OPTIMAL SECURITY PATCH MANAGEMENT FOR LARGE INFRASTRUCTURES BASED ON PRIORITIZED ASSET INVENTORY

By

Muhammad Muqeet Kamal

119193-MS(IS)-8-2015

Supervisor

Dr. SHAHZAD SALEEM

DEPARTMENT OF COMPUTING

A thesis submitted in partial fulfillment of the requirements for the degree of Masters in Information Security (MS IS)

In

School of Electrical Engineering and Computer Science

National University of Sciences and Technology (NUST)

Islamabad, Pakistan

(July 2019)

Dedication

I would like to dedicate this to my parents, friends, family, instructors, colleagues, work fellows and all those who have enabled me to pursue my interest and work in the field of my choice.

Certificate of Originality

I hereby declare that this submission my own work and to the best of my knowledge. It contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgment, is made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistic is acknowledged.

Author Name: **Muhammad Muqeet Kamal**

Signature: _____

Acknowledgement

I would start off by thanking the ALLAH Almighty for his countless blessing and enhancing me to pursue and complete my research. None of this would have been possible without HIS will. Then I am thankful to HIS most precious blessing on me in form of my highly supportive and infinitely lovable parents and their tireless efforts in providing me an excellent education career. I am also thankful to my teachers from school till now who have a major role in not only my educational achievements but my personality and grooming.

I am highly thankful to my supervisor Dr. Shahzad Saleem for his continuous guidance support, genuine goodwill for my career and encouragement that I required to complete this task. He has been more than a mentor to me not only during my research phase but during my whole tenure at SEECS, NUST as a postgraduate student. Without even shred of a doubt, I believe none of this would have been possible without his expert insight and requisite guidance.

The role of my committee members Dr. Hasan Tahir, Dr. Naveed Ahmed, and Ms Haleemah Zia in completing this dissertation is far beyond a few words of gratification. Their confidence in my abilities and timely support has contributed a lot towards the completion of my thesis.

Lastly, I will also express my gratitude to my workplace team leads Mr. Emal Khan and Mr. Mubsshar Ismail who encouraged and support me to pursue Masters along with my regular job and allowed me days off for my studies and facilitated in any way possible.

Despite all the assistance provided by supervisor, committee members and others, I take the responsibility for any errors and omissions which may unwittingly remain.

Muhammad Muqeet Kamal

Abstract

The ever-increasing trend of discovery of new vulnerabilities has created a challenge enterprise and organization to secure themselves from the associated threats. The past few years alone have witnessed many such critical vulnerabilities which were reported to cost considerable losses to the organizations. Although vendors and developers publish patches before an exploit is made publicly available but due to the unmanageable problem of deploying patches in the production environment, organizations fail to secure their systems and the exploits make way into their infrastructure. This is attributed towards the multi-dimensional challenges faced by the organizations such as production downtime, cost of patching versus the available resources and the potential impact of running into an unforeseen issue and loss of business both in terms of reputation and profit.

This has given rise to finding out methods in cases of emergency patching, where waiting for periodic cycle is not tolerable for organization to prioritize the enterprise's most valuable assets which can be secured against the vulnerabilities while keeping downtime and business loss to the minimum. While the efforts have produced various models and results, they have only been able to produce a generic output which do not provide a solution to every enterprise with various business dynamics. One such example is the Common Vulnerability Scoring System (CVSS). Although CVSS tries to accommodate environmental factors, but it lacks the knowledge of various organizational processes and challenges faced in patching. We have provided an asset prioritization solution based on the CVSS framework and enriching it with organizational constraints by following the weighted sum model.

To make it more organization specific, we have allowed the input of business constraints and resources and employed SMT solvers to produce a solution which is more specific and provides opportunity to secure maximum number of valuable assets under those constraints for a specific vulnerability. The outcome of this research is validated by subject matter experts and have been found to be helpful than the approaches they have been following in the past.

Table of Contents

List of Images

List of Tables

This page is intentionally left blank!

1 INTRODUCTION

This section highlights the background and the basic points to develop understanding of the research work. This is followed by the motivation, intended audience, problem statement and the aim of the research within the scope of the study. Moreover, technical terms and jargons are explained and discussed for knowledge of all related and non-related audience. The section concludes with the overview of design and how the research is structured. Following are the sub-sections:

- Section 1.1 Background
- Section 1.2 Motivation
- Section 1.3 Research Question
- Section 1.4 Problem Statement
- Section 1.5 Goals and Objectives
- Section 1.6 Intended Audience
- Section 1.7 Scope of the Study
- Section 1.8 Organization of Dissertation

1.1 Background

Risk assessment and prioritization forms the basis of much of the domain of information security. This research revolves around the same concept of analysis and prioritization of risks against a specific threat with the knowledge of the organizational environment and controls in place. In this section we go in the details of risk assessment, prioritization of risks, the process of mitigation of risk with a comprehensive analysis of patch management. This sub-section concludes with an overview of SMT solvers.

This shall be followed with the domain of qualitative and quantitative risk analysis and their individual pros and cons. The sub-sections shall also further elaborate the applicability of qualitative and quantitative risk analysis.

1.1.1 Risk Management & Vulnerabilities

In order to understand the crucial position of vulnerabilities and patch management, lets start with concepts of Risk Management and where does this patching lies within that management cycle. As per NIST SP 800-30,

'Risk is a function of the likelihood of a given threat-source's exercising a particular potential vulnerability, and the resulting impact of that adverse event on the organization'[1].

As organization's dependency grows more and more on the information systems and the data flowing through the system, their has been a sharp increase in the activities done by unauthorized actors with intent to gain access or harm the data. The three major aspects of data security are confidentiality, integrity and availability. Confidentiality of data means to protect from unauthorized access, integrity is described as to secure data against unauthorized modification while availability is to ensure that the data remains accessible for intended and authorized individuals at any moment of time.

Risk management refers to a complete process. It starts from identifying risks posed to an organization against threats to confidentiality, integrity and availability which arise due to exploitation of a vulnerability by a certain threat actor. Afterwards, the risks are evaluated and prioritized, and a treatment plan is established for handling those risks.

1.1.2 Risk Model

To understand the risk management process, it is important to establish relationships between various risk factors. The factors become inputs in risk models which ascertain the severity of risks. Listed below are various risk factors.

a) Threat

Threat is classified as any occurrence, event or chain of events that affects an organization adversely and may cause harm or damage to assets, operations or

individuals working within the organization. A threat exists due to certain actions and intentions of threat sources. These may be in the form humans or might be some natural disaster or a piece of computer code itself.

b) Vulnerability

A vulnerability is defined as a weakness in the organization's process, configurations, information system, access procedures or human practices. Vulnerabilities arise due to poorly defined processes, improper configuration of systems, lack of implementation of process or human actions due to improper awareness. Threats exploit vulnerabilities to cause harm to the confidentiality, integrity or availability of the information within an organization which leads to actualization of a risk.

c) Impact

The extent of damage or harm that is anticipated to result in event of the risk being materialized is classified as impact. This can be either in the form of financial loss or damage to reputation and goodwill of the organization. In terms of CIA, impact can be termed as unauthorized disclosure of data, unauthorized modification of data and corruption or disruption of data to render unusable to authorized users.

d) Likelihood

The probability of a risk being materialized because of an event in which a threat exploits a given vulnerability is termed as likelihood. Likelihood is estimated based on various factors such as historical statistics, adversary skills, intent and the goal.

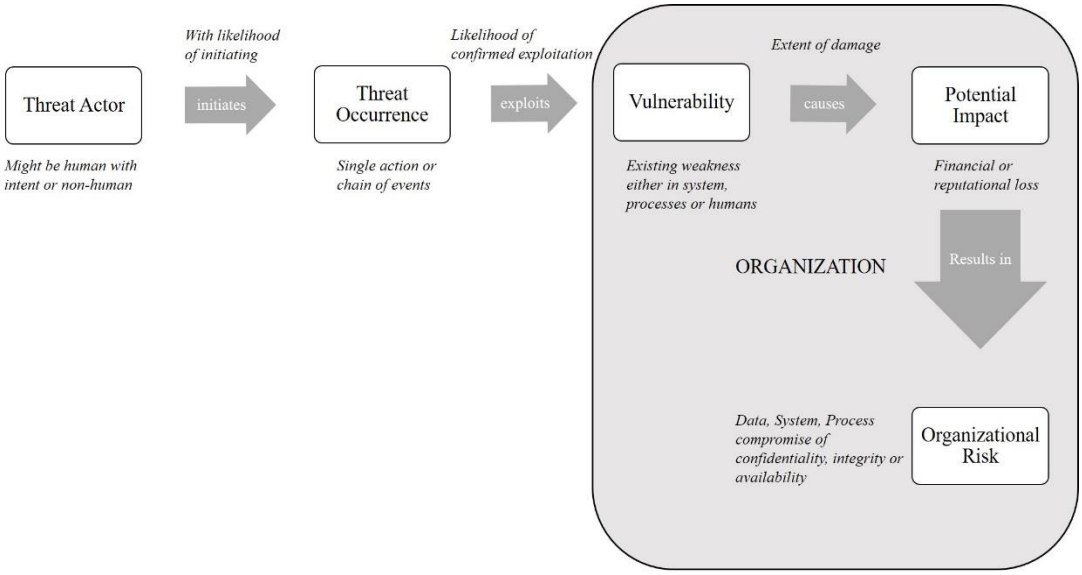


Figure 1-1 Risk Model and its various factors

1.1.3 Risk Assessment

The exercise of identifying existing risks and weighing their severity based on the likelihood of their occurrence and the magnitude of the adverse impact on organization is called Risk Assessment. Organizations tend to use various methods of carrying out the assessment. Smaller organizations rely on automated tools to generate risk reports to reduce costs while larger organizations go deeper and gather input from various process and business owners to get more in-depth picture[2]. Risk assessment is a cost intensive process. The costs of conducting assessments can jump significantly high based upon the infrastructure size, number of business process and the diversity of processes. Organizations also choose to go with pre-existing assessment approaches to reduce cost[3]. The value of risk is calculated by below equation

$$Risk = LikelihoodxImpact$$

Most risk assessment techniques can be classified in two types: qualitative and quantitative. In below sub-section, we take a look at both of these.

a) Qualitative Risk Assessment

Using a pre-defined rating scale, this type of risk assessment focuses on prioritization of identified risks in a broader class depending upon severity. Based on their likelihood of occurrence and the impact on operations, they are rated as Low, Medium or High.

b) Quantitative Risk Assessment

As the title suggests, quantitative risk assessment assigns numerical values or weights to the identified risks and prioritizes them in order of their severity rating to obtain nearest possible probability to the real value of the risk.

