addition to that we have used time series cross validation (forecast evaluation with a rolling origin) methods to identify the optimal number of hidden nodes, which refelects on the quality of the forecast of a given OS.

Figure 16, shows the basic architecture of an ANN and it represents a multivariate non-linear function mapping between a set of inputs and outputs variables (Bishop, 1995). These networks are organized as several interconnected layers. Each layer is a collection of artificial neurons (nodes) where the connections are governed by the corresponding weights. Data have been fed through the input layer, and then they pass through the one or more hidden layers, and the final outcome is given by the output layer. One of the challenges that we face when we use ANN in time series prediction in identifying the number of inputs which is not fixed. We used a procedure to identify the best possible number of lags.

3.3.3 Support Vector Machine (SVM)

Traditionally SVMs are used for classification in pattern recognition applications. These learning algorithms have also been applied to general regression analysis, the estimation of a function by

fitting a curve to a set of data points. The application of SVMs to general regression analysis case is called Support Vector Regression (SVR) and is vital for many of the time series prediction applications. SVMs used for time series prediction span many practical application areas from financial market prediction to electric utility load forecasting to medical and other scientific fields. One of the advantage in SVM is that it just correspond to a convex optimization problem when determining the model parameters and hence easily can be implemented. In using Support Vector (SV) regression, our goal is to find a function $\boldsymbol{f(x)}$ that has at most $\boldsymbol{\epsilon}$ deviation from the actually obtained targets $\boldsymbol{y_i}$ for all the training data, and will not accept any deviation larger than that. Anything beyond the specified $\boldsymbol{\epsilon}-$ will be penalized in proportion to $\boldsymbol{C}$, which is the regularization parameter. This can be explained with a linear function of the form

$$\boldsymbol{f(x) = w^T \phi(x) + b} \tag{3.6}$$

where our goal is to minimize

$$\boldsymbol{\frac{1}{2} w^T w + C \sum_{i=1}^{L} (\varepsilon_i + \varepsilon_i^*),} \tag{3.7}$$

with respect to the constraints

$$\boldsymbol{y(x_i) - f(x_i) \leq \epsilon + \varepsilon_i,}$$

$$\boldsymbol{f(x_i) - y(x_i) \leq \epsilon + \varepsilon_i^*,} \tag{3.8}$$

$$\boldsymbol{and \qquad \varepsilon_i^*, \varepsilon_i \geq 0}$$

The constant $\boldsymbol{C > 0}$, determines the trade-off between the flatness off and the amount up to which deviations larger than $\boldsymbol{\epsilon}$ are tolerated. Support vector machine can be generalized to deal with a nolinear function $\boldsymbol{f(x)}$, and minimize the weights with respect to the constraint

$$\min_{\boldsymbol{\alpha, \alpha^*}} \frac{1}{2} (\boldsymbol{\alpha} - \boldsymbol{\alpha^*})^{\mathrm{T}} \mathbf{Q} (\boldsymbol{\alpha} - \boldsymbol{\alpha^*}) + \boldsymbol{\epsilon} \sum_{\mathrm{i=1}}^{\mathrm{l}} (\boldsymbol{\alpha}_{\mathrm{i}} + \boldsymbol{\alpha}_{\mathrm{i}}^*) + \sum_{\mathrm{i=1}}^{\mathrm{l}} \mathrm{y_i} (\boldsymbol{\alpha}_{\mathrm{i}} - \boldsymbol{\alpha}_{\mathrm{i}}^*) \tag{3.9}$$

such that $\boldsymbol{0 \leq \alpha_i, \alpha_i^* \leq C}$, and $\sum_{i=1}^{l} (\boldsymbol{\alpha_i - \alpha_i^*}) = \boldsymbol{0}$ where, $\boldsymbol{\alpha_i, \alpha_i^*}$ are the Lagrange multipliers, Q is a l by l positive semidefinite matrix with, $\boldsymbol{Q_{ij} \equiv y_i y_j} \mathrm{K}(\boldsymbol{x_i, x_j})$ and $\mathrm{K}(\boldsymbol{x_i, x_j}) \equiv \boldsymbol{\phi}\ (\boldsymbol{x_i})^{\boldsymbol{T}} \boldsymbol{\phi(x_i)}$ is the kernel. However, in SVR we have no control on how many data vectors from the dataset become

support vectors and the correct choice of kernel parameters is crucial for obtaining desirable results [50]. Our objective is to conduct extensive analytical driven search procedure on the parameter space to obtain the optimal set of parameters that drive the model.

3.3.4 Analysis with ANN and ϵ SV regression

We began our analysis by dividing the vulnerability dataset into two groups; Training and Testing. The testing data set consists of vulnerabilities reported in year 2016. We then normalized the data by applying the min-max normalization method. Our analysis with ANN and SV regression makes the assumption that the number of future vulnerabilities depends on the vulnerabilities identified in the present and past months (lags). The number of significant lags in the partial auto correlation function has been used initially to determine the optimal number of lags. We proceeded by carring out further analysis by changing the number of lags from 2 to 10.

The radial basis functional kernel is used to develop the SV regression models with fine-tuning the two set of parameters; gamma and the regularization parameter C. In developing the ANN model we used 10-fold cross validation method for time series. When using this techniques we incremented the training sets data, gradually shifting the training data set window one by one. This was repeated for different number of hidden nodes. The optimal analytical model is selected based on the average mean absolute error (MAE). Finally, the selected analytical model is used to make the prediction in the testing data set.

3.4 Analysis

Our statistical analysis follows the process that we have introduced in Section 3, where we described the overall time series trend of each OS. We need further investigation on each signal to see if any trend, cycles, and seasonality exists. Usually time series data consist of a specific trend, cycles, and seasonality. To identify the best analytical forecasting model, we will first proceed to identify the time series pattern in the data, and then select an appropriate method that will capture the patterns effectively. Figure 17 and 18 is structured in two columns; which consist of six individual graphs. In Figure 17, 18, Fig(a), Fig(c), and Fig(e) descriebes the monthly behaviour of Mac OS X, Windows 7, and Linux Kernel OS respectively. These sub-series plots are inspected as a preliminary screening tool, that allow us for a visual inferences to be drawn from the data before proceeding to modeling and forecasting.

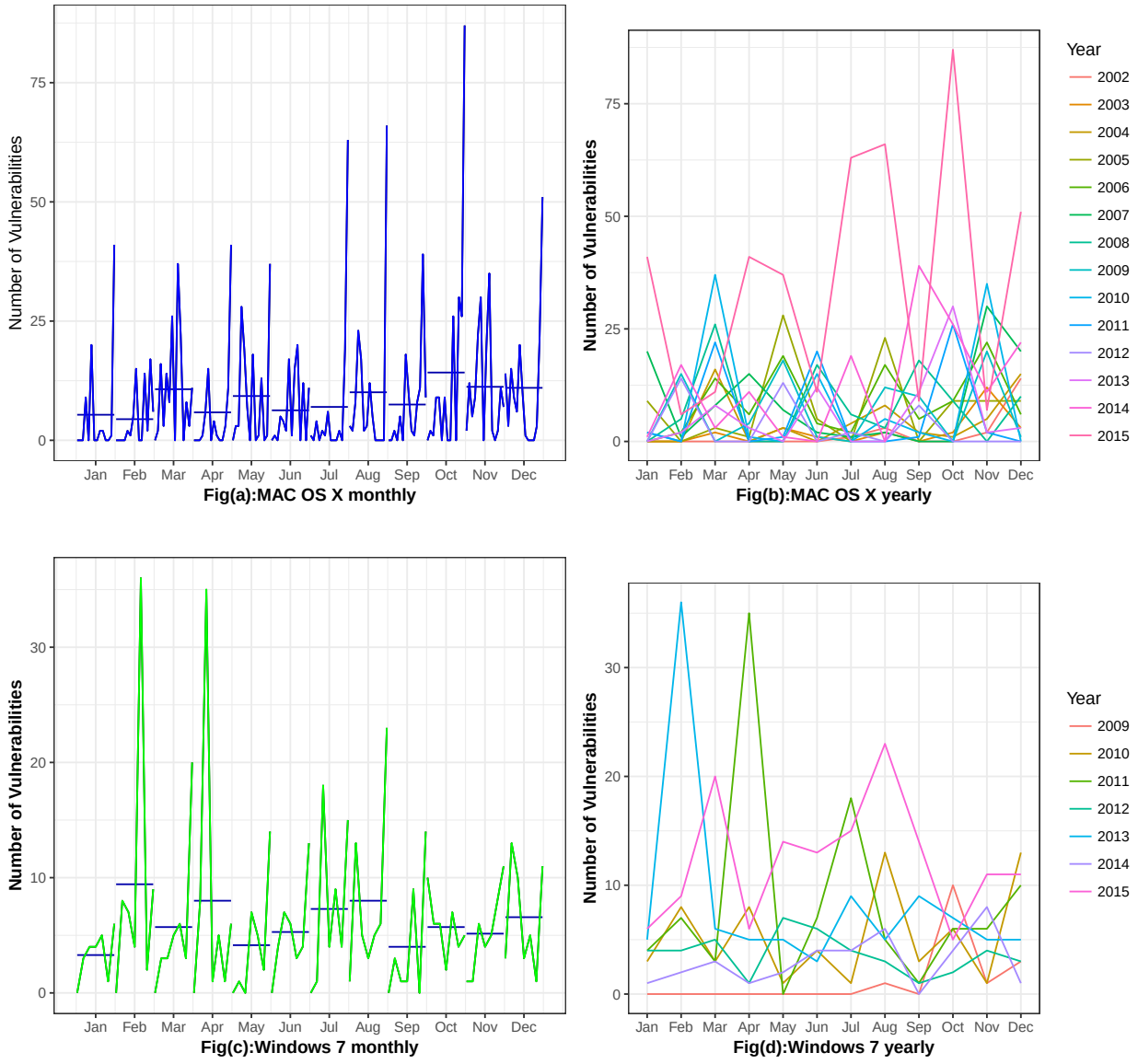

Figure 17.: Overall monthly deviation and yearly pattern of mac os x and windows 7 os

In Figure 17, 18 first columns' sub-series plots emphasize monthly patterns where the data for each month is collected together in separate mini time plots. The horizontal lines indicate the means for each month. This plot helps to find the monthly pattern over time. In each plot, none of the graph specifically are revealing any seasonal and cyclical pattern. Likewise there is not a significant variation in the means for each month.

In Figure 17, Fig(b) shows no seasonality or cyclic behavior is present over a year. It is clearly visible that there is sharp increase of vulnerabilities in year 2015 in comparison to the other year,

2002 to 2014. However, the Mac OS X system shows a significant random variation of the number of vulnerabilities. Similiarly, in Fig(d) each individual year shows random fluctuations of the number

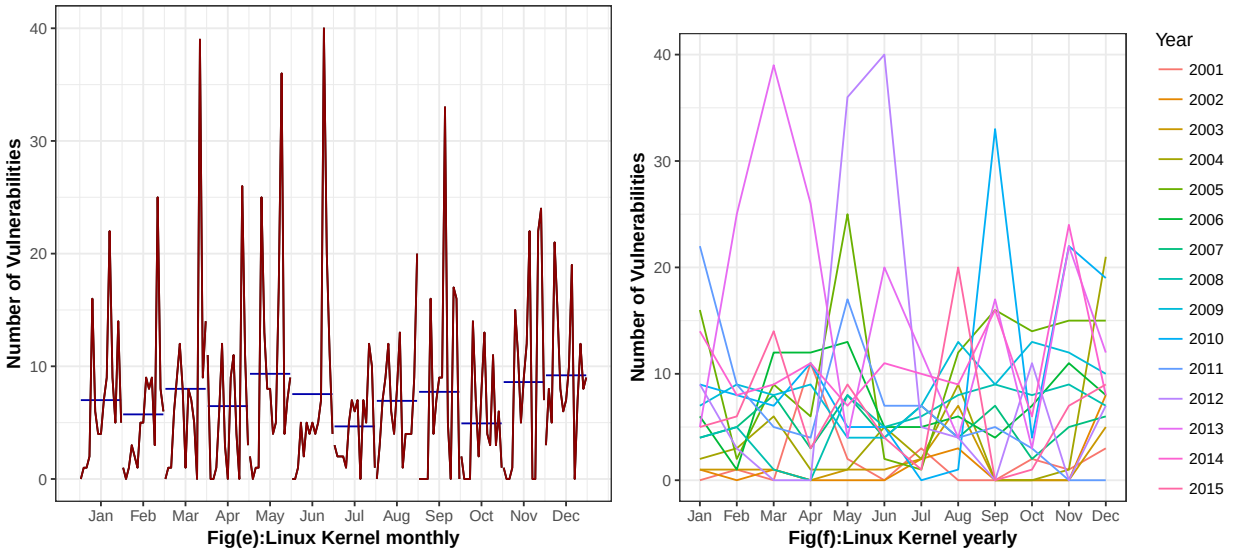

Figure 18.: Overall monthly deviation and yearly pattern of linux kernel os

of vulnerabilities over a twelve month period. We also concluded that there is no specific pattern of the behavior of the signals on a yearly basis of Windows 7 OS. Year 2009 and 2013 show the largest number of vulnerabilities followed by a significant reduction for year 2015 by a factor of 10. Lastly, Figure 18 of Fig(f), the signal of each individual year shows random variation of a number of vulnerabilities for a twelve month period. Year 2010, and 2013 show the largest number of vulnerabilities followed by a significant reduction for year 2010.

We plotted vulnerabilities against the individual months in which data are observed. Similarly, plots have been developed where data from each month is overlapped. These graphs allow us to make a decision that there is no specific seasonal or cyclical pattern seen in terms of monthly or yearly basis. We have found there is a large jump of vulnerability in specific years. The remaining years exhibit fluctuations on the number of vulnerabilities but no obvious seasonal or cyclic patterns. Inspecting the signal of the number of venerability in each OS, we have found that trend, level and random fluctuations are the major ingredients to build the forecasting model. Incorporating these facts, we have utilized ANN, SVM, and ARIMA models to forecast the future level of vulnerabilities for the three OS.

3.4.1 Predictive Capability of Models

One of the most important criteria for evaluating forecasting accuracy is to evaluate the error(residuals) generated by the testing data sets. An optimal model is selected based on how accurately it forecast our testing data sets. We have computed Root Mean Square Error(RMSE), Mean Absolute Error(MAE), and Symmetric Mean Absolute Percentage Error(SMAPE), for each model to assist in the selection process of the best model. For each forecast error estimation lower values are preferred.

Prediction accuracy of the analytic model is one of the most important criteria to evaluate the model performance and reliability. In addition to RMSE and MAE, we utilized an error analysis based on Symmetric Mean Absolute Percent Error(SMAPE) rather than Mean Absolute Percent Error(MAPE) to convinence the validity of our model. Even though SAMPE is based on MAPE, it does consider data containing zeros and non zero values that may skew the error rate. It consist of 0% of lower bound and 200% of upper bound, thus it reduces the impact of zeros and non zero values on our data sets. The error is computed based on the analytical form defined by the equation below:

$$\boldsymbol{SMAPE} = \frac{\mathbf{2}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} |\frac{\boldsymbol{P_i} - \boldsymbol{A_i}}{\boldsymbol{P_i} + \boldsymbol{A_i}}|, \tag{3.10}$$

where, N is the total number of prediction intervals, $\boldsymbol{P_i}$ is the predicted number of vulnerabilities, and $\boldsymbol{A_i}$ is the actual number of vulnerabilities. Once we employed ANN, SVM, and ARIMA model on our testing data set following optimal model are selected based on our error measurement criteria in Table 5.

Table 5: Output measurement criteria on testing data sets for each os

Criteria	**Mac OS X**			**Windows 7**			**Linux Kernel**		
	ARIMA	**ANN**	**SVM**	**ARIMA**	**ANN**	**SVM**	**ARIMA**	**ANN**	**SVM**
RMSE	19.645	28.563	24.674	21.597	9.55	3.581	22.900	4.080	3.990
MAE	16.174	22.060	19.925	21.272	8.911	3.150	24.200	3.410	3.280
SAMPE	0.312	1.225	0.950	0.992	1.284	0.124	1.570	0.730	0.410

Our ANN model evaluation results were quite good despite the fact that we did not have enough data to improve the training of our model. However, we believe that as more information of the subject matter becomes available the ANN model will be easier to implement and with higher accuracy in predicting the number of vulnerabilities of the present OS in the market place. For Windows 7 and Linux kernel is the analytical model, SVM driven by the final Equation 3.9. With

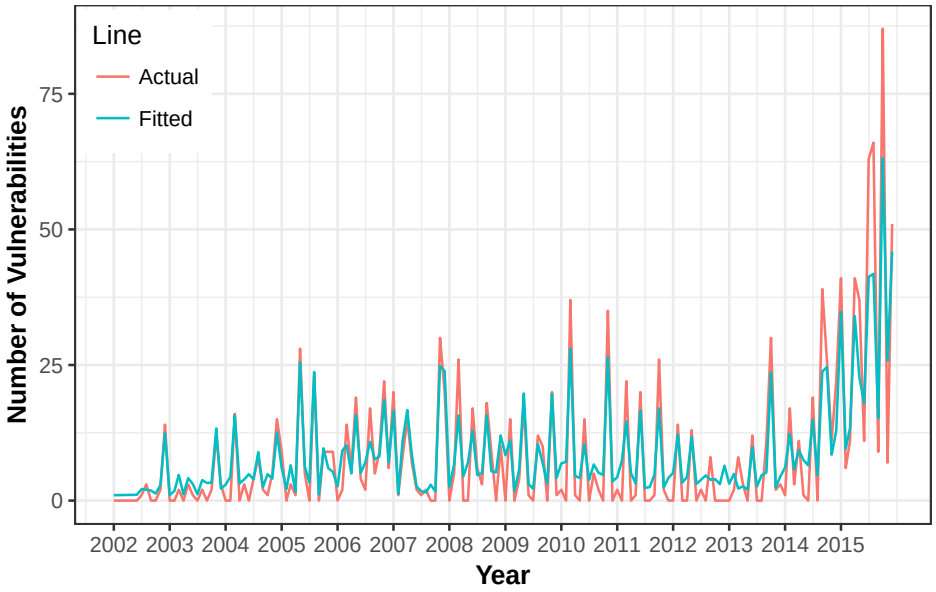

Figure 19.: Original vulnerability vs. fitted vulnerabilities for mac os x os

reference to the Table 5, the best models are selected for each OS based on the low error rate and the law of parsimony are listed in Table 6.

Table 6: List of best model selected for each os

OS	Best Model	SAMPE
Mac OS X	ARIMA	0.312
Windows 7	SVM	0.124
Linux Kernel	SVM	0.141

From Table 6 ARIMA(1,1,3) with drift, SVM with lag 5, and SVM with lag 5 models are selected for Mac OS X OS, Windows 7 OS, and Linux Kernel OS respectively. To be more specific, the forecasting model for Windows 7 OS had the lowest SMAPE of (12.45%) which implies it is a good forecasting model. The developed model provides good fit to the vulnerability data for Mac OS X and Linux Kernel but prediction accuracy is varied. In terms of forecasting, Linux kernel has a convincing SMAPE of (14.1%) but MAC OS X is reasonably accurate with a SMAPE (31.25 %). One possible reason for high percentage error may be due to missing components in our analysis such as OS development process, patch cycles, difference in security enforcement criteria, as well as market share and popularity of the OS.After the selection of the best model with minimum error rate, our study revealed the fact that the developed model provides a good fit for the OS datasets and can be used to forecast the future vulnerabilities. Fitting time series models to the

vulnerability database is demonstrated via the graph.

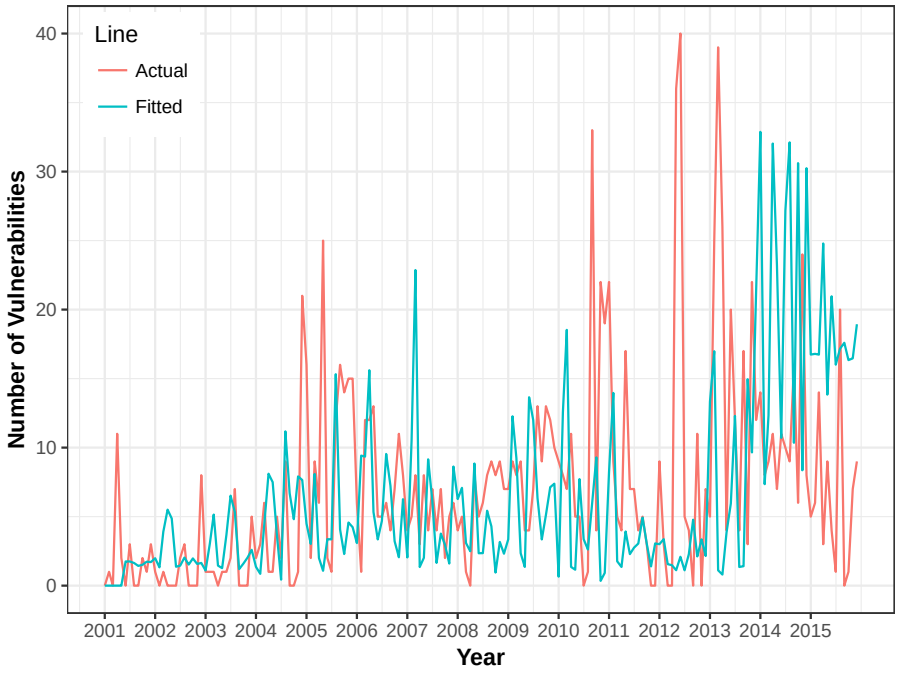

Figure 20.: Original vulnerability vs. fitted vulnerabilities for linux kernel os

Figure 19 shows the original vulnerability plot against fitted vulnerabilities of MAC OS X Operating System. Even though having random fluctuations that is increasing and decreasing behavior present in each year, our model accurately captures the holistic attributes of the signal with reasonable accuracy.

Figure 21, demonstrates results of fitted time series model and the actual vulnerabilities of the Windows 7 OS. Eventhough Windows 7 OS have limited amount of data comparing to other OS, it has lowest approximately 12% prediction error which attends to the high qualitiy of forecasting future vulnerabilities.

Figure 20 shows the fitting time series of the SVM model and the actual vulnerability data of Linux Kernel OS. Graphically our model shows a perfect fit with a prediction accuracy of 14% little bit higher than Windows 7 OS. This is probably due to very sharp increase of vulnerabilities in 2014 and a sudden decrease of in 2015.

All of the above plots provides a good fit for each OS but different degrees of prediction accuracy. From a careful reading of the fitted plots, we can conclude that best fitted model may not produce the best forecasting accuracy and vice versa. In case of Mac OS X, forecasted vulnerabilities is not that much better to the fit of data. Unlike Mac OS X, windows 7 has quite good fit but forecasted

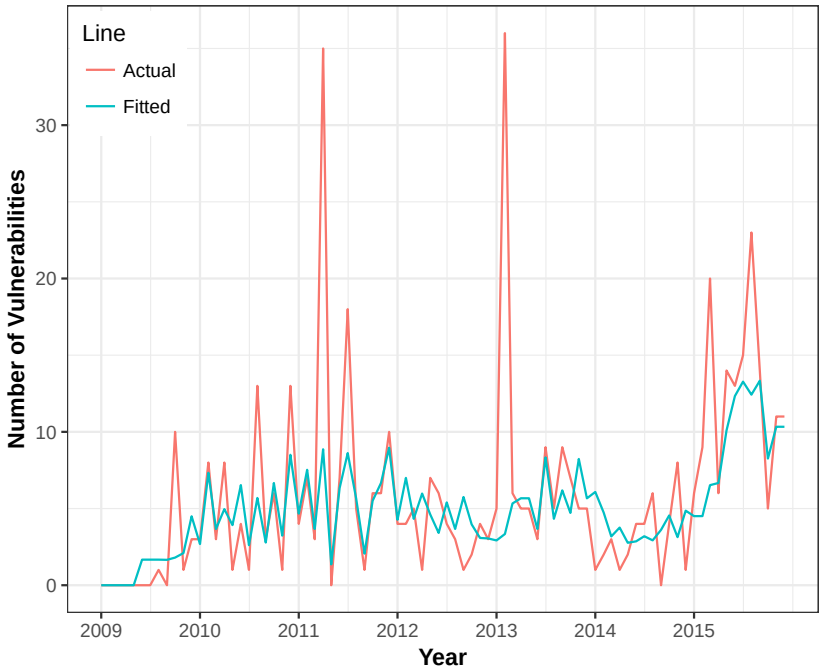

Figure 21.: Original vulnerability vs. fitted vulnerabilities for windows 7 os

vulnerabilities are a way better than Mac OS X. We eventually used our models to forecast the future vulnerabilities of these OS and recommended choice for predicting monthly vulnerabilities is summarized by Table 7. As an example, the following Table 7 highlights the fore-casted values for the 12 months of the year 2016. From the table we can say that predicted number of vulnerabilities for 2016 for Mac OS X is highest followed by Linux kernel and Windows 7.

Table 7: Forecasted vulnerabilities of mac os x, windows 7, and linux kernel os

OS	**Forecasted Vulnerabilities**												**Total**
	Jan.	**Feb.**	**Mar.**	**Apr.**	**May**	**Jun.**	**Jul.**	**Aug.**	**Sep.**	**Oct.**	**Nov.**	**Dec.**	
Mac OS X	28	24	32	26	32	34	31	26	27	36	38	37	371
Windows 7	11	10	9	11	9	11	11	10	10	11	11	13	127
Linux Kernel	7	16	7	18	34	8	7	24	7	7	12	18	165

Initially, we split our data sets in terms of training and testing data sets. The collection period of training data set of each OS is mentioned in Table 4. Similarly, 2016 is considered as testing data sets to validate our model for all OS. Utilizing our best developed model, we forecasted 2016 vulnerabilities which are given in Table 8. Now the following Table 8 compares the total true vulnerabilities and forecasted vulnerabilities for each OS. This comparison table shows the accuracy

and reliability of our developed analytical models.

Table 8: Actual and forecasted vulnerability comparison of the os

OS	2016 Vulnerabilities	
	Actual	Forecasted
Mac OS X	396	371
Windows 7	134	127
Linux Kernel	230	165

Our study revealed the fact that seasonality and trends are not the major components of the forecasting models. Nevertheless, the level of the time series is only the significant component to build the model. This suggests that it is difficult to predict vulnerabilities based on monthly seasonal patterns or trends. Further investigation is needed whether weekly, quarterly, or annual patterns might produce remarkable trends or seasonal components but such data is not publically availiable to improve the quality of the model.

The ANN model did not perform well in forecasting the vulnerabilities because we did not have enough data to improve the training process so as to improve its forecasting accuracy. With more vulnerability data we believe that the ANN model will be very competative in forecasting vulnerabilities of the OS.

3.5 Contributions

We have developed a high quality analytical forecasting model utilizing both linear and nonlinear methods to predict the number of vulnerabilities of a given operating system. In addition we performed a statistical evaluations of other models that perform the same task the predicting process of OS. The selected model provides overall trend and behaviour of the OS ahead of time; OS companies can make strategical and operational decisions such as secure deployment of OS, facilitate backup provisioning, disaster recovery, diversity plannng, and maintenance scheduling. In nutshell, we have summarized our chapter contributions in terms of the following points:

1. The best forecasting model helps to predict the number of vulnerabilities that may occur in the future for a given Operating System (OS) by utilizing linear and nonlinear method.

2. Operating System companies can make strategic and operational decisions like secure deployment of OS, facilitate backup provisioning, disaster recovery, diversity planning, and mainte-

nance scheduling.

3. Model helps to identify the prioritized patched decision of the Operating System.

4. Model help in accessing current security risk along with estimation of resources needed for handling potential security breaches and to foresee the future releases of security patches.

5. Overall trend and behavior of the OS ahead of time can be reported based on given level of vulnerabilities.

6. Outcome of the model helps to compare the OS.

Chapter 4
A Predictive Analytical Model for Vulnerability Discovery Process

In this chapter, we examine the existing models on the subject area and propose a new time-based nonlinear differential equation model. Our proposed model is based on the fact that the vulnerability saturation is a local phenomenon, and it possesses an increasing cyclic behavior within the software vulnerability life cycle. The daily vulnerability data is extracted from National Vulnerability Database (NVD) and is designed to obtain cumulative quarterly dataset. We apply the proposed model in cumulative quarterly vulnerability data for three Operating Systems: Mac OS X, Windows 7, and Linux Kernel. Our model performs significantly better when compared with the existing models in terms of fitting and prediction capabilities.