1.1.4 Risk Prioritization

Whether qualitative or quantitative, all risk assessment methodologies set their prime objectives as prioritization of high severity risks[4][5]. Risk prioritization includes the analysis of identified risks and their calculated scores and weigh them based on the value of asset to correctly identify which ones to be treated first. However, this is not as simple as it seems. Organizations have been facing challenges in prioritizing their risks and efforts have been made to provide a comprehensible solution to the problem[6].

Quantitative methods for risk prioritization focus on determining the loss organization shall have to face in financial terms in case a risk is materialized. Financial loss estimation not only includes monetary value of the asset but also the value of the data, the consequences resulting in the exposure or modification of data and the risk to reputation. All this is included in a monetary value and this is calculated against the *Exposure Factor* which defines what percentage of value of asset shall be lost against single occurrence of risk. This is called *Single Loss Expectancy (SLE)* and is defined as below.

$$SLE = EFxAV$$

Where EF is the Exposure Factor and AV is the total Asset Value at risk in terms of cost of asset, cost of information or data residing on the asset, costs incurred due to legal or

statutory violations and damage to reputation and goodwill of organization. This gives the us the loss at single occurrence of the risk. The annual loss that shall occur due to the particular risk is calculated below.

$$ALE = SLExARO$$

Where *ALE* is *Annual Loss Expectancy,* and *ARO* is *Annual Rate of Occurrence* depicting the frequency of risk to occur over a 12 months period. For example, an asset valued at USD 100,000 faces a risk which shall lead to 10% loss to the asset value shall have a SLE of USD 10,000 and if the risk is expected to occur thrice in one year, the organization shall have to face of loss of USD 30,000 in case actualization of risk in a single year[7].

1.1.5 Risk Treatment

Once all risks have been identified and prioritized, the next phase of treating those risks begin to bring them down to acceptable levels. These levels are determined by an organization's risk tolerance and risk appetite. Risk appetite is a broader organizational level of strategic risk that organization is willing to accept while risk tolerance thresholds are defined as the level of risk organization accepts in the objectives or tactics for achieving overall strategy[8]. These levels are decided based on the information from quantitative risk assessment results. If the cost of control to treat the risk is higher than ALE, either a compensating control is sought, or risk is accepted altogether. Enlisted below are few risk treatment methods

a) Risk Mitigation

This technique involves mitigating the risk completely. Organizations deploy security measures and controls to address the risk and eliminate it completely such that the likelihood is reduced to zero. For example, all risks arising from open network ports are mitigated by closure of that port in case it does not serve a basic purpose.

b) Risk Reduction

It is likely that not all risks can be completely mitigated or the cost of deploying security controls to mitigate those risks is higher than the ALE of that risk. In such

a case, organizations implement controls which reduce the value of risk by bringing down either its likelihood or impact such that the residual risk is within the tolerable limits. Residual risk is defined as the remaining risk after a treatment measure has been implemented. For example, the risk of physical intrusion cannot be eliminated but is reduced significantly by deploying fences, barbed wires and access control systems.

c) Risk Transfer

In events that the management of the controls for treating risks is challenging and expected to cause issues to the organization, it is decided to transfer the risk to another party such that the risk does not require treatment by the organization. This is usually done by outsourcing one or more business processes to a trusted third party. The risks associated with the outsourced business process are now transferred to the third party by the organization and it is defined as the former's responsibility to address those risks. For example, the risks arising due to internet outage are transferred to a third-party cloud service provider by obtaining their hosting services.

d) Risk Acceptance

Risk is accepted by the organization if it involves one of the below scenarios.

- The identified risk is within the limits of risk tolerance defined by the organization.
- The cost of treating the risk outweighs the ALE in case the risk is actualized.
- There no suitable control to treat the risk.

1.1.6 Vulnerability and Patch Management

Now having fully understood the role of vulnerabilities in causing harm to organization and its position in risk management process, let discuss the process of addressing these vulnerabilities and why it is important to prioritize the assets for patching the vulnerabilities. As defined in the above section, the mere existence of a vulnerability poses a risk to the entire organization. If that vulnerability is remediated, the chance of

actualization of the associated risk comes down to almost zero. This is done by applying patches to close out the vulnerabilities and loopholes in the infrastructure.

Vulnerability and Patch Management are now considered an essential responsibility of the IT team[9]. In the past, it was considered as more of a part time role of the IT department and patching was done based on availability of time rather than necessity. Since the threat surface has now increased exponentially, NIST calls for a formation of a specialized patch management group whose efforts are solely dedicated for vulnerability and patch management[10]. It goes further to enlist the recommended strategy for patch management.

Most of the organizations come to know about the vulnerabilities from external threat sources such as National Vulnerability Database (NVD) or US Federal Government's Computer Emergency Response Team (CERT)[11][12]. The next step is to identify which assets within the organizations are vulnerable. This is usually done with help of automated network vulnerability scanners. Once a list of vulnerable systems is obtained, the manual effort begins. Prioritization of the list is of utmost importance to address more critical systems and ones that have higher business dependency than others[10]. The next effort is of testing the patches to ensure no potential impact on business applications is encountered. After that patches are rolled out in production. Below figure describes the patch management process.

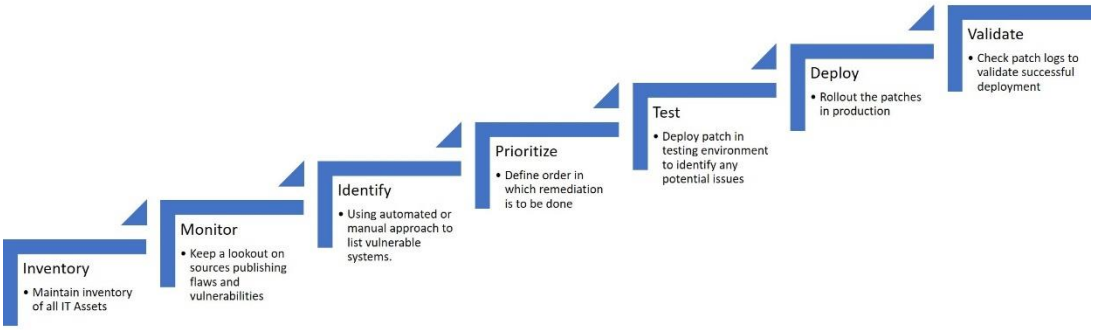


Figure 1-2 Patch Management Process

1.1.7 Prioritization of Patching

In a small infrastructure containing only a few assets, exhaustive patching might be viable but as the infrastructure grows, patching entire network at once is a very challenging tasks which comes with its own risk of bringing down the entire production. Secondly, the limited resources for patching might make it a very lengthy process to deploy patches across complete infrastructure. Furthermore, deploying patches requires and planned downtime window. While deployment of patches is necessary but provisioning of uninterrupted services is of top priority, therefore the provided downtime in most of the cases is not enough to patch all.

1.1.8 Common Vulnerability Scoring System (CVSS)

In the past, all organizations used different methods to score the vulnerabilities for determining severity and had their own scales to weigh them. Thus this led to the problem of same vulnerabilities being scored differently by different methods. The Common Vulnerability Scoring System or CVSS was introduced by NIST in 2005 as a first-generation open scoring system and is now most widely used by information

security experts and researchers for ranking vulnerabilities[13]. The introduction of CVSS has provided the below benefits.

The way CVSS works is that against each vulnerability it has defined three type of scores; base scores, temporal and environmental scores. The base scores are calculated using base variables such as Confidentiality Requirement, Integrity Requirement, Availability Requirement, Attack Vector, Attack Complexity, requirement of user interaction, requirement of privileges or changing scope. The temporal variables allow calculation of temporal scores on top of the base scores. These variables include the availability of official fix, the availability of exploit etc. The third score which is the environmental scores are calculated by modifying base requirements and generate organization based scores.

a) Standardization

With introduction of CVSS, all vulnerabilities whether they are platform related, network related or applications, they are rated on a same scale and in a same manner. This allows experts to formulate a single vulnerability management policy and rank all the vulnerabilities using same set of scores.

b) Contextual factors

Along with the base metrics, CVSS allows customization of scores on temporal factors such as official fix publish and existence of a published exploit. Not only this CVSS also allows organizations to adjust the scores based on the asset details and requirements within the organization.

c) Public Framework

The entire working of the CVSS including its scoring criteria and formulas for calculating scores has enabled organizations and researchers to fully understand the working behind the calculation of those scores. Not only this, it has allowed organizations to include CVSS in their own vulnerability management systems.

1.1.9 Satisfiable Modulo Theory (SMT)

Since we have made used of SMT solvers to find solution of prioritizing patching problem under organizational constraints, therefore we feel that this area should be explained.

The research into SMT started in 1990s but it was already been discussed for as early as late of 1970s. In the field of artificial intelligence and decision making using formal methods, the applicability of SAT or Boolean Satisfiability formulas cannot be overlooked[14]. This is used to find out an interpretation which might satisfy a Boolean formula. However, the problem with SAT is it does not take into consideration background data or assertions. This has led to the development of Satisfiable Modulo Theories or SMT which makes use of background context rich information to find a solution which shall satisfy a formula or a set of formulas under a given or set of given condition in more expressive logics. SMT solvers are basically SMT procedures[15].