This chapter is organized as follows: Section 2.1 describes the our philosophy of vulnerability life cycle and existing models are presented. 4.2 introduces and explains our model entitled as Modeling Approach. Section 4.3 explores the application of our model in the realm of available datasets for three operating systems. The result section is further classified into three subsections: an application to vulnerability data, model selection and comparisons, and prediction. Finally, section 4.5 major contributions of this chapter are listed out.

4.1 Introduction

A software vulnerability can be defined as a loophole that allows an attacker to compromise the system. Normally, software is compromised with respect to its integrity, availability, or confidentiality. There is no software or Operating System (OS) without vulnerability and this scenario is most likely to continue in the foreseeable future. The existence of known vulnerabilities possess extremely high risk to all the stakeholders of the software. It may not be feasible to identify and rectify every vulnerability present in any software. The software developers and users need to estimate the level of risk produced by the given vulnerabilities and efficient counter measures need to be implemented. Developers need to stay ahead from the attackers by efficient allocation of resources and adopting ongoing vulnerability testing and software patch development procedures.

On the other hand, end users may invest in intrusion detection/prevention mechanism and different data safeguard techniques based on their requirements. The investment by developers and users depend on the level of risk posed by the vulnerability. Thus, one can argue that the investment is directly proportional to the level of vulnerability risk involved.

A plethora of research is conducted focusing on qualitative aspects of vulnerability. However, there is a need of developing statistical models that allow risks of the vulnerabilities to be evaluated quantitatively. Vulnerability models are required to assess the current security risk along with the estimation of resources required for handling potential security breaches. In addition, a robust model could potentially help to make an informed decision about future releases of software patches and evaluate the risk of vulnerability exploitation. A strong statistical data driven model that best fits the available data, and projects future vulnerabilities along with current and future trends based on historical data is the demand of the current time.

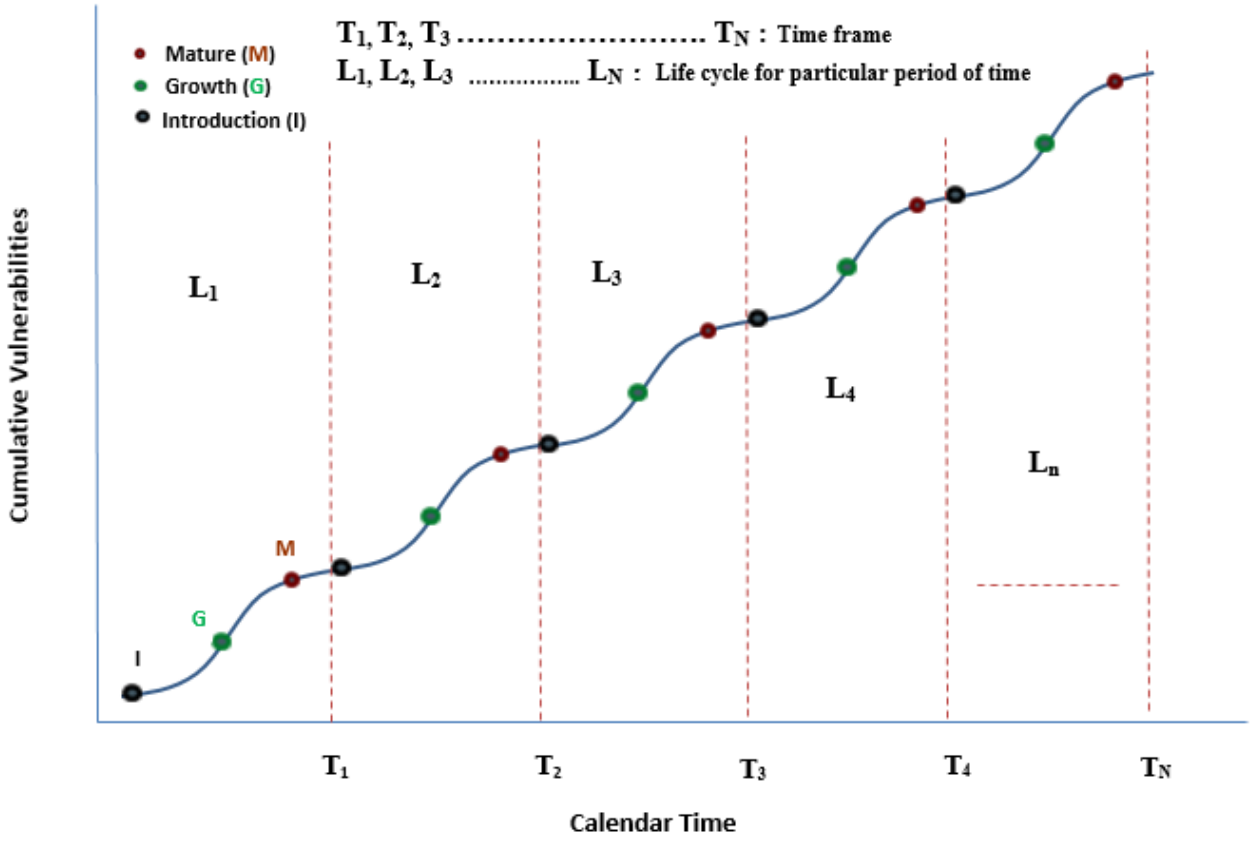

Figure 22.: Proposed vulnerability life cycle

Our current study of vulnerability gives us a different perspective to look at the transition phases of vulnerability life cycle as explained by Figure 22. It consist of different time frames ranges from

$t_1$to t_n where within each time frame, three transition phases (introduction, growth, and mature) exist locally and we believe this is a more realistic approach. The combination of three phases for the particular period of time generates partial life cycle. In real life, the transition phases exist locally but with the stability and continuity of the software, user base increases again and so does the vulnerability. All the partial life cycles from l_1 to l_n make a complete vulnerability life cycle. The combination of all the transition phases and time frames generates cyclic increasing behavior to explain the entire vulnerability life cycle.

The existing models shown in Figure 22 computed their model parameters by utilizing three transition phases of vulnerability life cycle due to simplicity of interpretation and analytical tractability. The study by Alhazmi and Malaiya [53], published in 2005, mentions that the Rescorla Linear Model for cumulative vulnerability given by

$$\Omega(t) = Bt^2 + Kt,$$

obtained from the vulnerability rate $\omega(t) = Bt + K$, where B is the slope, and K is a constant; both regression coefficients and the Rescorla Exponential Model given by

$$\Omega(t) = N(1 - e^{\lambda t}),$$

where N is the total number of vulnerabilities, and λ is the rate constant failed the goodness of fit test for Windows 95 cumulative vulnerability data. It was discovered that the models such as RL, RE,and AT failed the goodness of fit tests except the AML model. The AML model is given in [55] by

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

where (meaning of A,B,C.) This model assumes that the vulnerability discovery rate increases at the beginning, reaches a steady rate, and then starts to decline. This model is suitable only for one partial life cycle but when there is a cyclic increasing behavior of many partial life cycles, it fails to model the situation.

In the present study, we propose a new time based nonlinear differential equation model given by

$$\Omega''(t) + \omega^2\Omega(t) = f(t), \tag{4.1}$$

where $\Omega(t)$ is the cumulative vulnerability count at time t, and $f(t)$ is the quadratic forcing term.

A general solution of the differential equation 4.1 is given by

$$\Omega(t) = c_1 \cos(\omega t) + c_2 \sin(\omega t) + c_3 t^2 + c_4 t + c_5, \tag{4.2}$$

where c_1, c_2,...,c_5 are the coefficients that derives the model. The model 4.2 is considered as the final mathematical model, named as Pokhrel-Khanal-Tsokos differential equation model (PKT Model). We compare this model with the existing ones shown in Figure 23 in terms of fitting and prediction accuracy. The parameter estimation and their significance are discussed in a homogeneous manner using Nonlinear Regression methodologies. To support our arguments, we use the quarterly data, from National Vulnerability Database(NVD), of three Operating System(OS): Windows 7, Mac OS X, and Linux kernel.

The schematic network of desktop Operating System, given by Figure 23, which displays a layout of the process that our analytic study will follow. In broad classification in-terms of the type of OS, two Operating Systems exist in the market as described in Figure . A proprietary Operating System which in particularly conceptualizes, designs, and is sold by a private company does not share the source code to the public. Microsoft and Apple are the two giant companies developing proprietary desktop Operating System. Similarly, Linux develops one of the nonproprietary desktop Operating System referred as Linux kernel. We have collected the vulnerabilities for each Operating System, with the earliest available data from NVD to December 2015 as training data, however, the whole one year, 2016 data is considered as testing data to validate our proposed model. We summed the total vulnerabilities over a quarterly period.

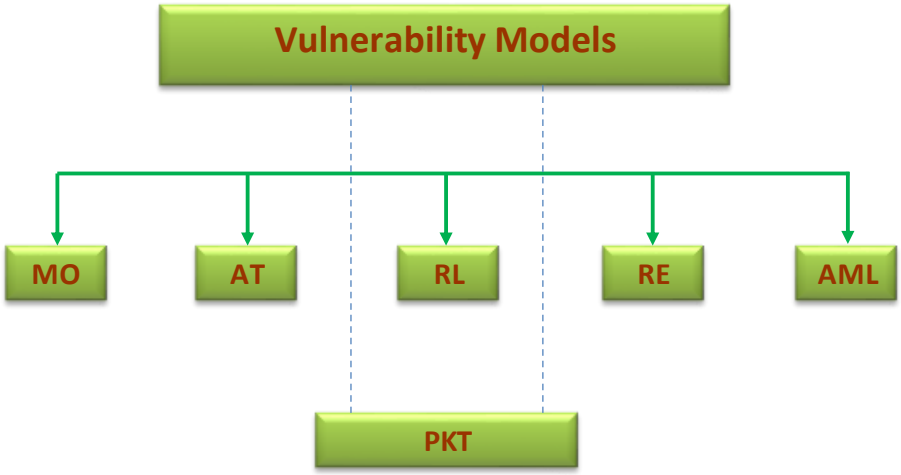

Figure 23.: Existing and proposed models

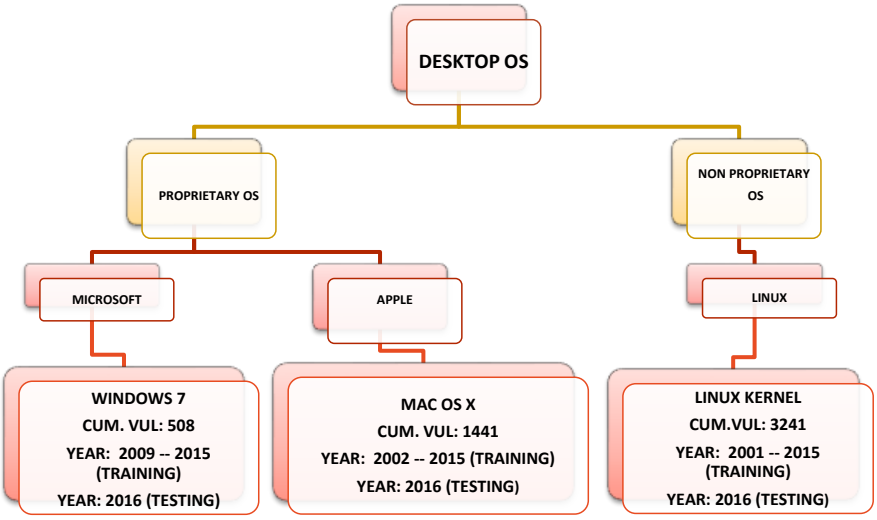

Figure 24.: Classification of os

Figure 24 gives a schematic view of the market share of Desktop OS worldwide, with Microsoft dominating the subject industry. According to Netmarketshare upto October 2017, [65] almost 90% of the market share of desktop Operating System is captured by the Microsoft company. Likewise, 4% of market share is captured by the Apple company and approximately 3% from Linux kernel which is graphically illustrated in Figure 25. To be more precise, out of 90% market coverage of all

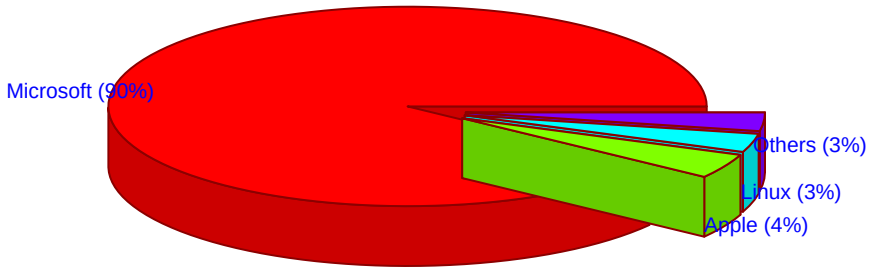

Figure 25.: Market share of os

Microsoft's existing Operating System, Windows 7 covers almost 48%. The only desktop Operating System developed by the Apple company is Mac OS X. On the other hand, Linux develops Linux kernel and is considered as one of the oldest Operating System. This OS has minimum market coverage according to Netmarketshare but still it is popular due to availability of source code to the public. From the reported facts and popularity among the users if we aggregate the total market share of three desktop Operating System, they almost cover most of the market share in the desktop environment. Thus, it is appropriate to select Windows 7, Mac OS X, and Linux Kernel for our present study. These Operating Systems are the product of three industry leaders, Microsoft,

Apple, and Linux. In Figure 26, we have produced six panels of displays. Each panel consists of two graphs for Mac, Linux, and Windows respectively. The graphs in (a), (c), (e) display the number of vulnerabilities versus time on monthly basis. The graphs show a completely nonlinear behavior with random fluctuations and irregular spikes. A recently published paper [66] uses linear (ARIMA) and nonlinear (ANN, SVM) time series models for monthly data and concludes that there is no influential trend or seasonality components. In the present study, we plan to develop a customized model that captures the nonlinear behaviors. The existing vulnerability models use cumulative number of vulnerabilities with time as a independent variable. As one of our goals in this research is to compare our model with the existing ones, for uniformity, we structurize our data in a cumulative quarterly basis.The graphs in (b), (d), and (f) display the scatter plots of quarterly vulnerability data for the corresponding OS.