One of the efficient SMT Solvers developed by Microsoft Research and released in 2007 is known as Z3[16]. The tool is aimed to address problems in software verification and analysis. It is available online and its APIs are developed and made available on GitHub to allow integration with different platforms[17][18].

1.2 Motivation

The reliance on information systems for processing, storing and transmitting data is increasing day by day. Advances in cloud computing, mobile devices technology and fast internet speeds has eliminated the need of information to exist physically. From citizenship documents to currencies, almost everything is now made available in electronic form. Organizations have been finding ways to leverage technology to make their offices paperless.

Statistics show a 38% increase in five-year trend and 14% increase in one year in the amount of detected vulnerabilities[19]. Vulnerabilities being disclosed at such rapid rate and the impact of their exploitation has left the organizations in a bit of trouble. Although 86% vulnerabilities have a patch available on day of disclosure, yet organizations constantly fall behind in timely deployment of patches and are left at mercy of the cyber criminals[19].

The motivation behind this research stems from a real-world problem. In 2017, a real-world organization named ABC Inc. (to protect anonymity) faced an uphill challenge of patching more than 15,000 vulnerable servers and endpoints when a Microsoft Windows vulnerability was targeted worldwide in its SMB protocol by a ransomware which had hit more than 230,000 computer systems resulting to loss of data and availability of services[20] [21]. Since ABC Inc. deals in customer support services and processes sensitive data on behalf of customers, it had a greater threat from the ransomware that if exploited could cause exposure to sensitive information as well as compromise to availability of its business which was deemed catastrophic for business by one of its senior officials. Therefore, they had to protect their assets against the attack but given the nature of business could not afford a downtime of more than a few hours. Given the limited amount of resources, they were facing a problem of patching the right assets which could limit their exposure for time being and then continue patching in their periodic cycles. Thus, the information security team was given the task to come up with an approach that would protect their most valuable assets and limit their exposure to minimum possible value under the given resources and strictly within the given downtime window.

The initial approach was to proceed with CVSS rating but the rating could not determine that actual risk to asset based on its existing configurations. Moreover, it could not provide an optimal answer based on the constraints faced by the organization. Thus, a need was felt to find a feasible solution to this problem in form of research and development of an application.

1.3 Research Question

The outcome of this research is expected to enable the intended audience to figure out answers to below questions:

- Which of the vulnerable assets in the infrastructure are more susceptible to the vulnerability exploit than others?
- Are the existing security measures of any help to provide defense against exploitation of the said vulnerability? If so, to what extent?

- In a fixed time window and with available resources, how many of the vulnerable assets can be patched?
- What can be done to increase the number of assets that can be patched?
- Is it possible to ensure high availability for business critical assets during patch deployment such that the services remain available for these assets?
- Which of the vulnerable assets should have top priority for being patched against the vulnerability?

1.4 Problem Statement

Risk and Patch prioritization has been a concern for organizations for a large period of time. A numerous amount of efforts have been made to provide a solution which is practical as well as can accommodate every organization's business model and their patching constraints. These constraints render exhaustive patching infeasible. NIST frameworks SP800-30, SP800-40 and others like ISO 27005 talk about prioritizing risks and patches but fail to provide a quantitative method for prioritization. CVSS was an initiative in this direction however despite multiple revisions it has still not been able to address the organization related contextual factors and the challenges faced by the responsible personnel in deploying those patches. Therefore, a solution must be provided which not only considers contextual factors but also incorporates challenges faced by organization to deploy patches.

1.5 Goals and Objectives

This study aims at providing a solution to application developers, network and system administrators which shall enable them to prioritize their patching efforts against a specific vulnerability by determining the exposure of asset against a vulnerability and its value to the organization both in monetary terms and with respect to confidentiality, availability and integrity.

Objectives include establishing exposure of assets against a vulnerability after determining the existing security controls and configurations of the asset and also prioritize the asset based on its value to the organization and nature of the data residing

on it. Last but not the least, prioritize assets for patch deployments based on the challenges and constraints faced by the organization.

1.6 Audience

This study is targeted at all those personnel who are in charge of remediating the vulnerabilities in their applications, systems and networks and deploying patches while meeting the organizational and production challenges. These shall include but not limited to application development teams, project managers, information security architect and consultant, server administrators, cloud administrators, network managers and architects etc. Using this study, they shall get a list of the organization's crown jewels which are exposed by a vulnerability and are required to be patched at priority than the rest of the network.

1.7 Scope of Study

This research assumes that organization is already aware of the vulnerable assets. A vulnerability assessment has been run and it has been established which assets are vulnerable and need patch deployment. The research also assumes that the organization has determined value of its assets in terms of finances, the criticality of data residing on the process, its importance to business operations and legal, statutory and compliance issues arising due to exposure of data residing on the servers. Keeping in mind these, the organization has assigned a dollar value to these servers and has rated them from Low to High in terms of confidentiality, integrity and availability. It further assumes that organization has established primary and secondary assets in terms of high availability and is aware of the patching resources and has secured a downtime for the patching of the assets.

This research focuses on the vulnerabilities present in the NIST National Vulnerability Database (NVD) and have been assigned a CVSS vector and a base score and that the vulnerability's patch have been released by the official distributor. The study neither recommends which patch to deploy and nor takes responsibility for efficacy of the deployed patches and should any adverse impact come to the asset as a result of deployment of the patch. All patches should be tested and verified before roll out to production.

1.8 Organization of Thesis

This thesis has been organized in form of six chapters for the sake of easy comprehension and study of the factors involved.

Chapter-1 "***Introduction***" opens by providing background and knowledge about origin of need of patching vulnerabilities within assets and prioritizing them before patching. It further goes to discuss the CVSS scores and their viability to ease down patching efforts. This is followed by motivation, problem statement, research questions and the scope of the work.

Chapter-2 "***Literature Review***" provides the insight into the challenge of patch prioritization and the efforts of research community to provide various models, ontologies and solutions to solve this problem. This lays out the support required for this work.

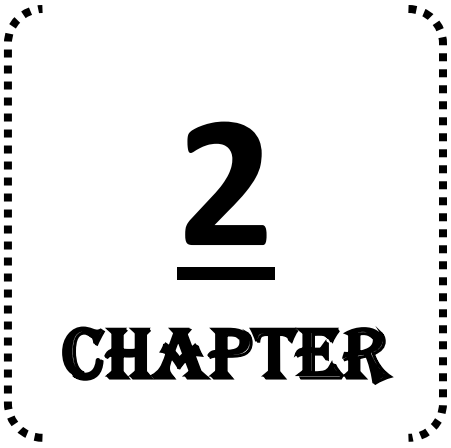
Chapter-3 "***Requirements Gathering***" outlines the functional and non-functional requirements gathered to produce this solution and the architecture, design, flows and use cases for the tool developed to provide the solution.

Chapter-4 "***Research Methodology***" discusses the implemented research cycle and the adopted methodology and gives details of the frameworks designed to conduct experiments.

Chapter-5 "***Experimentation and Analysis***" discusses various outputs provided against the vulnerabilities and the asset details and analyses the results to demonstrate that the provided solution if the optimal one under given set of constraints.

Chapter-6 "***Conclusion and Future Prospects***" enlightens the concluded results of this work and highlights the future research directions.

"***References***" encloses the bibliographic sources of supported literature of this research work.



2 LITERATURE REVIEW

2.1 Introduction

This chapter covers the research work pertaining to this thesis and the contributions made by the research community. It provides support to our study and highlights the importance of the study and the concerns of the research community in the field. The following include the various methodologies, approaches and ontologies proposed for prioritization of vulnerabilities.

- Section 2.2 Vulnerability Prioritization, Mitigation and Management
- Section 2.3 Risk-based vulnerability prioritization and management
- Section 2.4 Quantification of security risks based on CVSS

2.2 Vulnerability Prioritization, Mitigation and Management

According to Farris, Shah et. al in 2018 [22], mitigation of all vulnerabilities found by a scanning large enterprise networks is not realistically possible between periodic scans. They build the foundation on inability of the CVSS to provide contextual information and argue that the vulnerabilities lacking easy-to-implement solutions or those which do not have patches available need more advanced and systematic approach to be addressed. Since the reports generated by vulnerability scanners are too large and complex to be addressed by human, most of the times the resources required to patch are not allocated as per the need which results in high vulnerability exposure and less than enough resource allocation. Thus, there is a very strong need of

prioritizing vulnerabilities. Building on this hypothesis, they propose a mixed-integer multi-objective optimization algorithm for effort prioritization while patching vulnerabilities. They term this as "VULCON" (short for Vulnerability Control). This proposed approach takes input in the form of vulnerability scan metadata form tools such as Nessus, data from "National Vulnerability Database (NVD)" and evaluates two factors; 1) "Time-to-vulnerability remediation (TVR)" which is defined time frame between identification and mitigation of vulnerability and 2) "Total Vulnerability Exposure (TVE)" which they have defined as the aggregated combination of vulnerabilities density from previous month and the new vulnerabilities. This approach prioritizes the vulnerabilities based on the above metrics such that both the metrics are not allowed to go beyond a limit defined by organization and based on available personnel-hours.

The objective of this paper is to provide organization's "CSOC (cyber-security operation center)" to monitor the exposure caused from unmitigated vulnerabilities and the time since their existence. This is to enable the CSOC to deploy remediation of vulnerabilities which are identified during periodic scans and are awaiting patch deployment for a time greater than defined by the organization. However, this approach does not provide a solution for the patches published on the day a critical vulnerability is disclosed. Such circumstances require selective and optimized patching efforts which are outside the routine periodic scans and patching procedures. The only organizational constraint is the personnel-hours available. Although they have input data from NVD and mission critical services, still the context rich information is not considered for the optimization. The organization's requirements for Confidentiality, Integrity and Availability are not factored in and the knowledge of existing security controls is lacking. These factors might lead to calculations of more severe results than they actually are.

2.3 Risk-based vulnerability prioritization and management

As per approach proposed by Bambos and Miura-Ko in 2007[23], it is not possible to prioritize vulnerabilities and estimate their associated risks without taking the network

topology in account. Since the development and testing of patches is time consuming effort, therefore deployment of patches within entire infrastructure might be a lengthy process and also because some systems might not be available during regular patching cycle, this might slow down the process and require more human effort. If an attack takes place or a virus outbreak happens during the process of patching, the results could still be drastic.

The authors have classified prioritization efforts into three categories; 1) Density Prioritization where each asset within the entire infrastructure is taken as a network node. Each node is then prioritized based on the number of neighbors or connected nodes they have 2) Source Prioritization involves allocation high priority to nodes which might be the potential entry point of an attacker or a malware. These should be mitigated with utmost priority and 3) Type Prioritization is the standard method used by organizations by classifying vulnerabilities into high, medium and low risk and then patching high risk vulnerabilities first.

A method called 'SecureRank' is proposed which prioritizes patching of vulnerabilities by assigning the neighbors of the nodes a vulnerability score product based on the exploitability and prevalence of the vulnerability. It proposes to start with the highest risk node. While this approach is useful for small networks, but within a network having thousands of nodes and given the patching constraints, the proposed approach might be too much resource intensive with little results.

2.4 Quantification of security risks based on CVSS

As per Singh & Joshi in 2016[24], the current CVSS scoring methodology views each vulnerability in isolation and does not consider attack interdependencies. Therefore, it is imperative that a technique be devised to quantify security critical intrinsic element for network-based risk level estimation calculations. The prioritization of identified vulnerabilities classified by their security vulnerabilities is essential for applying remediations to provide acceptable level of security assurance. The CVSS scores lack definition of specific network configurations although the vulnerability severity has impact on the security risk of the network. Thus the information about network architecture and design and the vulnerabilities affecting it are essential factors to

consider for successful future predictions of events and the likelihood of occurrence of the vulnerabilities.

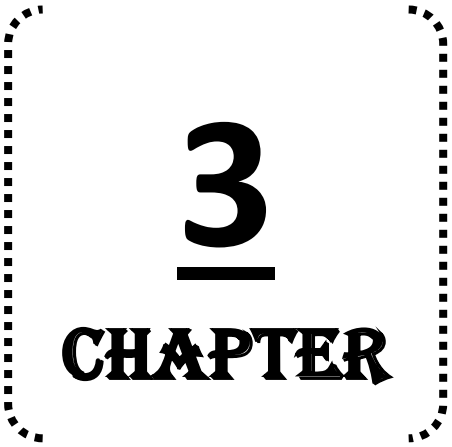
A new metric termed as "Hazard Metric (HM)" is defined by the authors. It determines the likelihood of probability of exploitation of vulnerabilities in the network and comprises of sub-metrics such as "Maturity Level", "Frequency of Exploit", "Exploitability Impact", "Amendment Level" and "Authentication Level". While the first metric is defined as the ration of number of days from discovery of vulnerability and the date of exploitation in network, the other metrics are based on computations and calculations of various CVSS metrics.

Furthermore WU, WEN & ZHANG in 2019[25], proposed a revised systems based on CVSS scoring for improvising the dispersion of vulnerability risk scores. The existing CVSS scoring scheme has some issues with objectivity and dispersion of vulnerability scores where objectivity is referred to as how well the results reflect the nature of the practical samples and dispersion being how much the scores are distributed and what degree of variation lies in that score. This allows accumulation of same scored vulnerabilities and organizations are left in a disarray as to how to classify them based on severity.

The authors proposed a new scoring scheme and termed this as "CVSS_PCA" based on Principal Component Analysis. The solution aims to resolve the dispersion problem with existing CVSS scores by equilibrating the probabilities of CVSS metrics, increasing the number of metrics and reducing the correlation or interaction between the CVSS metrics. This new scheme is applied on forty thousand plus vulnerabilities and compared against CVSS 2.0 and VRSS 2.0[26] which Liu and Zhong introduced in 2011. The result was that better dispersion values were achieved by CVSS_PCA as compared to CVSS and VRSS.

In 2015, Younis and Malaiya[27] examined 813 vulnerabilities in Microsoft products "Internet Explorer" and "Windows 7" and evaluated them based on CVSS and Microsoft rating approaches. The made a comparison between an expert opinions approach (CVSS) and technical rating system (Microsoft). CVSS does the rating based

on exploitability and impact while Microsoft rates vulnerabilities based on impact and vulnerability. They concluded that in their default nature both rating systems have high false positive rates and Attack Vector (AV) exploitability metric has higher influence on the score as compared to others.



3 REQUIREMENTS GATHERING

3.1 Introduction

In accordance with IEEE 29148:2011(E) standard for software requirement specification, this chapter describes in detail the elicited requirements and the functionality required from the solution. The requirements were gathered from a company ABC Inc. (pseudo name to entertain anonymity requested by the organization) which is a large customer services providing organization having global presence and an infrastructure comprising of 800 servers and a total of 15000+ endpoints.

3.1.1 Purpose

The purpose of this document is to detail the process for gathering, analysis and refining the requirements for designing and development of the software to provide solution to the discussed problem of asset prioritization for patching of vulnerabilities faced by stakeholders in ABC Inc. This document also goes on to show that trace of the refined requirements for original gathered requirements from ABC Inc. which were vague and ambiguous.

3.1.2 Scope

This solution is a Java desktop application which can be installed on any Java Runtime Environment supported machine.

3.1.3 Glossary

This subsection lists down all terms, abbreviations and acronyms used in this chapter.

- NVD : National Vulnerability Database maintained by NIST.
- API: Application Programming Interface
- CVSS: Common Vulnerability Scoring System
- CSV: Comma Separated Values

3.1.4 References

- IEEE 29148 :2011(E) Systems and software engineering — Life cycle processes — Requirements Engineering[28]
- Custom Software Requirements Specification Document Example - Belitsoft[29]

3.1.5 Overview

This document lays out the Software Requirements for the proposed tool "Asset Vulnerability Management" in a modified IEEE 29148:2011(E) format. Note that few sections of the document which are already covered in other chapters such as audience etc. are omitted to avoid redundancy. Below is the general outline of the document.

- Section 3.2: General Description of product and associated interfaces, screenshots and requirements
- Section 3.3: Lists down all functional and non-functional requirements. Unlike IEEE 29148, all requirements are detailed under a single category.
- Section 3.4: details the requirements gathering process.

3.2 General Description

The application is designed to provide a solution to those responsible for patching vulnerabilities under a set of constraints. The application is designed to obtain

organizational inputs such as list of vulnerable assets and their characteristics, the vulnerability selection from NVD, organization patching constraints and provide a prioritized list of assets which shall serve as the optimized list for patching vulnerabilities under the set of given circumstances.

This intended use of this application is to aid server administrators, network administrators, application developers, database administrators and information security and governance team within an organization to make timely decisions for prioritization and patching of the vulnerabilities.

3.2.1 Product Perspective

The Asset Vulnerability Management application is a standalone asset prioritization system for which the users shall be required to have a Java enabled environment to make use of the system. Since Java is available free of cost and can run on a variety of platforms and devices, this makes the application cross-platform supported. This will enable the users to leverage the use of application and its intended functionality without investment in any other piece of software since Java is available for install on virtually any devices around the world.

3.2.1.1 System Interfaces

The system shall make use of various hardware and software interfaces as listed below, it shall require standard interfaces to work with installed hardware. As said the application is a standalone desktop piece of software which shall require no communication with internet or the organization's network. However to import information into the application, a network or internet connection might be required. The below system interfaces shall be required for the application.

- Connection to MySQL database containing asset and vulnerability information.

3.2.1.2 User Interfaces

All user interaction with the application happens through a Java Runtime software.

3.2.1.3 Hardware Interfaces

No hardware interfaces other than standard I/O interfaces are required for this application.

3.2.1.4 Software Interfaces

The application shall require below software interfaces:

- Interface with API calls for calculating CVSS scores of vulnerabilities.
- Interface with API calls for using Z3 SMT Solver for prioritization of assets.
- Interface with CSV files imported from vulnerability scanners to import asset list.
- Interface with JSON feeds from NVD for importing vulnerabilities.

3.2.2 Product Functions

The gathered functional and non-functional requirements from ABC Inc. were analyzed and refined and all ambiguities were resolved. The resolution was deviated to some extent for a limited set of certain requirements provided by ABC Inc. This section lays down all identified and agreed upon requirements of the solution.

3.2.2.1 Enterprise Requirements

Below section outline the requirements presented by the stakeholders from ABC Inc. These stakeholders are defined in section 3.4.

a) All users can perform same actions.

Since it is a standalone solution, each user who shall desire the use of the application shall have the ability to create his own instance on his machine where he shall be permitted to access all application functions without any restrictions.

b) Users can bulk import assets and customize them.

Users shall have the option to bulk import vulnerable assets in the application and customize their characteristics either prior to import or after importing.

c) Users can specify existing network security configurations.

Against each asset, users shall be able to provide information about the contextual network information such as and position of asset within the network.

d) Security configuration shall influence exposure of asset.

The security configurations defined for each asset shall determine the actual exposure of the asset in case vulnerability is exploited.

e) Backups can be marked and identified during patching

While producing list for patching, assets having high availability requirements shall have their backups entered into the system such that the primary and secondary servers cannot be patched in the same cycle.

f) User shall be able to search for the specific vulnerability.

The selection of vulnerability against the assets shall be done by searching through the repository and the user is given the option to update the repository before obtaining the list.

g) The vulnerability repository shall contain six months of data.

These user shall be able to select data from last six months from the vulnerability repository.

h) The vulnerability scores shall be adjusted.

These scores against vulnerabilities shall be adjusted depending upon the place of the asset within the infrastructure, its existing security controls and the confidentiality, integrity and availability requirements.

i) User shall be able to establish patching constraints.

Constraints such as available resources, simultaneous patching options, time to patch each server and maximum allowed maintenance window shall be available to the user for input.