Our proposed analytical model strongly captures the complicated linear and nonlinear behavior of the historically available data points and helps us to predict the future vulnerabilities. The proposed model is compared with the existing ones based on Akaike Information Criteria (AIC) and goodness of fit test for all operating systems. It is also found that the prediction accuracy is noticeably higher for the proposed new model. Based on the outcomes of the developed model, all the stakeholders associated with Operating System will find this new predictive model is of significantly important and useful. As a software developer, one can evaluate and proceed to be confident with their strategic and operational policies. They can make appropriate plans to allocate the human and financial resources effectively and efficiently. Moreover, they can also make streamline patch decisions about OS and can utilize the outcomes for security testing procedure of the Operating System. Additionally, knowing the future vulnerabilities offer several benefits; one can identify the OS that are in need to be restricted to reduce their vulnerability, the predictive vulnerability score can be used for competitive market analysis, monitor the behavior of competing OS using the forecasted vulnerability and accordingly take appropriate actions. Most importantly, this information is extremely relevant to the IT manager for his/her strategic planning to minimize the risk of a chosen OS that will not be exploited. Finally, our results offer a unique marketing strategy for purchasing the best OS available in the market place that will have the best(smallest) future vulnerabilities.

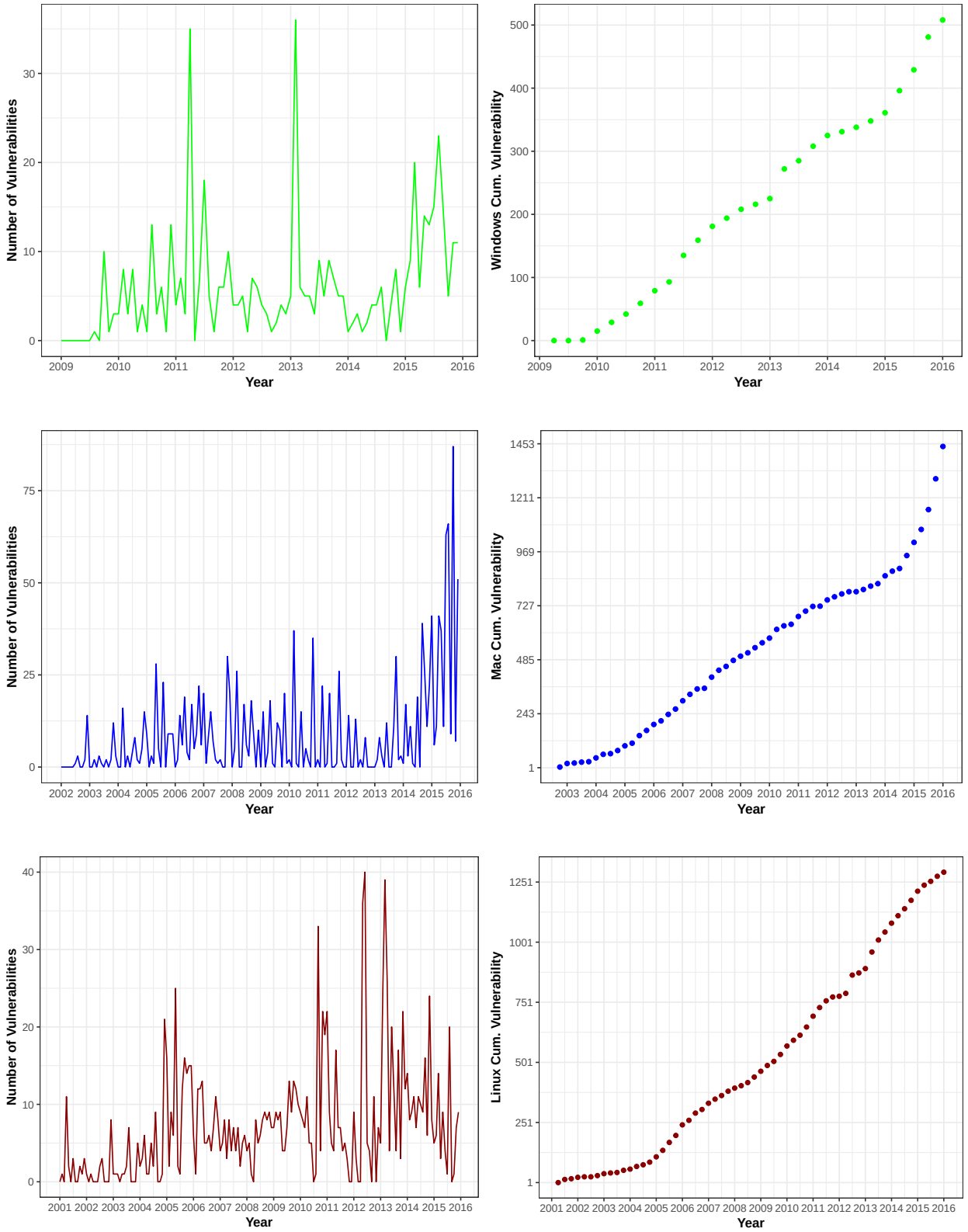

Figure 26.: The monthly time series and cumulative quarterly scatter plot for three OS

4.2 Modeling Approach

A number of vulnerability models have been proposed in the past to predict the number of vulnerabilities of a given OS: Musa-Okomoto Model, Anderson Thermodynamic Model, Rescorla Linear Model, Rescorla Exponential Model, and Alhazmi-Malaiya Logistic Model. Our aim in the present study is to develop a differential equation model approach that captures the rate of change of the total vulnerabilities discovered in three widely-used operating systems: Mac, Windows, and Linux. The existing models for software vulnerability have been developed based on some specific underlying assumptions and defined frameworks. In the present study, we plan to develop an analytic model that more accurately captures the dynamics of the total cumulative vulnerabilities of a given OS. We propose a new time-based nonlinear differential equation model and proceed to compare the results of our model with some of the existing models on the subject area, and include in our proposed model reasonably strong improvement in estimation and prediction performance.

A careful reading of the scatter plots given in Figure 26 for the three different operating systems do not support the claim that the vulnerability attains a saturation phase. For a particular time frame, the transition phases exist locally but with the stability and continuity of the software, user base increases again and so does the vulnerability. Thus, the combination of all these different transition phases and time frames make a cyclic increasing behavior within the life span of the vulnerability life cycle.

A careful study of data, suggests that there are two types of trends: one cyclic behavior and another steady increase. If $\boldsymbol{\Omega = \Omega(t)}$ is the cumulative vulnerability counts at time $\boldsymbol{t}$ then the vulnerability rate and its rate of change are given by,

$$\begin{aligned} \boldsymbol{\Omega'(t)} &\approx \frac{\boldsymbol{\Omega(t+h) - \Omega(t-h)}}{\boldsymbol{2h}} \quad \text{and} \\ \boldsymbol{\Omega''(t)} &\approx \frac{\boldsymbol{\Omega'(t+h) - \Omega'(t-h)}}{\boldsymbol{2h}}, \quad \text{where} \end{aligned}$$

$\boldsymbol{h}$ is the time step. The data set for $\boldsymbol{\Omega''(t) + \Omega(t)}$ portrays a very strong quadratic behavior. So, we propose a differential equation model of the form

$$\boldsymbol{\Omega''(t) + \omega^2 \Omega(t) = f(t),}$$

with a quadratic forcing term $\boldsymbol{f(t) = At^2 + Bt + C}$, where $\boldsymbol{\omega = \frac{2\pi}{T}}$, $\boldsymbol{T}$ being the period and the period depends upon the data set considered.

The complementary solution of the homogeneous part of equation (4.1) is

$$\boldsymbol{\Omega_c(t) = c_1 \cos(\omega t) + c_2 \sin(\omega t)}$$

and the particular solution is

$$\boldsymbol{\Omega_p(t) = c_3 t^2 + c_4 t + c_5}$$

where $\boldsymbol{c_3 = \frac{A}{\omega^2}, c_4 = \frac{B}{\omega^2}}$, and $\boldsymbol{c_5 = \frac{C}{\omega^2} - \frac{2A}{\omega^4}}$. Therefore, the general solution of the differential equation (4.1) is given by

$$\boldsymbol{\Omega(t) = c_1 \cos(\omega t) + c_2 \sin(\omega t) + c_3\, t^2 + c_4\, t + c_5.}$$

We can use an appropriate method based on the given data to estimate the values of the coefficients $\boldsymbol{c_1}$, $\boldsymbol{c_2}$, $\boldsymbol{c_3}$, $\boldsymbol{c_4}$, and $\boldsymbol{c_5}$.

The solution of the differential equation (4.1) is given by the equation (4.2) and it is our proposed analytical model, named as Pokhrel-Khanal-Tsokos Differential Equation Model (PKT Model). PKT analytical model can be used for estimating and predicting the total cumulative software vulnerability. The proposed model is applied to the available data for three operating systems: Mac OS X, Windows 7, and Linux Kernel.

4.3 Results

This section is divided into three subsections. The first subsection discusses the application of the proposed model to the vulnerability data for all three major operating systems: Mac OS X, Windows 7, and Linux Kernel. The development of the model together with fitting the model to the data with 95% confidence band will be discussed in detail. The second subsection is dedicated to the model validation and comparison with other existing models. The third subsection is devoted for the prediction accuracy of PKT analytic model using available vulnerability data of all three major operating systems.

4.3.1 An Application to Vulnerability Data

We have extracted the vulnerability data from the National Vulnerability Database (NVD). It is the U.S. governments' repository that integrates publicly available vulnerability resources and provides common references to the industry resources. NVD is a product of the National Institute of

Standards and Technology (NIST), Computer Security Division, and is sponsored by the Department of Homeland Security's National Cyber Security Division. It contains reported vulnerabilities based on their Common Vulnerabilities and Exposures (CVE) identifier. Each CVE is assigned a quantitative score to identify the severity level of the vulnerability that ranges from 0 to 10. We have collected the vulnerabilities for three Operating System namely Mac OS X, Linux Kernel, and Windows 7; from the earliest reported date to December 2015. More specifically, collection period of Mac OS X starts from 2002, Linux Kernel starts from 2001, and Windows 7 starts from 2009. According to the published date of CVE, we find quarterly sum of vulnerability counts. The vulnerability data of four quarters of 2016 is used as testing data to validate our analytic model.

In the proposed model

$$\boldsymbol{\Omega(t) = c_1 \cos(\omega t) + c_2 \sin(\omega t) + c_3 t^2 + c_4 t + c_5}$$

given in equation (4.3), $\boldsymbol{\omega = \frac{2\pi}{T}}$ depends upon the time period $\boldsymbol{T}$ and $\boldsymbol{T}$ depends on the data set. In our case for the operating system MAC, there are 54 quarterly data points. For one cycle, $\boldsymbol{T = 54}$ and $\boldsymbol{\omega = \frac{2\pi}{54} \approx 0.116355283466}$; for one and half cycle, $\boldsymbol{T = 36}$ and $\boldsymbol{\omega = \frac{2\pi}{36} \approx 0.174532925199}$; for two cycles, $\boldsymbol{T = 27}$ and $\boldsymbol{\omega = \frac{2\pi}{27} \approx 0.232710566933}$ and so on. For Mac OS X, we consider just one cycle for simplicity and use $\boldsymbol{\omega \approx 0.116355283466}$ in our model.

One of the most important issues is to estimate the five integral coefficient $\boldsymbol{c_1, \cdots, c_5}$ using nonlinear approach. Like in linear regression, nonlinear regression provides estimated values of the coefficients based on the least square criterion. However, unlike linear regression, no explicit mathematical solution is available and specific algorithms are needed to solve the minimization problem, involving iterative numerical approximations [57]. The estimated parameters are obtained by minimizing the residual sum of squares (RSS) with respect to the unknown parameters. There are no explicit formula and procedure to estimate the unknown parameters of the model. We used a recursive method to estimate the parameters that best fits the given data. The iterative procedure demands a starting values. To determine the best starting values and to avoid asymptotic behavior, we run the model using linear time series approach by using total number of quarterly vulnerability as response and time as a predictor variable. We set the starting value of the nonlinear regression from the estimated value obtained from linear time series method. We have achieved the convergence within two iteration in all cases.

By Implementing the computational procedure mentioned above, the estimated values of the pa-

rameters are: c_1=26656.79, c_2=32220.98, c_3=272.47, c_4=-4033.11, and c_5=-26376.42. Therefore, the model that provides the cumulative vulnerability behavior at any time t for Mac OS X is given by

$$\Omega(t) = 26656.79\cos(0.12t) + 32220.98\sin(0.12t) + 272.47t^2 - 4033.11t - 26376.42. \tag{4.3}$$