3.2.2.2 Constraints

The application is to be developed within the below specified constraints while remaining within its bounds. These constraints are derived from multiple functional and non-functional requirements specified in this chapter while others exist as per requirements specified by stakeholders, the awareness of which is crucial for all stakeholders during development of the application.

- Since the application is developed to have a standalone instance, this shall provide full functionality to the user by default.

- Application is to be developed as a standalone desktop software using Java to enable cross-platform support.
- The data must be stored in a localized instance of a relational database for robust searching and management.
- The application shall have a GUI for ease of use and error-free execution.
- The application shall allow addition of assets by inputting directly into GUI and import list of assets.
- For importing list of assets, the list has to be formatted in a CSV format with pre-defined order of data for successful format.
- The application shall allow modification to current list of assets.
- The application shall allow update of vulnerabilities database prior to execution via JSON feeds downloaded from NVD website.
- The application shall allow user to adjust and modify the patching constraints and provide optimal patching assets list.

3.2.2.3 Assumptions and Dependencies

- Application shall be installed on a supported Windows Operating System, Java 8 or higher and MySQL 5.x.
- The user shall be responsible for correctness of data imported via CSV or JSON format.
- Updating the vulnerability repository will clear the repository and replace with the ones imported.
- There are no security requirements for access control or encryption of data.

3.3 Specific Requirements

This section specifies the detailed requirements that the application shall meet.

3.3.1 External Interfaces

All user interfaces are laid out in this section. All user interfacing is done through a Java Swing UI forms. These UI forms are shown below.

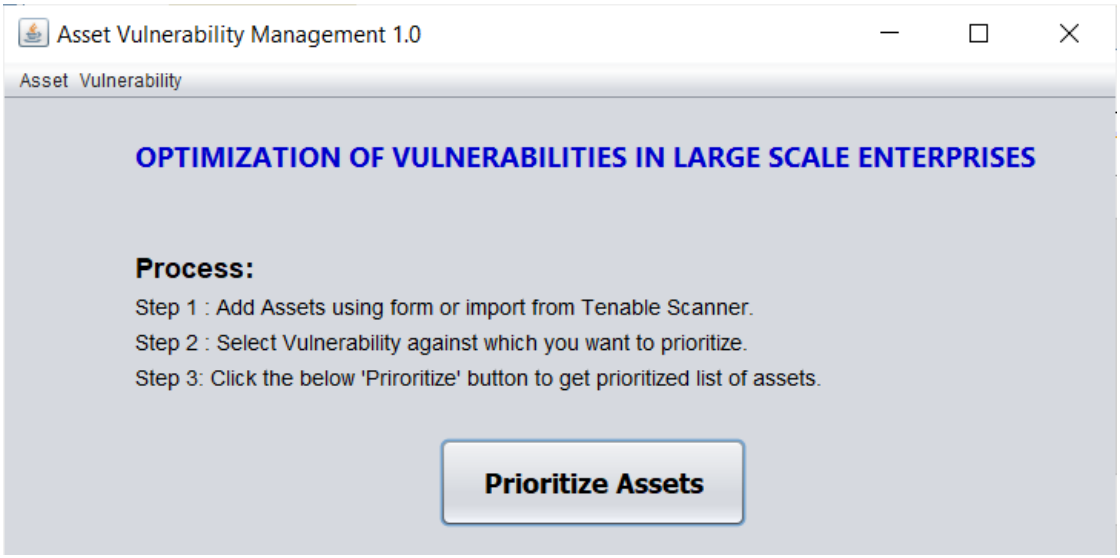


Figure 3-1 Home Page of Java GUI

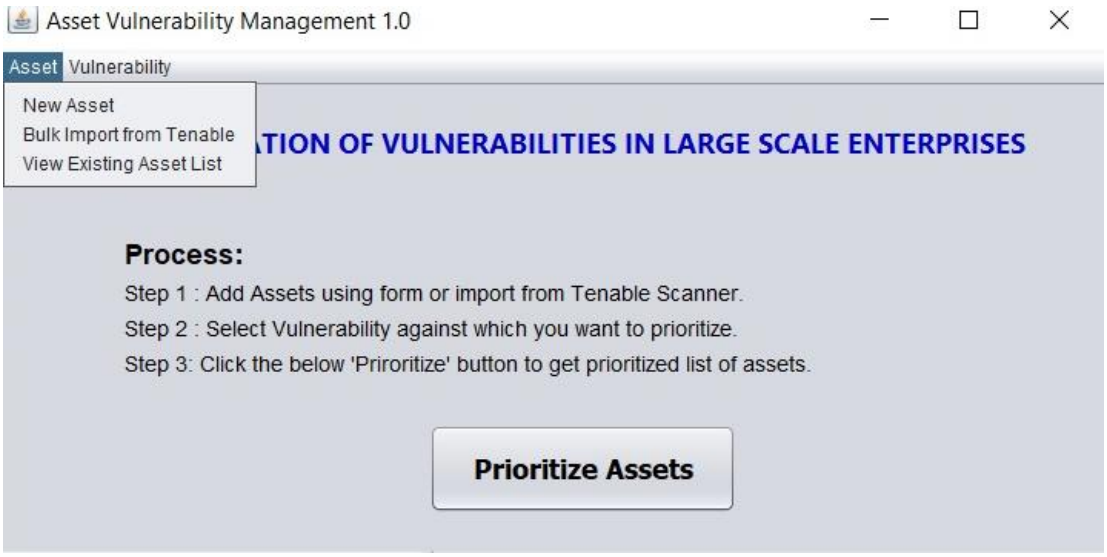


Figure 3-2 Asset Menu

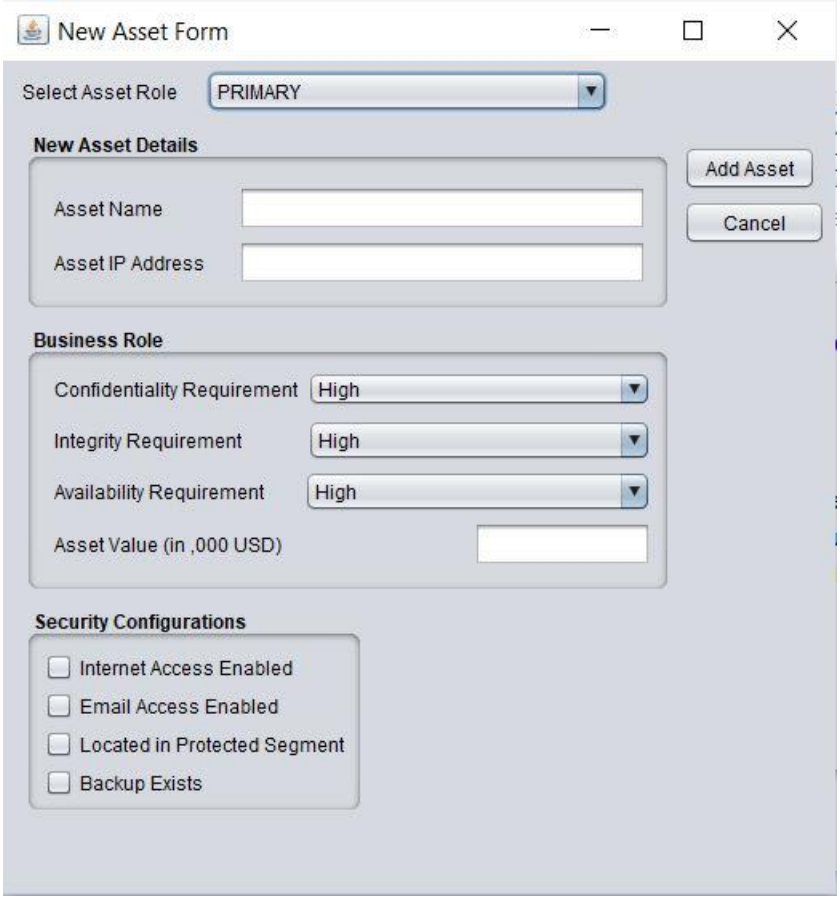


Figure 3-3 New Asset Form

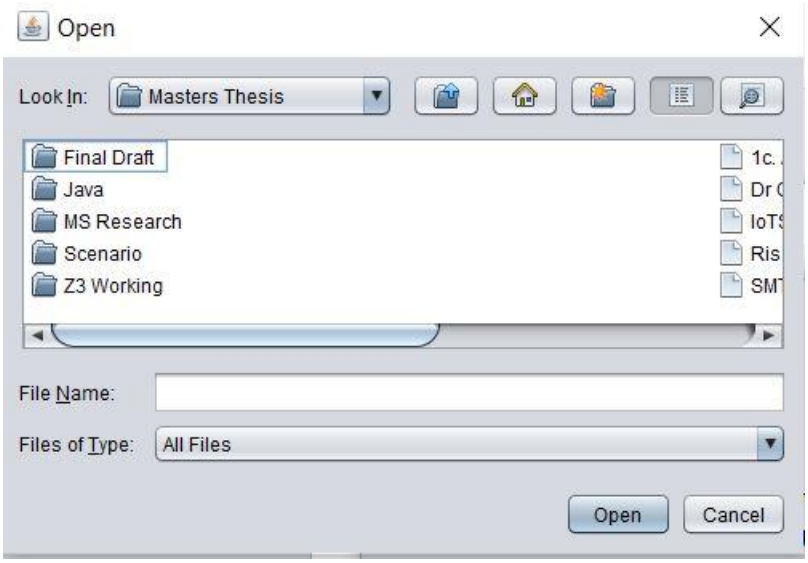


Figure 3-4 Bulk Asset Import via CSV file

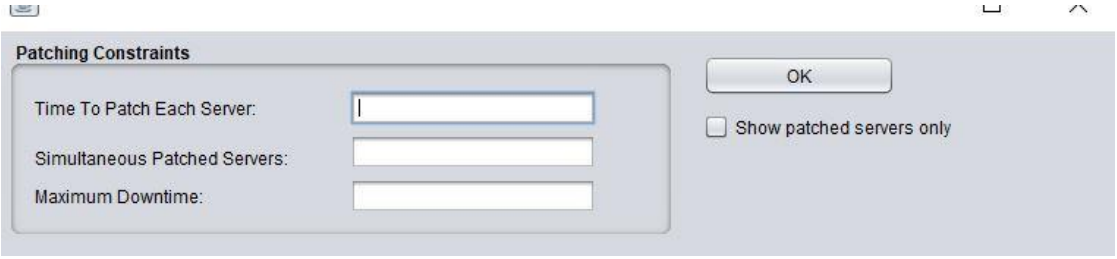


Figure 3-5 Input of Patch Constraints

3.3.2 Functions

For understanding the functional requirements, the specific use cases are detailed in the following sub-section for understanding of system behavior and the specific requirements which branch out from the use cases.