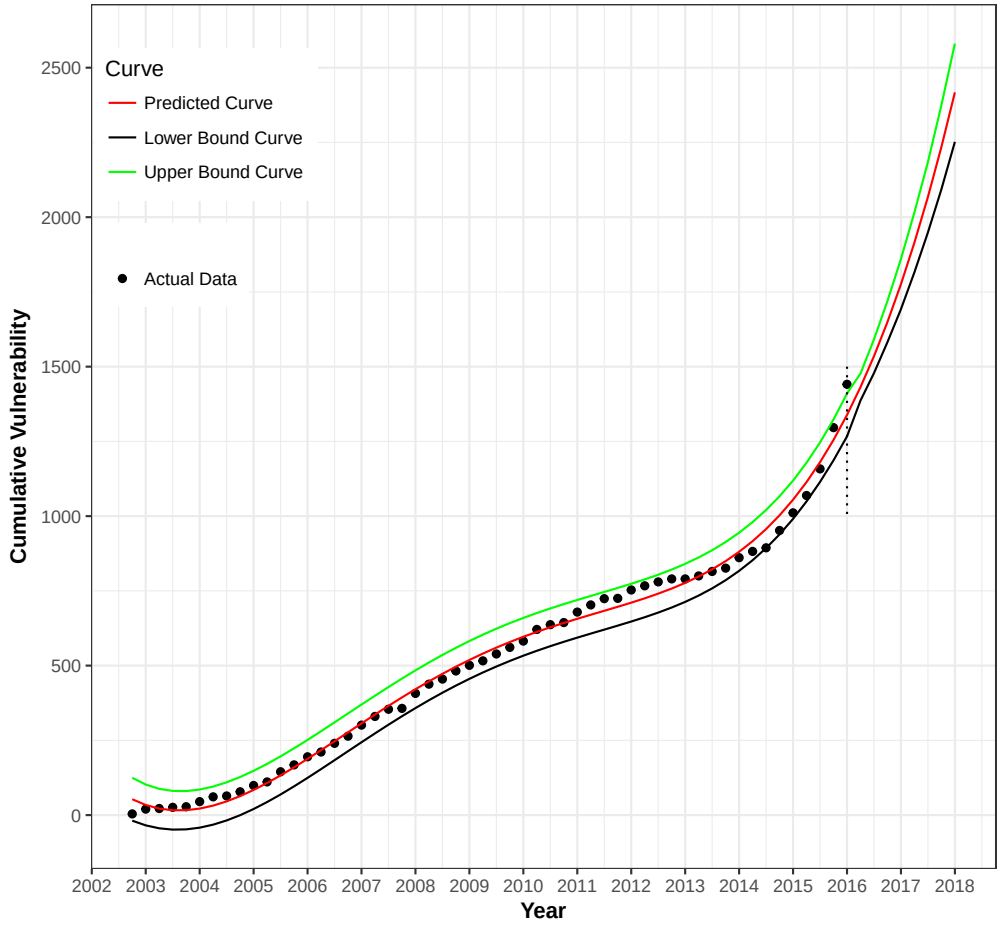

Figure 27.: Prediction with a 95% confidence band of mac os x: PKT

Figure 27 exhibits predicted 95% confidence band values of cumulative vulnerabilities given by the PKT analytical model (4.3) along with the actual data points. The solid red curve represents the actual and predicted values by the proposed model. The black and green lines represent lower and upper 95% confidence of the true values. Figure shows a good fit within the given confidence limits except one point in the fourth quarter of 2015. The total number of vulnerabilities in 2015

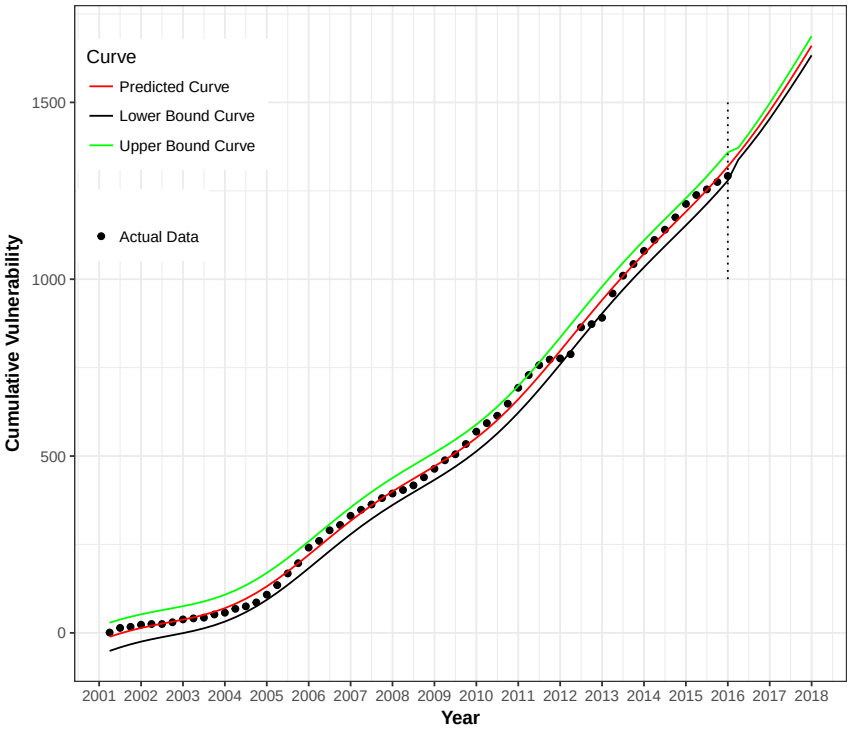

Figure 28.: Prediction with a 95% confidence band of linux kernel: PKT

increases almost four times larger than the previous year, which is an extreme observation compared to the rest of the data set. One of the prominent reasons is due to the rapid market share gains of Mac OS X which leads to growing attack surface for sensitive data. There are several malicious malware introduced in 2015, for instance, XcodeGhost which inserts malicious components into the applications made with Xcode (Apple's official tool for developing IOS and OS Apps)[64]. Thus, using Figure 27, for the last quarter of the year 2016, we predict the cumulative vulnerability value to be 1775. Furthermore, we are 95% confident that the true confidence level of Mac OS X will be greater than 1693 and less than 1860.

The fitted curve shows an increasing trend with increasing rate of cumulative vulnerabilities until 2008 followed by a slow growth until 2013 for Linux Kernel using the PKT model. The overall trend sharply increases from 2014 onwards.The dashed vertical line after 2015 displays the future prediction for the next two years. The AML model does not seem to fit the data because it demands stability for the entire vulnerability life cycle and the reality does not reveal such a fact. RL and RE models are better suited to capture the trend than AML model after 2014 but PKT model addresses these issues with excellent results.

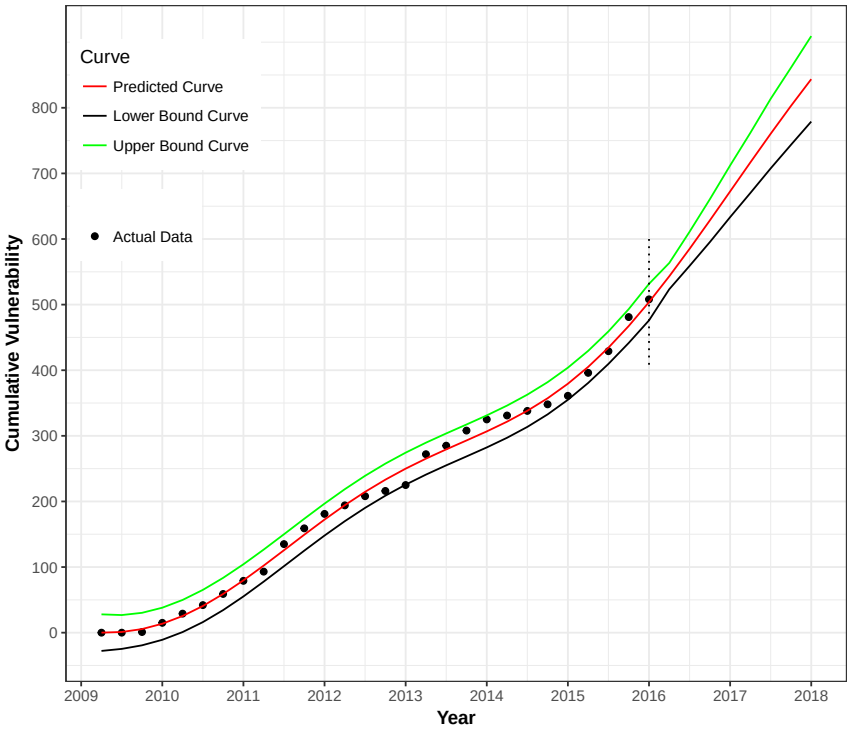

Figure 29.: Prediction with a 95% confidence band of windows 7: PKT

The PKT model is also applied to develop the nonlinear models for Linux Kernel and Windows 7 OS. The final model for Linux and Windows 7 are given by the following equations:

$$\Omega(t) = 1.13\cos(1.05t) + 23.55\sin(1.05t) + 3.97t^2 + 24.57t - 71.04,$$

$$\Omega(t) = 18.94\cos(1.15t) - 27.79\sin(1.15t) + 7.57t^2 + 4.54t + 7.57.$$

The fitted values given by the above models together with cumulative vulnerability data and 95% confidence and prediction band are given in the Figures 28 and 29. Thus, using Figure 28 and 29, for the last quarter of 2016, we predict that the cumulative vulnerability value to be 1475 and 672 respectively. Furthermore, we are 95% confident that the true confidence level of Linux Kernel will be greater than 1453 and less than 1497 and the true confidence level of Windows 7 will be greater than 633 and less than 711.

4.3.2 Model Selection and Comparisons

Here, we proceed to compare PKT model with the other existing vulnerability discovery models, namely RL, RE, and AML.The same techniques of evaluating estimation of these parameters for PKT model are used to evaluate the parameters of the other models. The comparison is based on Sum of Squares (RSS) and Akaike Information Criteria(AIC) which is defined by:

$$\boldsymbol{AIC = (-2 \times logLik) + 2P,}$$

Where, logLik represents log-likelihood value and P is the number of parameters in the fitted model. The maximum likelihood estimates are used to calculate the weights of AIC. Lower AIC value indicates a better fit of the data. Table 9 lists RSS and AIC values for RL, RE, AML, and PKT models for all three operating systems. In each case, PKT model depicts lower RSS and AIC values.

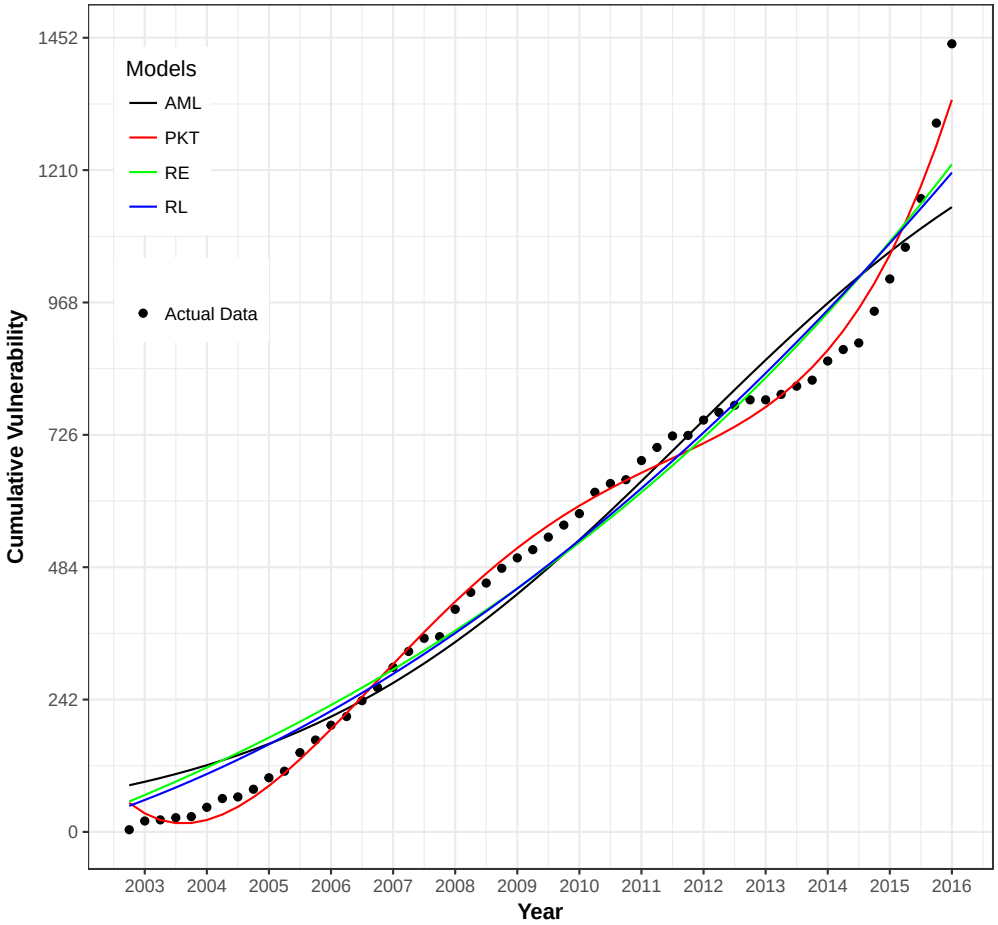

Figure 30.: Model comparison of PKT with RL, RE, and AML for mac os x

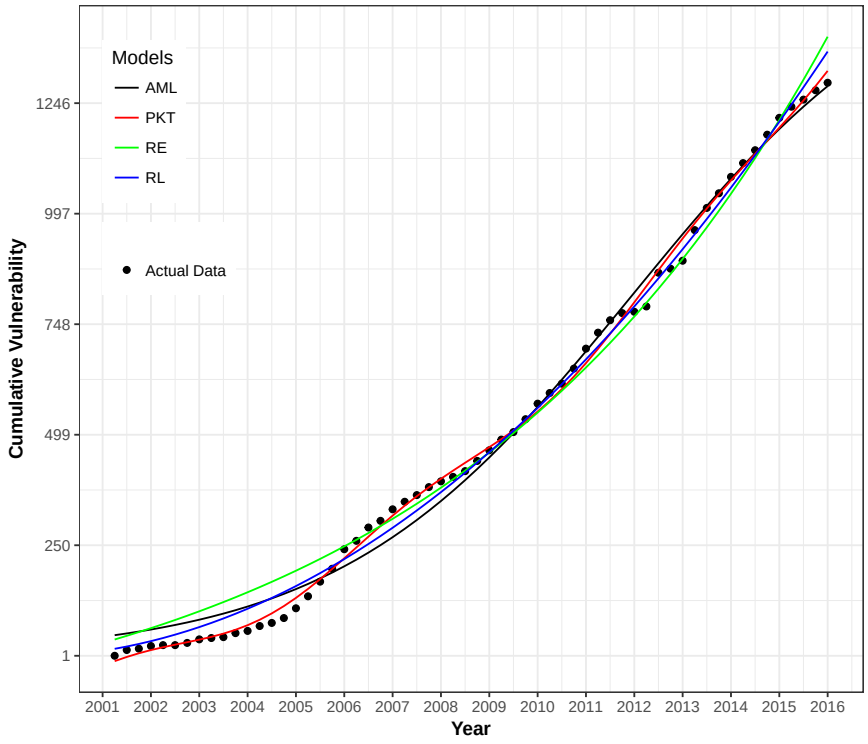

Figure 31.: Model comparison of PKT with RL, RE, and AML for linux kernel

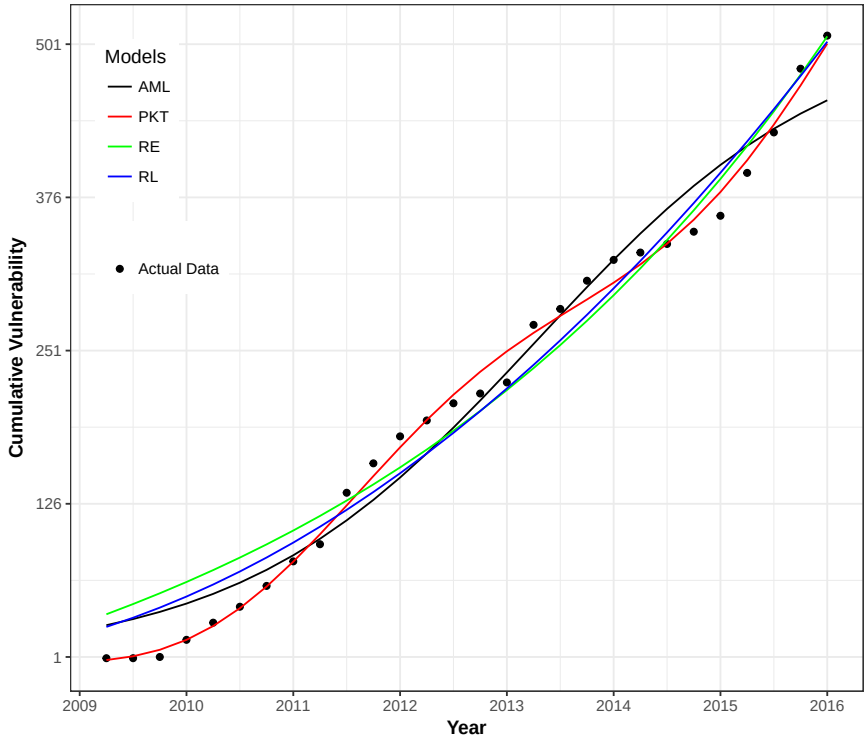

Figure 32.: Model comparison of PKT with RL, RE, and AML for windows 7

Table 9: Model comparison based on Akaike Information Criteria (AIC)

Operating Systems	Models	RSS	AIC
MAC	RL	241314.7	633.5574
	RE	262502.3	638.2703
	AML	334296.1	653.8092
	PKT	**45584.43**	**529.1156**
Linux Kernel	RL	48998.95	578.5852
	RE	124456.7	634.5147
	AML	78961.47	609.2149
	PKT	**18451.8**	**525.987**
Windows 7	RL	16595.93	264.2324
	RE	22418.38	272.6527
	AML	17965.02	268.4519
	PKT	**2808.963**	**220.4948**

The proposed model together with RL, RE, and AML are presented in Figure for Mac OS X. The curves in different color represent estimated fit of cumulative vulnerability data given by different models. We used maximum likelihood estimate method to estimate the best fitting parameters for all models. The goodness of fit tests for different models are presented in the Table 9.

From Figure 30, we can see that the RL, RE and AML predictions overestimate the cumulative vulnerability from 2003 to 2007, underestimate from 2008 to 2012 and overestimate again up to 2016. The PKT follows the actual data quite well. The PKT model comparing with RL, RE, and AML are presented in Figures 31 and 32, and for Linux Kernel and Windows 7. The curves in different color represent estimated fit of cumulative vulnerability data given by different models. For Linux and Windows, the fitted PKT model captures the cyclic trend reasonably better than the other models Figure . The variability of data is higher towards the right tail for all operating systems and PKT model stands out to capture the trend.

The model assumptions are visually assessed through standardized residual plots and normal qq plots. In addition, normal assumption of error is evaluated through Shapiro-Wilk goodness of fit test. The Table 11 demonstrates that the PKT model strongly agrees with the normal assumptions of error. We have utilized Shapiro Wilk Test to assess whether or not residuals are normally distributed. We have made the following hypothesis:

Table 10: Predicted vulnerabilities using PKT model

Operating Systems		2016			
		Q1	**Q2**	**Q3**	**Q4**
Linux Kernel	Predicted Interval	[1340-1370]	[1377-1410]	[1416-1452]	[1457-1496]
	Predicted Vulnerability	1354	1393	1433	1475
	Actual Vulnerability	1306	1407	1443	1522
Windows 7	Predicted Interval	[503-556]	[536-602]	[569-650]	[637-749]
	Predicted Vulnerability	537	573	609	644
	Actual Vulnerability	538	569	596	642
Mac OS X	Predicted Interval	[1252-1497]	[1319-1614]	[1392-1742]	[1472-1888]
	Predicted Vulnerability	1431	1534	1649	1775
	Actual Vulnerability	1499	1573	1656	1756

$\boldsymbol{H_0}$: Standardized Residuals are Normally Distributed.

$\boldsymbol{H_1}$: Standardized Residuals are not Normally Distributed.

We have assumed an alpha level of 5% i.e. if the P value of the Shapiro Wilk Test is below 0.05 null hypothesis is rejected. To verify normality of residuals we should not reject the null hypothesis. Similarly , Shaprio Wilk test statistics value closer to 1 indicates the better agreement to the normality assumptions. Table 11 demonstrate the Shaprio Wilk test statistics, P value for each OS. In every cases, null hypothesis is not rejected. This shows the strong statistical evidence that residuals produced by PKT model are normally distributed.

4.4 Prediction

The proposed analytical model for software vulnerability and other existing models can be used to project the future vulnerability trends. We have applied the proposed model given in equation (4.3) and the other models to estimate the cumulative vulnerability counts for the four quarters of 2016 and the first two quarters of 2017 for all three operating systems: Mac, Linux, and Windows.

The models are fitted using data up to the last quarter of 2015 and vulnerability counts of 2016 and 2017 are estimated by using the fitted model. We used actual vulnerability data of 2016 and 2017 for validation purpose. Nonlinear regression method does not have explicit prediction algorithm to predict the future data. We have implemented the bootstrap method to obtain prediction interval based on fitted value of nls() on some random sample re-sampled from the original one. From the bootstrapped fitted values and predictions, 90% predicted intervals are estimated. Table 10 presents the 90% predicted interval using our proposed PKT model together with the actual vulnerability data for all three operating systems. Notice that the prediction interval

Table 11: Model diagnostics based on Shapiro Wilk goodness of fit test

Operating Systems	Models	Shaprio-Wilk Test	P-Value
MAC	RL	0.92423	0.002169
	RE	0.92273	0.001898
	AML	0.89681	0.000221
	PKT	**0.96599**	**0.1279**
Linux Kernel	RL	0.96072	0.005103
	RE	0.9154	0.000503
	AML	0.942555	0.007053
	PKT	**0.9776**	**0.3358**
Windows 7	RL	0.90054	0.01178
	RE	0.92762	0.05372
	AML	0.96042	0.3567
	PKT	**0.97119**	**0.6128**

in case of linux changes in linear fashion whereas the other two operating system do not exhibit a specific pattern. One of the reason for this observation could be the choice of $\boldsymbol{\omega}$. The value of $\boldsymbol{\omega}$ is determined based on the nature of the data itself and the range of the prediction interval is influenced by $\boldsymbol{\omega}$.

Table 10 shows that the prediction is very accurate for all quarters in case of Linux and Windows. The prediction for the first quarter of 2016 in case of Mac is somewhat underestimated but the prediction for the next three quarters are within the 90% confidence limit. The total number of vulnerability in year 2015 is four times higher than the previous year. One of the prominent reason of higher vulnerability in 2015 is that there were defects in Apple's official tool that provides the framework to develop IOS and OS Apps [64]. The defect on the framework itself caused to increase the total number of vulnerabilities. The prediction results obtained by using the PKT analytical model are found to be more accurate than those given by any other existing models.

The entries in the brackets are lower and upper bounds of 90% prediction interval for the four quarters of 2016. We have not listed the information related to predicted interval and predicted vulnerability of the other models used in this study because we proved that our model stands out best among the existing ones in the previous sections. We have utilized the information presented in Table 10 to compute Sum of Square of Error(SSE) of the predicted vulnerability using PKT model and compared with RL, RE, and AML models presented in the following Table 12.

Table 12: SSE of predicted vulnerabilities

Operating Systems	SSE			
	PKT	**RL**	**RE**	**AML**
Linux Kernel	**1603**	4259.33	13839.33	13710
Windows 7	**63.33**	179.33	109.67	17494.67
Mac OS X	**2185**	151149	128300.3	260835.3

On SSE scale of Linux Kernel OS, PKT model gives the best SSE of 1603, followed by RL at 4259.33, AML at 13710, and RE at 13839.33. Similarly, in case of Windows 7 OS, SSE of PKT model is 63.33, followed by RE at 109.67, RL at 179.33, and AML at 17494.67. Eventually, for Mac OS X OS, it yields SSE of 2185, followed by RE at 128300.3, RL at 151149, and AML at 260835.3. In either of the OS, PKT model has lower SSE in terms of predictive capabilities. Hence, we conclude that PKT model performs better not only for fitting but also for prediction purpose among other models presented in this research.

4.5 Contributions

We have developed an effective differential equation model for software vulnerabilities by utilizing the vulnerability datasets of three major OS: Windows 7, Linux Kernel, and Mac OS X. The proposed analytical model is significantly much better among the existing models in terms of excellent fitting and prediction accuracy. We have presented the chapter contributions in terms of the following points:

1. The developed model can be used by the developers, the users community, and individual organizations to predict their vulnerability levels of their softwares.

2. Developers of the OS can examine the software readiness by predicting the future vulnerability trend.

3. IT manager can allocate the security maintenance resources to detect the forthcoming vulnerabilities.

4. Software company can plan for proper software security patch.

5. The users of OS can benefit by comparing different software of particular domains in terms of risk with their vulnerability. They also can access the risk before patches are applied.

6. This invention is not only applicable to computer OS but it can be used in engineering products, health oriented systems, business and finance forecasting, among others.

Chapter 5

Health Science : A Predictive Analytical Model for Stomach Cancer Data

We have developed a predictive software vulnerability model in Chapter 4. This predictive model is not only applicable to computer OS but it can be used in various domains such as engineering, finance, business, health science, among others. For instance, we have implemented the idea on health science, to predict the malignant tumor size of stomach cancer as a function of age based on the given historical data.

This chapter is organized as follows: Section 5.1 describes the background and introduction to stomach cancer in detail. 5.2 explains the structure of the database. Section 5.3 explores the idea behind the model development. In Section 5.4 results of the model are presented along with model diagnostic and prediction. Finally, in Section 5.6 major contributions of this chapter are highlighted.

5.1 Introduction

In general cancer appears when cells in the body start to grow out of control. If the cancer starts in the stomach then it is called stomach or gastric cancer. There are several risk factors associated with this type of cancer namely gender, ethnicity, age, geography, tobacco use, and diet. There is no concrete way to prevent stomach cancer. When a tumor grows, the patients may have more serious symptoms. Surgery, chemotherapy, targeted therapy, and radiation therapy are the best possible available treatment options for this cancer. The location of the tumor, size, age, and stage are the primary risk factor considered while treatment.

According to the American Cancer Society [67], it is the fifth leading cancer and the third leading cause of death of human beings. It also estimates for 2018 that there are about 26,240 cases of stomach cancer that will be diagnosed (16,520) in men and 9,720 in women) and about 10,800 people will die from this type of cancer (6,510 men and 4,290 women). The total survival rate of the patients depends on several factors such as which stage patient falls in, age, and previous treatments applied. Stomach cancer is very common in underdeveloped and developing nations due

to the diet habits and food they eat. For instance, the Japan Cancer Society reports (2007) one in every three deaths was attributed to stomach cancer.

The recent study [68] exclusively focused on statistical analysis and modeling of stomach cancer data. More specifically, extensive parametric analysis was performed on race and sex of patients with malignant tumors. The overall conclusion of the study was malignant stomach tumor sizes significantly different on gender and races. Similarly, quantile regression and decision tree analysis techniques were implemented to find the probabilistic behavior of the given phenomenon. Similarly, quantile regression model explored that patient age was the most significant variable to determine the size of the malignant tumor. When age of the patient increases, so does the tumor size. Thus, the relationship between tumor size and age of the patient can be expressed as:

$$\textbf{Tumor Size (mm)} = \textbf{f(Age)}$$

This function is exactly similar to the predictive model we developed in Chapter 4. The model was based on cumulative number of vulnerabilities of the given OS with respect to the calendar time. We implement the same philosophy in case of stomach cancer data.

5.2 Data Description

Our analysis and modeling is based on the data obtained from Surveillance Epidemiology and End Results (SEER) program of the United States. The SEER database include incidient and population data with respect to age, sex, race, year of diagnosis, geographic areas (SEER registry and county), and size of tumors. Figure 33 presents the schematic diagram that reveals the holistic picture of the data structure.

Figure 33 depicts data from a a total of 11,462 stomach cancer patients collected from 2004 to 2013. The total numbers of patients are further classified into male and female with malignant and benign tumors. There are so many patients with the same age having different tumor size. We have computed the mean of tumor size of each age category, then the cumulative tumor size of each category is determined. Cumulative tumor size measured in millimeters with respect to age is the fundamental quantitative value for our analysis. We have developed the statistical predictive model of malignant tumor size for white female category only.