3.3.2.1 Use Case: 1 Add Asset to Repository

a) **Goal Context:** The user has been shown New Asset Form. Their goal is to input the details of the asset and the asset is successfully added into the repository and user is notified of the success or if not then user is notified of the reason.
b) **Scope:** Asset Management System
c) **Level:** Primary Task
d) **Preconditions:** None
e) **Success End Condition:** Asset has been successfully added to the repository and user has been notified.
f) **Minimal Guarantee:** User is notified whether the action is successful or not.
g) **Primary Actor:** User
h) **Trigger:** User opens New Asset Form

3.3.2.2 Use Case: 2 Import Asset List from CSV

a) **Goal Context:** The user is asked to browse bulk list of assets from local directory in CSV format and the list is successfully imported into the system.
b) **Scope:** Asset Management System
c) **Level:** Primary Task

d) **Preconditions:** The csv file exists on local file system and contains data in a specific format to be input into the database correctly.
e) **Success End Condition:** List is successfully imported into the database.
f) **Minimal Guarantee:** User is notified whether the action is successful or not.
g) **Primary Actor:** User
h) **Trigger:** User opens file browser for importing CSV.

3.3.2.3 Use Case: 3 Select vulnerability from database

a) **Goal Context:** The user searches for existing vulnerability from the database and if found selects the vulnerability and the vulnerability is assigned to each asset and scores are adjusted as per security configuration of the asset.
b) **Scope:** Vulnerability Management System
c) **Level:** Primary Task
d) **Preconditions:** Vulnerability exists in the database.
e) **Success End Condition:** Vulnerability is assigned to each asset and user is notified.
f) **Minimal Guarantee:** User is notified whether the action is successful or not.
g) **Primary Actor:** User
h) **Trigger:** User opens Select Vulnerability menu.

3.3.2.4 Use Case: 4 Import vulnerability from NVD JSON Feed

a) **Goal Context:** The user is shown file browser window for locating the JSON feed and uploading it to vulnerability database.
b) **Scope:** Vulnerability Management System
c) **Level:** Secondary Task
d) **Preconditions:** JSON feeds have been downloaded from NVD website.
e) **Success End Condition:** Vulnerabilities have been successfully imported.
f) **Minimal Guarantee:** User is notified whether the action is successful or not.
g) **Primary Actor:** User
h) **Trigger:** User opens Import Vulnerability form.

3.3.2.5 Use Case: 5 User obtains prioritized asset list.

a) **Goal Context:** The user inputs patching constraints and successfully obtains list of assets he should patch and should not patch.

b) **Scope:** Asset Vulnerability Management

c) **Level:** Primary Task

d) **Preconditions:** Assets exist in the database and vulnerabilities have been assigned to the assets.

e) **Success End Condition:** Prioritized asset list is provided to the user.

f) **Minimal Guarantee:** User is notified whether the action is successful or not.

g) **Primary Actor:** User

h) **Trigger:** User click 'Prioritize Asset'.

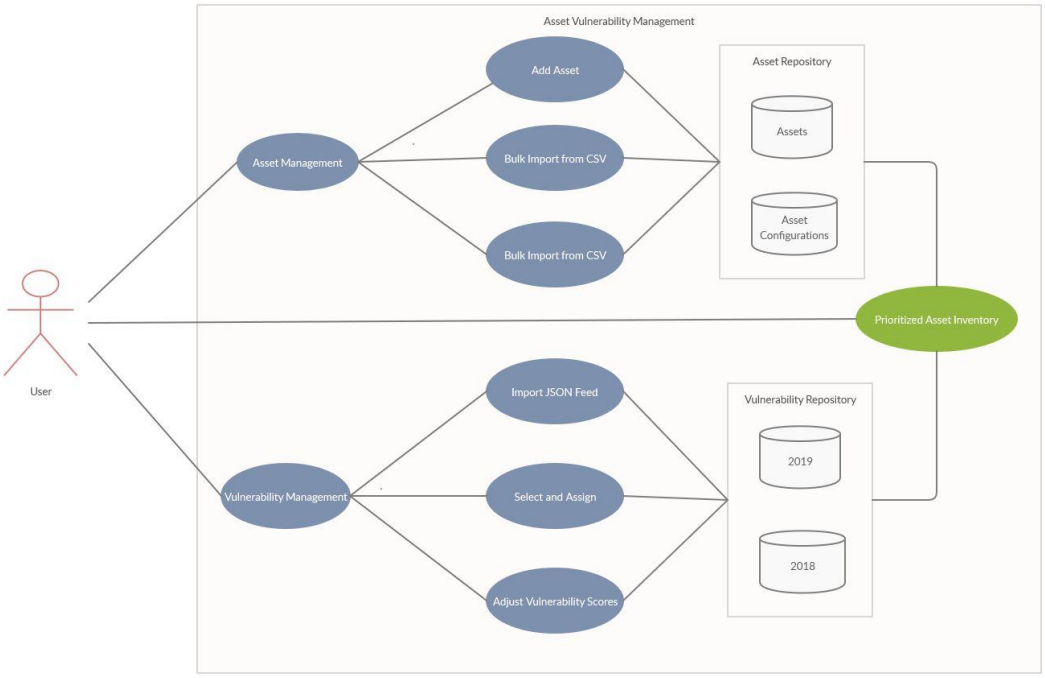


Figure 3-6 Use Case Diagram

3.3.3 System Functional Requirements

Below are the function requirements of the applications

3.3.3.1 Allow user input of asset value and contextual information within organization.

The application shall enable the user to provide the asset value and the placement of asset within the network to specify any existing security settings which might influence the overall exposure of the asset against the vulnerability.

3.3.3.2 Value of asset to the organization shall and its security requirements shall be factored.

The dollar value of asset within the organization, and the confidentiality, integrity and availability requirements of the assets shall be the criteria coupled with the exposure of asset for prioritization of vulnerabilities.

3.3.3.3 High availability requirements for assets shall be addressed.

Assets requiring 24/7 uptime and high availability shall have a special provision within the solution. The application should not propose a solution which includes both primary and secondary servers within same patching window.

3.3.3.4 Total patching time shall not exceed maximum allowed downtime.

The application should not propose any such solution in which the patching time shall increase the allowed maintenance downtime. Similarly, it also should not propose a solution where more number of assets can be patched within the maintenance window than the ones proposed.

3.3.3.5 The total asset value of all assets proposed for patching shall be maximum under the set of constraints.

The application should provide a list of assets under the given constraints such that the combined asset value of all assets being patched is maximum and no other valid combination exists such that the proposed value is less than that combination.

3.3.3.5 User shall be allowed to update vulnerability repository either automatically or manually.

There shall be a provision within the application to update vulnerability repositories at any given time.

3.3.4 Non - Functional Requirements

Enlisted below are the non-functional requirements. However the implementation of all of these is not necessary for the first release of the solution and some of these can be catered for in the subsequent updates and releases.

3.3.4.1 Centralized repositories of assets and vulnerabilities shall be created.

In order to decrease the manual effort to add assets into the tool, the repositories shall be centrally maintained with appropriate user roles and rights over the repositories.

3.3.4.2 View and modification of assets shall be provided to user.

The user can view or modify existing assets.

3.3.4.3 Application is directly linked to vulnerability scanners for importing assets.

The application shall be made able to directly fetch asset list automatically upon finishing of a scan instead of exporting from scanner first and then importing.

3.4 Requirement Elicitation Process

This section describes the stakeholders and their roles and the process used to analyze the requirements. Moreover, it further lists down the discovered issues and their resolution.

3.4.1 Analysis Stakeholders and Roles

Below are the stakeholders and their roles. The names have been replaced with arbitrary names to protect anonymity as requested by the stakeholder.

- R. Ahmed – Principal Server Administrator ABC Inc.
- W. Ashfaq – Director Application Development ABC Inc.
- A. Hussain – Principal Network Administrator ABC Inc.
- M. Hassan – Senior Cyber Security Architect ABC Inc.
- M. Kamal – Requirements Engineer

3.4.2 Analysis Process

The problem faced by each of the stakeholders was understood and the subsequent requirements discussed. Each of the Enterprise Requirements was obtained from the stakeholders and then analyzed and refined to fix necessary issues. The requirements were then further refined to ensure consistency, completeness and testability. Also, the research work within the requirements was established.

4 RESEARCH METHODOLOGY

4.1 Introduction

Research is defined as the careful application of specific tools and study pertaining to a problem or concern in a scientific manner[30]. This chapter mentions different aspects of research process, in below sections.

- Section 4.2 Research Cycle Followed in This Study
- Section 4.3 Steps of Research Process
- Section 4.4 Gather User Required Input
- Section 4.5 Evaluate vulnerability scores using CVSS
- Section 4.6 Relate Total Asset Value and CVSS Scores
- Section 4.7 Solve prioritization problem using Z3 Satisfiable Modulo Theory

4.2 Research Cycle

A research cycle outlines step by step processes performed for doing a research. During the cycle, every step has its own techniques for analyzing the data and producing the desired results for achieving the objectives. In this study, we make use of the below research steps:

- Defining problem statement
- Analyzing literature review

- Gathering the requirements
- Research Design
- Experimentation and Analysis of results
- Validation of results and findings

A graphical representation of whole cycle is given below.