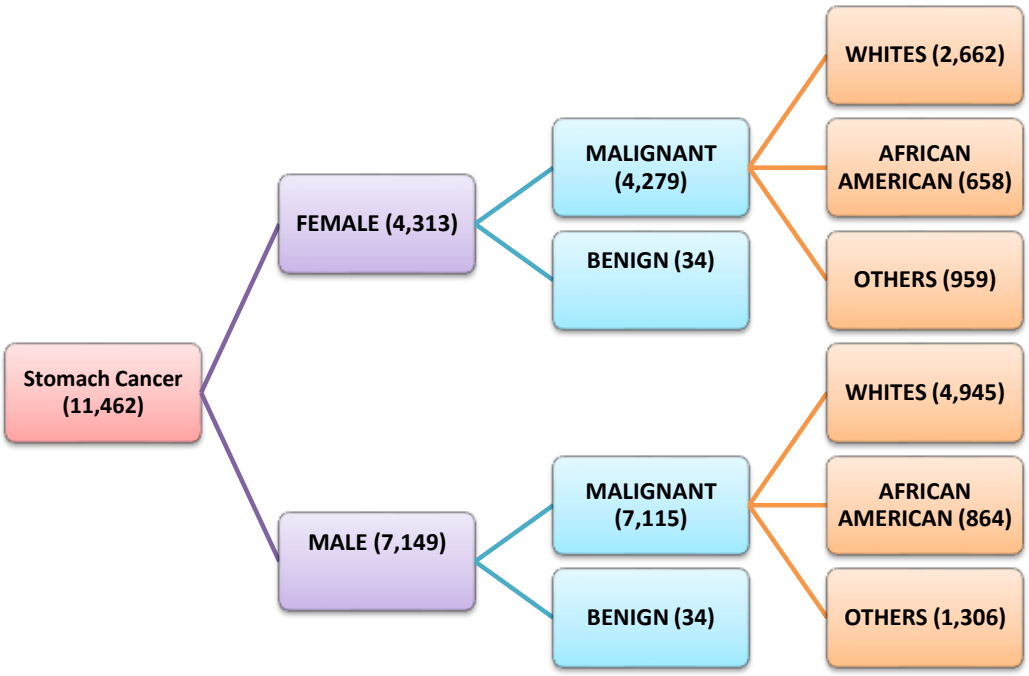

Figure 33.: Schematic diagram of stomach cancer patients with malignant and benign tumor size

5.3 Modeling Approach

In this section, the basic aspect of modeling part is highlighted. The derivation of the model was completely explained in Chapter 4 under section 4.2. The final form of the analytical model is,

$$\boldsymbol{\Omega(t) = c_1 \cos(\omega t) + c_2 \sin(\omega t) + c_3 t^2 + c_4 t + c_5} \tag{5.1}$$

In the given equation (5.1), t refers to the time; in this case age of the patients; $\boldsymbol{\Omega(t)}$ is the cumulative tumor size in terms of millimeters. $\boldsymbol{\omega = \frac{2\pi}{T}}$ depends upon the time period $\boldsymbol{T}$ and $\boldsymbol{T}$ depends on the data set. In our case for the White Female patient, there are 41 age specific data points. For one cycle, $\boldsymbol{T = 41}$ and $\boldsymbol{\omega = \frac{2\pi}{41} \approx 0.1532484}$; for one and half cycle, $\boldsymbol{T = 27}$ and $\boldsymbol{\omega = \frac{2\pi}{36} \approx 0.23271056}$; for two cycles, $\boldsymbol{T = 21}$ and $\boldsymbol{\omega = \frac{2\pi}{21} \approx 0.2991993}$ and so on. We have considered just one cycle since we do not have enough cumulative data points in our model and use $\boldsymbol{\omega \approx 0.1532484}$. Single cycle describes the overall behavior of the given data.

By implementing the computational procedure mentioned in Chapter 4 under Section 4.2, the estimated values of the parameters are: $\boldsymbol{c_1}$=55.62, $\boldsymbol{c_2}$=-12.99, $\boldsymbol{c_3}$=0.24, $\boldsymbol{c_4}$=11.96, and $\boldsymbol{c_5}$=-792.97. Therefore, the model that provides the cumulative tumor size of the patients at any age t for White

Female is given by,

$$\begin{aligned}\boldsymbol{\Omega(t)} \quad &= \quad \boldsymbol{55.62\cos(0.15t) - 12.99\sin(0.15t)} \\ &\quad \boldsymbol{+0.24t^2 + 11.96t - 792.97.}\end{aligned} \tag{5.2}$$

Figure 34 exhibits predicted 95% confidence band values of cumulative tumor size produced by our analytical model (5.2) along with the actual data points. The solid red curve represents the actual and predicted values by the proposed model. The black and green lines represent lower and upper 95% confidence of the true values. Figure shows a good fit within the given confidence limits. Since we have a limited number of cumulative data points, it does not perfectly reveal the cyclic increasing behavior graphically but when the number of data points increases, then it is feasible to see its pattern via graph.

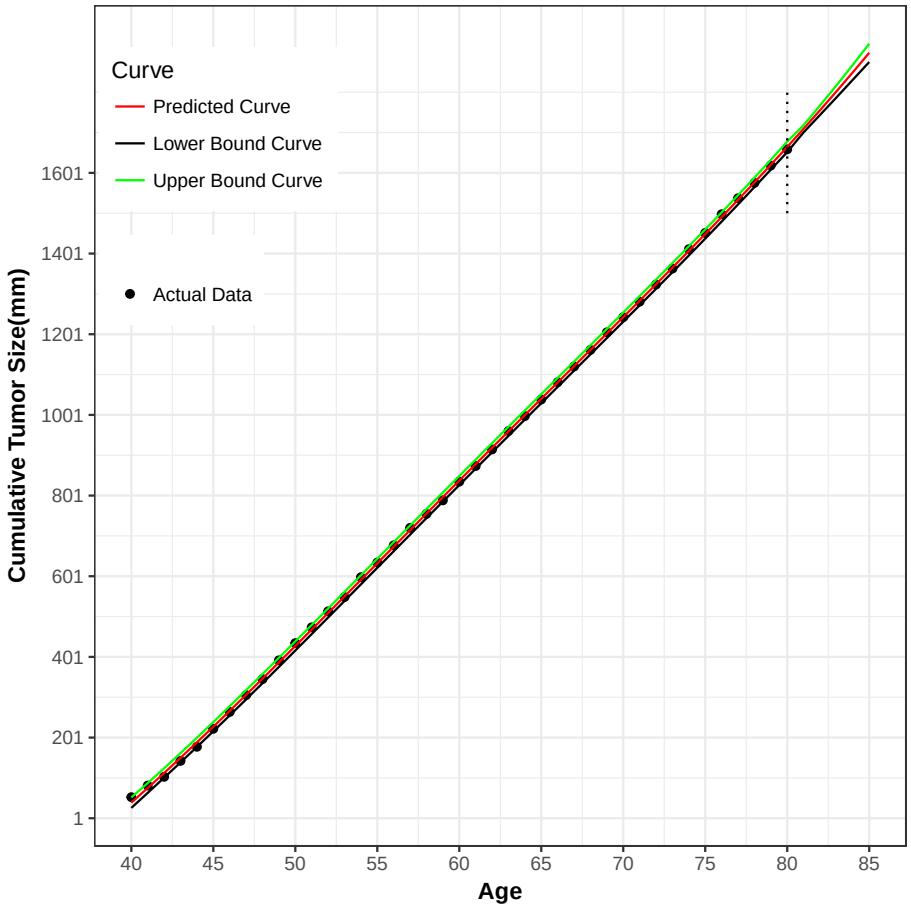

Figure 34.: Prediction with a 95% confidence band of white female patients

5.4 Model Diagnostic

We have implemented a graphical procedure to validate our model to detect the model violations if they exist. We assess goodness of fit through the standardized residuals. It is computed by dividing the centered residuals by the residual standard error.

Figure 35, the very first graph (left) is plotted to standardized residuals vs. fitted value to evaluate if there is any indication of variance heterogeneity. Similarly, the second graph (right) is normal probability plot (or QQ plot) that compares the standardized residuals vs. theoretical values from a standard normal distribution.

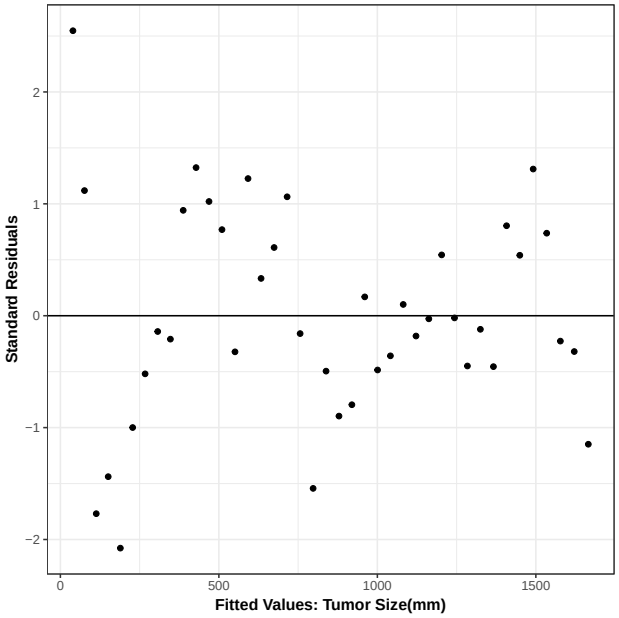

Figure 35a.: Standardized residual plot (left)

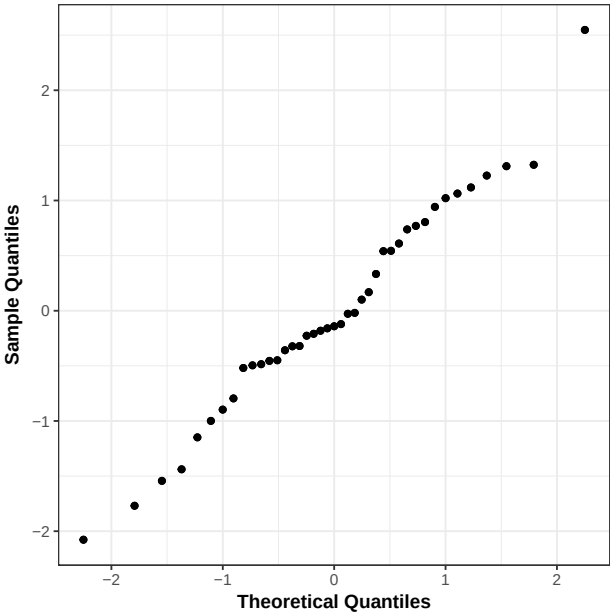

Figure 35b.: Standardized QQ plot (right)

Figure 35, clearly shows no problems with the model assumptions, residual seem to be approximately normally distributed. In addition to supplement more evidence statistically, we have utilized Shapiro Wilk Test to assess whether or not residuals are normally distributed. We have made the following hypothesis:

$\boldsymbol{H_0}$: Standardized Residuals are Normally Distributed.

$\boldsymbol{H_1}$: Standardized Residuals are not Normally Distributed.

We have assumed an alpha level of 5% i.e. if the P value of the Shapiro Wilk Test is below 0.05 the null hypothesis is rejected. To verify normality of residuals we should not reject the null hypothesis. Similarly, Shapiro Wilk test statistics value closer to 1 indicates the better agreement

to the normality assumptions. In this case, the value of Shapiro Wilk Test is 0.98164 along with p value of 0.7374. In a nutshell, the null hypothesis is not rejected. This shows the strong statistical evidence that residuals produced by our analytical model are normally distributed.

5.5 Prediction

The predictive analytical model for cumulative tumor size (mm) is used to project the future tumor size of the patients. We have applied the proposed model given in equation (5.2) to estimate the cumulative tumor size from age 81 to 85. The models are fitted using data from age 40 to age 80 and tumor size ranges from age 81 to 85 are estimated by using the fitted model.

Table 13: Cumulative: actual vs predicted tumor size (mm)

	AGE					
	80	81	82	83	84	85
Actual	1659	1701	1746	1798	1850	1896
Predicted	1665	1710	1756	1802	1850	1898

Table 13, shows the predicted cumulative tumor size (mm). To make it more user friendly and interpretable value needs to be converted to exact tumor size with respect to age. Once we subtract the current cumulative tumor size to its immediate past cumulative tumor size, the exact tumor size (mm) in a particular age is obtained. The exact predicted tumor size along with its age is illustrated in Table 14.

Table 14: Exact: actual vs predicted tumor size (mm)

	Age				
	81	82	83	84	85
Actual	42	45	52	52	46
Predicted	45	46	46	48	48

Utilizing the developed model, we forecasted the tumor size ranges from the age of 81 to 85. This comparison Table 14 reveals the accuracy and reliability of our developed model.

5.6 Contributions

We have generalized the differential equation model initially developed for software vulnerabilities to stomach cancer data. The proposed analytical model shows significantly better results in terms of excellent fitting and prediction accuracy. We have presented the chapter contributions in terms of the following points:

1. The developed model can be used by the physician to make more accurate decision about the patients.

2. Based on the projected tumor size of the patient different preventative actions can be implemented to decrease the situation more severe.

3. It helps in planning any sort of resources ahead of time both for the patient and doctors.

4. This invention is not only applicable to computer health oriented systems, but it can be used in engineering products, business and finance forecasting, among others.

Chapter 6

Future Research

Our future works mainly consist of three directions. The very first is to integrate software vulnerability and software reliability using Bayesian approach. The second is to improve internal structure of Common Vulnerability Scoring System (CVSS) framework. The third is to apply power law process and non-homogeneous Poisson process.

6.1 Integration of Software Vulnerability and Software Reliability

Security and reliability are the major ingredients of a complex software system. When the software contains a high level of vulnerabilities, then it can be inferred that given software is less reliable and vice versa. Thus, software reliability can be estimated when the level of the vulnerabilities in the software is provided and vice versa.

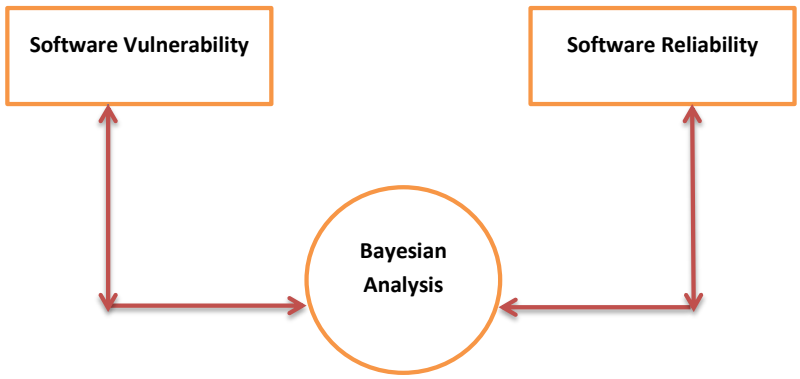

Figure 36.: Software reliability vs. software vulnerability using Bayesian approach

In the near future, we want to integrate the two independent domains using Bayesian analysis to quantitatively define the relationship. This idea is demonstrated via the schematic diagram in Figure 36.

6.2 Redesigning the CVSS Framework

Another important research objective that we have for the near future is to completely redesign the CVSS framework as a whole. As we explained in the previous chapters, CVSS is the open framework that provides the quantitative scores representing the overall severity and risk of the known vulnerabilities. It is maintained by the **Forum of Incident Response Team** (FIRST) [14]. A CVSS score is on the scale of 0 to 10 and consists of three major metrics group: base, temporal and environmental as mentioned in Figure 1. Vulnerabilities with the base score ranging from 0-3.9 is considered **Low** vulnerability, 4.0-6.9 as **Medium**, and 7.0-10 as **High**. Calculation of the base score depend upon the several metrics and standard equations. CVSS does not explain the internal methodology to measure the degree of confidence with respect to the calculation of these scores. We expect to coordinate the NVD team to improve the overall scoring system further.

6.3 Applying Power Law Process and Non-homogeneous Poisson Process

Our primary objective is to develop a set of statistical model and methodologies in cybersecurity. With respect to the most commonly used and the best available data source, two major methodologies can be applied to answer several questions in software vulnerability area. Once the intensity function is identified based on the given data sets in the subject area, we can have several answers for instance probability of system being exploited is increasing, decreasing, or remains same.

References

[1] National Vulnerability Database (NVD),
https://nvd.nist.gov/

[2] Flexera Vulnerability Review (2017),
https://resources.flexera.com/web/pdf/Research-SVM-Vulnerability-Review-2017.pdf

[3] CVE Details,
http://www.cvedetails.com/

[4] Cyber Crime Report (2017),
https://cybersecurityventures.com/2015-wp/wp-content/uploads/2017/10/2017-Cybercrime-Report.pdf

[5] Abraham, S., & Nair, S. (2014) Cyber security analytics: a stochastic model for security quantification using absorbing markov chains, Journal of Communications, 9(12), 899-907.

[6] Kijsanayothin, P. (2010) Network security modeling with intelligent and complexity analysis, Ph.D Dissertation, Texas Tech University.

[7] Phillips, C., & Swiler, L. P. (1998) A graph-based system for network-vulnerability analysis, In Proceedings of the 1998 workshop on New security paradigms, 71-79.

[8] Balzarotti, D., Monga, M., & Sicari, S. (2006) Assessing the risk of using vulnerable components, In Quality of Protection, Springer, 65-67.

[9] Mehta, V., Bartzis, C., Zhu, H., Clarke, E., & Wing, J. (2006) Ranking attack graphs, International Workshop on Recent Advances in Intrusion Detection, Springer, 127-144.

[10] Joh, H., & Malaiya, Y. K. (2011) Defining and assessing quantitative security risk measures using vulnerability lifecycle and cvss metrics, International Conference on Security and Management (SAM), 10-16.

[11] Singhal, A., & Ou, X. (2017) Security risk analysis of enterprise networks using probabilistic attack graphs, Network Security Metrics, 53-73.

[12] Xie, L., Zhang, X., & Zhang, J. (2013) Network security risk assessment based on attack graphs, JCP, 8(9), 2339-2347.

[13] Mell, P., Scarfone, K. and Romanosky, S.(2007) A complete guide to the common vulnerability scoring system version 2.0, FIRST-Forum of Incident Response and Security Teams, 1-23, https://www.first.org/cvss/cvss-v2-guide.pdf

[14] Forum of Incident Response and Security Teams (FIRST), https://www.first.org/about

[15] Rahimi, S., & Zargham, M. (2013), Vulnerability scrying method for software vulnerability discovery prediction without a vulnerability database, IEEE Transactions on Reliability, 62(2), 395-407.

[16] Scandariato, R., Walden, J., Hovsepyan, A., & Joosen, W. (2014), Predicting vulnerable software components via text mining, IEEE Transactions on Software Engineering, 40(10), 993-1006.

[17] Shin, Y., & Williams, L. (2013), Can traditional fault prediction models be used for vulnerability prediction?., Empirical Software Engineering, 18(1), 25-59.

[18] Nguyen, V. H., & Tran, L. M. S. (2010), Predicting vulnerable software components with dependency graphs,Proceedings of the 6th International Workshop on Security Measurements and Metrics, ACM.

[19] Alhazmi, O. H., & Malaiya, Y. K. (2006, January), Prediction capabilities of vulnerability discovery models, Reliability and Maintainability Symposium, IEEE, 86-91.

[20] Musa, J. D., & Okumoto, K. (1984), A logarithmic poisson execution time model for software reliability measurement, Proceedings of the 7th international conference on Software engineering, IEEE, 230-238.

[21] Rescorla, E. (2005), Is finding security holes a good idea?., IEEE Security & Privacy, 3(1), 14-19.

[22] Anderson, R. (2002), Security in open versus closed systems?the dance of Boltzmann, Coase and Moore., Technical report, Cambridge University, England.

[23] Pokhrel, N. R., & Tsokos, C. P. (2017), Cybersecurity: A Stochastic Predictive Model to Determine Overall Network Security Risk Using Markovian Process, Journal of Information Security, 8(02), 91-105.

[24] Jha, S., Sheyner, O., & Wing, J. M. (2002), Minimization and reliability analyses of attack graphs, Technical Report, Carnegie-Mellon University Pittsburgh PA School of Computer Science.

[25] Kemmerer, R. A., & Vigna, G. (2002), Intrusion detection: a brief history and overview, Computer, 35(4), supl27–supl30.

[26] Bilge, L., & Dumitras T., An empirical study of zero-day attacks in the real world, CCS'12, 16-18.

[27] Xie, A., Cai, Z., Tang, C., Hu, J., & Chen, Z. (2009), Evaluating network security with two-layer attack graphs, Computer Security Applications Conference IEEE, 127–136.

[28] Bolch, G., Greiner, S., De Meer, H., & Trivedi, K. S. (2006), Queueing networks and Markov chains: modeling and performance evaluation with computer science applications, John Wiley & Sons.

[29] Trivedi, K. S. (2011), Probability & Statistics with Reliability, PHI Learning Pvt. Limited.

[30] Sahner, R. A., Trivedi, K.,& Puliafito, A. (2012). Performance and reliability analysis of computer systems: an example-based approach using the SHARPE software package. Springer Science & Business Media.

[31] Abraham, S., & Nair, S. (2014), Cyber security analytics: a stochastic model for security quantification using absorbing markov chains, Journal of Communications, 9(12), 899-907.

[32] Cinlar, E. (2013), Introduction to stochastic processes, Courier Corporation.

[33] Sheyner, O., & Wing, J. (2003), Tools for generating and analyzing attack graphs, International Symposium on Formal Methods for Components and Objects, 344–371.

[34] Mehta, V., Bartzis, C., Zhu, H., Clarke, E., & Wing, J. (2006), Ranking attack graphs, Advances in Intrusion Detection Springer, 127-144.

[35] Hewett, R., & Kijsanayothin, P. (2008), Host-centric model checking for network vulnerability analysis, Computer Security Applications Conference, IEEE, 225–234.

[36] Ammann, P., Pamula, J., Ritchey, R., & Street, J. (2005), A host-based approach to network attack chaining analysis, Computer Security Applications Conference, IEEE.

[37] Pokhrel, N. R., Rodrigo, H., & Tsokos, C. P. (2017), Cybersecurity: Time Series Predictive Modeling of Vulnerabilities of Desktop Operating System Using Linear and Non-Linear Approach, Journal of Information Security, 8(04), 362.

[38] National Institute of Standard and Technology (NIST) Report (2014), http://www.nist.gov/

[39] Secunia Vulnerability Review (2015), https://secunia.com/?action=fetch & filename=secunia _vulnerability _review _2015 _pdf.pdf

[40] Vulnerability Review (2016), http://www.flexerasoftware.com/enterprise/resources/research/vulnerability-review/

[41] Microsoft Vulnerabilities Study (2015), Mitigating Risk by Removing User Privileges, http://learn.avecto.com/2015-microsoft-vulnerabilities-report

[42] Othmane, L. B., Chehrazi, G., Bodden, E., Tsalovski, P., & Brucker, A. D. (2017), Time for addressing software security issues: prediction podels and impacting factors, Data Science and Engineering, 2(2), 107-124.

[43] Desktop Operating System Market Share, https://www.netmarketshare.com/

[44] https://assets.documentcloud.org/documents/2459197/bit9-carbon-black-threat-research-report-2015.pdf

[45] Phillips, P.C.& Perron, P.(1998) Testing for a unit root in time series regression, Biometrika,75(2) 335–346.

[46] Frank, R. J., Davey, N., & Hunt, S. P. (2001), Time series prediction and neural networks, Journal of Intelligent and Robotic Systems, 31(1-3), 91–103.

[47] Edwards, T., Tansley, D., Frank, R., & Davey, N. (1997), Traffic trends analysis using neural networks, Int Workshop on Applications of Neural Networks to Telecommunications.

[48] Patterson, D. W., Chan, K. H., & Tan, C. M. (1993), Time series forecasting with neural nets: a comparative study, International Conference on Neural Network Applictions to Signal Processing, 269-274.

[49] Bengio, S., Fessant, F., & Collobert, D. (1995), A connectionist system for medium-term horizon time series prediction, Intl. Workshop Application Neural Networks to Telecoms, 308-315.

[50] Cortes, C., & Vapnik, V. (1995), Support-vector networks, Machine Learning, 20(3), 273-297.

[51] Musa, John D and Okumoto, Kazuhira(1984), A logarithmic poisson execution time model for software reliability measurement, 7th international conference on Software engineering, 230–238.

[52] Anderson, Ross(2002), Security in open versus closed systems? the dance of Boltzmann, Coase and Moore, Technical report, Cambridge University, England.

[53] O. H. Alhazmi and Y.K. Malaiya(2005), Quantitative vulnerability assessment of systems software, Annual Reliability and Maintainability Symposium, 615-620.

[54] O. H. Alhazmi and Y. K. Malaiya(2008), Application of vulnerability discovery models to major operating systems, IEEE Transactions on Reliability, 57(1), 14-22.

[55] Alhazmi, Omar H., and Yashwant K. Malaiya(2005), Modeling the vulnerability discovery process, Software Reliability Engineering, IEEE, 10-pp.

[56] Alhazmi, Omar H., and Yashwant K. Malaiya(2006), Prediction capabilities of vulnerability discovery models, Reliability and Maintainability Symposium, IEEE 86-91.

[57] Baty, Florent, Christian Ritz, Sandrine Charles, Martin Brutsche, Jean-Pierre Flandrois, and Marie-Laure Delignette-Muller(2015), A toolbox for nonlinear regression in R: the package nlstools, Journal of Statistical Software 66(5), 1-21.

[58] Bishop, Peter, and Robin Bloomfield. A conservative theory for long-term reliability-growth prediction [of software](1996), IEEE Transactions on Reliability 45(4), 550-560.

[59] G. Grothendieck, nls2: Non-linear regression with brute force, R package version 0.2., https://CRAN.R-project.org/package=nls2

[60] Joh, HyunChul, and Yashwant K. Malaiya(2011), Defining and assessing quantitative security risk measures using vulnerability lifecycle and cvss metrics, International conference on security and management (sam), 10-16.

[61] V. H. Nguyen and F. Massacci(2012), An idea of an independent validation of vulnerability discovery models, 89-96.

[62] E. Rescorla (2005), Is finding security holes a good idea, IEEE Computer Securuty & Privacy, 14-19.

[63] R Core Team(2017), R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/

[64] https://assets.documentcloud.org/documents/2459197/bit9-carbon-black-threat-research-report-2015.pdf

[65] Net Market Share,
https://www.netmarketshare.com/

[66] Pokhrel, N.R., Rodrigo, H. and Tsokos, C.P. (2017), Cybersecurity: time series predictive modeling of vulnerabilities of desktop operating system using linear and non-linear approach, Journal of Information Security, 8, 362-382.

[67] American Cancer Society (ACS),
https://www.cancer.org/cancer/stomach-cancer/about/key-statistics.html.

[68] Chao Gao (2017), Statistical analysis and modeling of stomach cancer data, Ph.D Dissertation, University of South Florida.

[69] Abbas, A.K., Bassam, R.(2009), Phonocardiography signal processing, Morgan and Claypool Publishers.

[70] Abdulkader, S.N., Atia, A., Mostafa, M.M. (2015), Brain computer interfacing: Applications and challenges, Egyptian Informatics Journal. 12(2), 213–230.

[71] Addison, P.S. (2005) Wavelet transforms and the ECG: A review, physiological measurement. 26(5), 155-199.

Appendix A

Base Score: Common Vulnerability Scoring System (CVSS)Version 2.0

Base Equation:

Scoring equations and algorithms for the base metric groups are described below. Further discussion of the origin and testing of this equation is available at www.first.org/cvss. Base score is the foundation of the scoring system, it is listed as:

Base Score = round_to_1_decimal(((0.6*Impact)+(0.4*Exploitability)–1.5)*f(Impact))

Impact = 10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))

Exploitability = 20* AccessVector*AccessComplexity*Authentication

f(impact)= 0 if Impact=0, 1.176 otherwise

```
AccessVector     = case AccessVector of
                       requires local access: 0.395
                       adjacent network accessible: 0.646
                       network accessible: 1.0

AccessComplexity = case AccessComplexity of
                       high: 0.35
                       medium: 0.61
                       low: 0.71

Authentication   = case Authentication of
                       requires multiple instances of authentication: 0.45
                       requires single instance of authentication: 0.56
                       requires no authentication: 0.704

ConfImpact       = case ConfidentialityImpact of
                       none:             0.0
                       partial:          0.275
                       complete:         0.660

IntegImpact      = case IntegrityImpact of
                       none:             0.0
                       partial:          0.275
                       complete:         0.660

AvailImpact      = case AvailabilityImpact of
                       none:             0.0
                       partial:          0.275
                       complete:         0.660
```

Appendix B

Base Metric Evaluation Score

Example:

In this section, we have selected the specific vulnerability (CVE-2003-0818) to demonstrate how its base score is computed. More explanation is available at www.first.org/cvss.

BASE METRIC	EVALUATION	SCORE
Access Vector	[Network]	(1.00)
Access Complexity	[Low]	(0.71)
Authentication	[None]	(0.704)
Confidentiality Impact	[Complete]	(0.66)
Integrity Impact	[Complete]	(0.66)
Availability Impact	[Complete]	(0.66)

FORMULA	BASE SCORE
Impact = 10.41*(1-(0.34*0.34*0.34)) == 10.0	
Exploitability = 20*0.71*0.704*1 == 10.0	
f(Impact) = 1.176	
BaseScore =((0.6*10.0)+(0.4*10.0)-1.5)*1.176	== (10.0)

Thus, final value 10.0 is the base score of CVE-2003-0818 vulnerability.