4.3 Steps of Research Process

In this section we shall discuss all the steps of the research cycle with reference to this study. We shall also discuss the sub-steps contained in each of these for better understanding. This shall be aided with comprehensive details, visual diagrams and tables.

4.3.1 Problem Statement

Every research begins with determining the research domain. This is followed by researcher's pursuit for finding existing concerns of the research community and the unaddressed issues in the research done by the fellow researchers. After all this effort, the researcher is able to obtain a list of areas that needs further work or enhancements to provide better solutions. The end-result is that the researcher arrives at a problem which needs implementation of scientific and research methods to produce a viable solution which effectively addresses the concerns of the research community[31].

However, finding the solution needs complete understanding and knowledge of not only theoretical but of practical aspects as well for finding out a feasible and practical solution that addresses the concerns in a purposeful manner.

The problem statement for this study is detailed in Chapter 1, however we feel that the five W's of questions and problem solving must be addressed for better understanding[32]. We address these as below.

- **WHAT**: The goal of this research is to find a solution for patching problem which enables the organizations to deploy patches and secure their most valued assets in a way that is tailored and suitable to the organization's model and needs.
- **WHY**: To provide organizations a true and context-aware picture of the existing risk due to newly found vulnerabilities and the optimal method of mitigating those risks under a specific list of constraints.
- **WHO**: All server, network, database administrators, application developers, cyber security experts and those in charge of management of the vulnerabilities on their assets.
- **WHERE**: The research was conducted in KTH-Lab located in IAEC, National University of Sciences and Technology.
- **WHEN**: From time t1 of publish of an official patch to the time t2 when all vulnerable assets are patched.

4.3.2 Literature Review

A literature review provides a background knowledge of related surveys, books, research papers, journals and scholarly articles showing the existing work done in the domain of research like yours. It also strengthens your proposed study by demonstrating the importance of the problem and that the proposed solution is better than the existing ones.

There are many advantages of a strong literature review. It can allow insight into the interpretation of previous material and how the new one pans out in light of the old one. It also allows linkage of the proposed study to existing approaches and helps identify new ways to interpret existing studies[33].

The below steps were taken to build foundations of literature review for this research.

a) Search online articles, journals and papers globally and locally.
b) Select 25 papers of the pertinent domain.
c) Build an online library for easy tracking of those papers.
d) A quick analysis of abstract, introduction and conclusion of each paper.
e) Further shortlist papers based on relevancy.
f) A quick go through for summarizing the discussed topics in the papers.
g) Analysis and comparison of all summaries.
h) Based on the summaries, we select 5 most relevant papers.
i) Comprehensive and in-depth understanding of the final papers.
j) Build critical analysis based on the understanding of the papers.

The complete literature review of this study is detailed in Chapter 2.

4.3.3 Gather Requirements

Since the topic of the research stems from a real-world problem and the solution proposed is in the formation of an application, we gathered the application requirements from stakeholders within a real-world organization. These requirements are gathered and specified in IEEE 29148 :2011(E) Systems and software engineering — Life cycle processes — Requirements Engineering format.

The gathered requirements were analyzed and refined to produce use-cases, functional and no-functional requirements. The details of the gathered requirements are enclosed in Chapter 3 of this study.

4.3.4 Research Design

The design of research and how it is conducted is the main crux of the whole thesis. It explains in detail the step by step processes done to achieve the end result and the logic behind each of the process. The research performed in this study can be broken down into below sub-sections, all of which shall be explained in detail later in this chapter.

- It starts with clearly defining the inputs taken from the user and the attributes of the inputs and their relation to problem solving.
- The next sub-section explains the working of CVSS and how CVSS allows us to incorporate existing controls and the asset requirements for confidentiality, integrity and availability and to determine the actual exposure and value of the asset.
- This is followed by determination of value of asset to the organization using the monetary value provided by user and the re-calculated CVSS scores using weighted sum model.
- The last sub-section provides the Boolean logic equations and application of formal methods to solve the problem within the supplied constraints using Z3 as SMT Solver.

4.3.5 Interpretation and Analysis of Results

Every research shall produce some quantifiable results which can be put to test to determine the viability and efficacy of the research. The research should explain the detailed interpretation of the received results and their significance leading to the quantifiable validation of the results.

For our study, this section discussed in Chapter 5 shall explain the prioritized list of assets our proposed solution has produced and demonstrate how is it the most optimal solution under the set of provided constraints and why there is no other solution that exists which can have more optimal results than the proposed one.

4.3.6 Validation of Results

All produced results shall be able to be tested against some real-world criteria such that their efficacy is proved in a real-world environment. We shall validate our results such that our study shall be able to prove that our proposed solution is able to patch maximum number of assets and the total asset value is highest for all the assets patched under the given set of constraints. We shall also demonstrate, that no other solution exists such that either the number of assets patched or total values of patched assets is higher than our proposed solution.

4.4 Gathering user required input

This is a very important phased since the evaluation and validation of all results and findings depend upon enough and correct gathering of input. This sub-section will discuss in details all inputs gathered for finding the solution and their significance in production of the results.

All input data in our application is gathered in the form of tables in backend MySQL database. Therefore we will discuss this section in the light of below database diagram which shall provide a clear representation of data stored in the application and is created using an online tool[34]. The actual table and column names might differ from the below diagram.

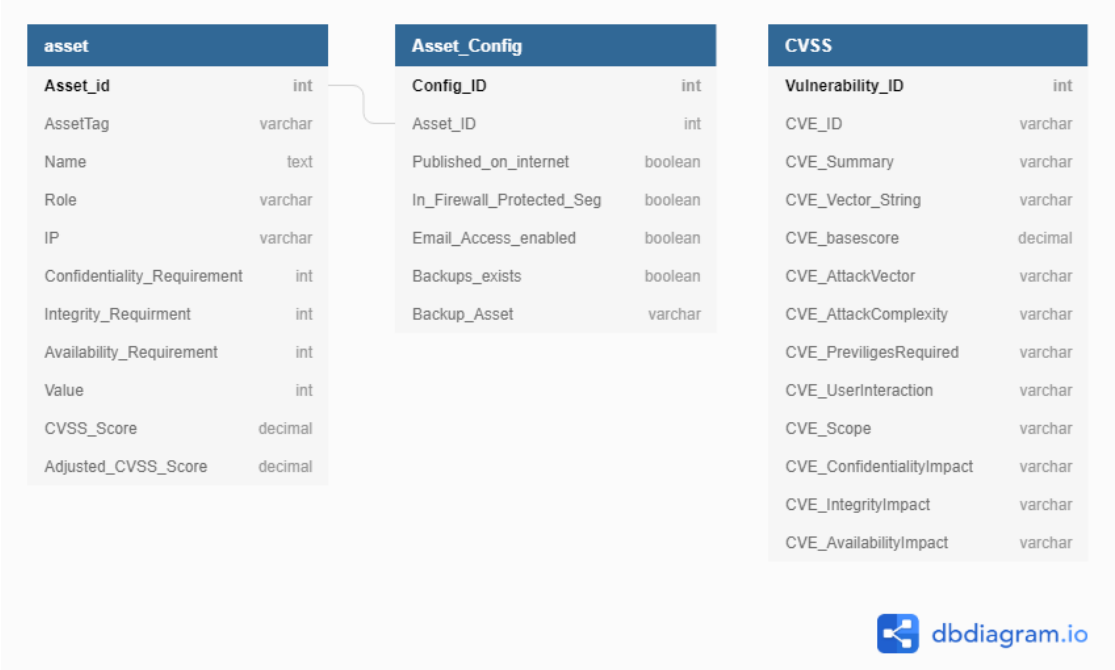


Figure 4-1 Database Diagram

Let's discuss each database table one by one.

4.4.1 Asset Table

This table stores the properties of each vulnerable asset in the organization. Entries to this table can be made either from 'New Asset Form' or bulk importing list of assets from CSV file. Below is the detailed description of each column.

a) **Asset_id** This is the primary key of the table to distinctly identify each asset within the table. It is incremented automatically when a new entry is created.
b) **AssetTag:** This field is generated automatically when the primary key is created to allocate a distinct asset tag which is further used to identify asset during problem solving within Z3 SMT Solver.
c) **Name:** This is the Fully Qualified Domain Name (FQDN) of each asset in the organization.
d) **Role:** This field identifies the primary or secondary role of an asset in a high availability, redundancy environment.
e) **IP:** This is the IP address of each asset.

f) **Confidentiality_Requirement:** This field holds the confidentiality value of the asset depending upon the data processed, stored or transmitted by the asset. An integer value (3, 2 &1) is assigned to the asset based on user choice between 'High' 'Moderate' & 'Low' respectively.

g) **Integrity_Requirement:** This field specifies the importance of integrity of the asset or the data processed, stored or transmitted by the asset. An integer value (3, 2 &1) is assigned to the asset based on user choice between 'High' 'Moderate' & 'Low' respectively.

h) **Availability_Requirement:** This field specifies the criticality of availability of the asset or the data processed, stored or transmitted by the asset to the business operations of the organization. An integer value (3, 2 &1) is assigned to the asset based on user choice between 'High' 'Moderate' & 'Low' respectively.

i) **Value:** This field holds the dollar value in multiples of 1,000 US Dollar of the asset to the organization. This value is generally comprised of the cost of the asset, value of data residing on the asset and costs of legal, statutory and compliance violations in case of breaches as well as the goodwill.

j) **CVSS_Score:** This value is automatically assigned to each asset after user specifies which vulnerability is affecting these assets and is equal to the CVSS Base Score of the vulnerability.

k) **Adjusted_CVSS_Score:** This value is input after calculation of CVSS score after factoring in the organizational and contextual values. This process is discussed in following sections in the same chapter.

The below table identifies which values are input by the user and which are processed by the application.

INPUT VALUES	**STORED BY APPLICATION**
Name	Asset_id
Role	AssetTag
IP	CVSS_Score

Confidentiality_Requirement Integirty_Requirement Availability_Requirement Value	Adjusted_CVSS_Score

Table 4-1 Asset Table Values by Input Method

4.4.2 Asset_Config Table

This table stores the user provided configurations of the asset which are required in either determining the contextual factors in assigning CVSS scores or high availability and redundancy requirements. Below is the description of each field in the table.

a) **Config_ID:** This is the primary key of the table to distinctly identify each asset configuration within the table. It is incremented automatically when a new entry is created.
b) **Asset_ID:** This is the foreign key from the Asset table pointing to the primary key of foreign table. This field identifies the configuration of each asset in the table.
c) **Published_on_internet:** This value determines whether or not the asset is publicly exposed on internet.
d) **In_Firewall_Protected_Seg:** This value determines if asset resides in a network segment entry to which is controlled via a firewall from rest of the organization infrastructure such as DMZ or isolated segment.
e) **Email_Access_Enabled:** It determines if the asset is used to access emails.
f) **Backup_Exists:** This determines if emails the asset has a redundancy requirement.
g) **Backup_Asset:** This field contains AssetTag from Asset Table specifying the primary instance of this asset.

INPUT VALUES	STORED BY APPLICATION
Published_on_internet In_Firewall_Protected_Seg Email_Access_Enabled Backup_exists Backup_Asset	Config_ID Asset_ID

Table 4-2 Asset_Config Table Values by Input Method

4.4.3 CVSS Table

This table contains all the vulnerabilities imported from NIST's National Vulnerability Database[35]. These vulnerabilities are imported in the form of JSON feeds from the NVD's website and parsed and entered in the database[36]. All details on the CVSS data field and properties are available on "Forum of Incident Response and Security Teams" (FIRST) website[37].

All values in this table are populated from JSON feed and no value is input by user.

4.5 Evaluating Vulnerability Scores using CVSS

As introduced in Chapter 1, the "Common Vulnerability Scoring System (CVSS)" is an open framework by FIRST for standardizing the characteristics and severity of software vulnerabilities. It comprises of three metric groups; "Base", "Temporal" and Environmental. While the detailed specification of CVSS has already been published for public use, we shall define few of these in order to establish relevancy with our work[38].

4.5.1 Base Metrics Group

This group contains properties of a vulnerability which remain same over temporal variations and within different user environments. This metric group is further broken down into "Exploitability Metrics" and "Impact Metrics". The former denotes the ease of exploiting the vulnerable component while the latter describes the impact as a result

of successful exploit. Below figure represents the composition of this metric group and will brief describe each of the groups below.

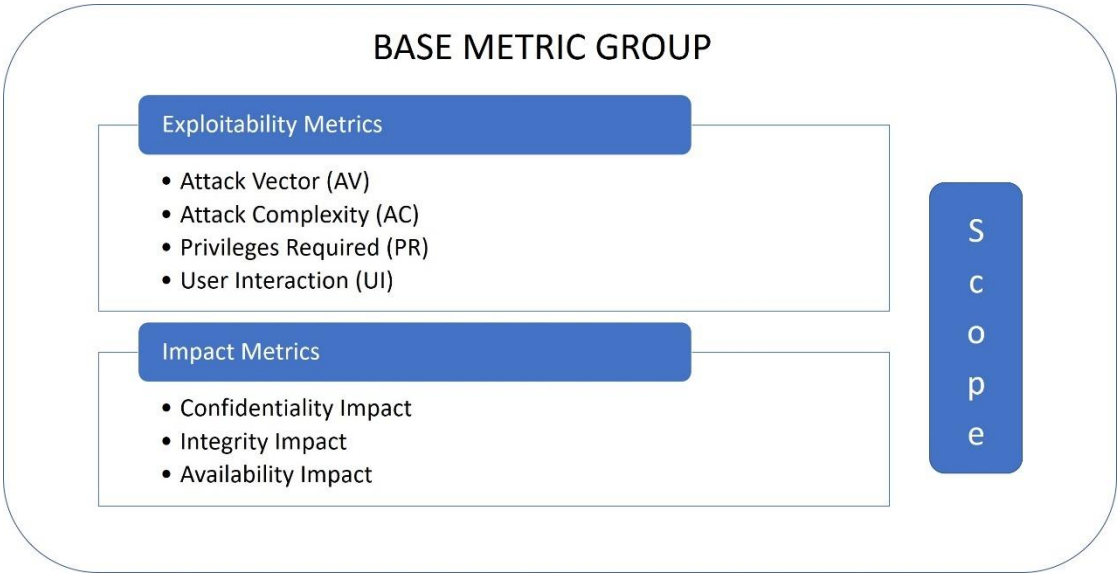


Figure 4-2 Base Metric Group

a) Attack Vector: This metric describes the contextual variable through which exploitation of vulnerability can be done. The value and the score of Base metric shall be higher if the possibility of exploitation is more remote in terms of physical and logical access. This is because the number of potential attackers shall increase for a vulnerability which can be exploited over the internet as compared to the one which can only be exploited locally. The possible values of this metric and their associated scores are Network (0.85), Adjacent Network (0.62), Local (0.55), Physical (0.2). The details of these values are specified in [37] and not detailed here to avoid redundant efforts.

b) Attack Complexity: This metric details the existing pre-conditions on the vulnerable asset whose existence is mandatory for the exploitation and are not controlled by the attacker. These pre-conditions can include finding certain information about the target, existing configurations or necessary computational settings but do not require the "User Interaction" since that is a separate metric. The possible values of this metric and its associated values are Low (0.77) and

High (0.44). The details of these values are specified in [37] and not detailed here to avoid redundant efforts.

c) **Privileges Required:** This metric describes if acquiring privileges for successful exploitation of the vulnerability shall be required by the attacker and to what extent. The possible values of this metric and its associated values are None (0.85), Low (0.62, 0.68 if Scope is Changed), High (0.27, 0.5 if Scope is Changed). Please see below for Scope metric. The details of these values are specified in [37] and not detailed here to avoid redundant efforts.

d) **User Interaction:** This metric specifies if any participation by a user other than the attacker is required for exploitation of the vulnerability. It also specifies attacker's ability to exploit vulnerability all by himself or requires a victim to participate. The possible values of this metric and their associated scores are None (0.85), Required (0.62). The details of these values are specified in [37] and not detailed here to avoid redundant efforts.

e) **Scope:** Also known as Authorization Scope, this metric says that if the vulnerability one software component under a specific authorization scope affects another component under different authorization scope, then the scope has been changed. Here scope refers to the set of privileges defined by a "computation authority" such as operating system, software or a virtual environment. The possible values of this metric are Unchanged and Changed. They do not have their own associated scores however influence other metrics such as Privileges Required etc. The details on this influence is mentioned in [37] and not detailed here to avoid redundant efforts.

f) **C/I/A Impact:** This metric measures the impact to Confidentiality, Integrity and Availability (C,I,A) on the asset and data where a successful exploitation of vulnerability has occurred. Confidentiality means restriction of access to data and preventing unauthorized access and disclosure. Integrity means the validity and correctness of the information while Availability refers to the continuous access to authorized intended users at any given moment of time. The possible values and their associated scores for this metric are High (0.56), Low (0.22)

and None (0). The details of these values are specified in [37] and not detailed here to avoid redundant efforts.

4.5.2 Temporal Metrics Group

As the name suggests this metric group defines the values which might change over time and how their variations might affect the CVSS scores. While this metric group is not mandatory for calculation of base scores, they can be defined by an analyst to check the influence on the scores. Below are the detailed metrics of this group.

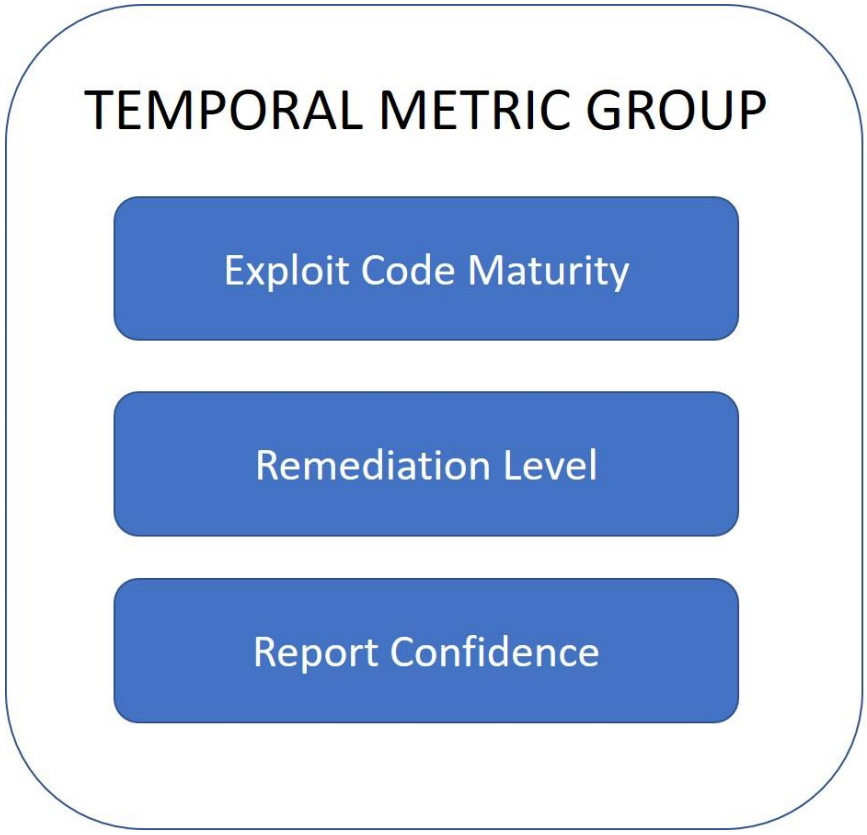


Figure 4-3 Temporal Metric Group

a) **Exploit Code Maturity:** This metric describes the likelihood of occurrence based on the current techniques of exploitation and if they are active, or "in-the-wild". The publish and disclosure of exploit-code, proof-of-concept, and a working exploit code publicly results in higher values. The possible values of this metric and its associated scores are Not Defined (1), High (1), Functional

(0.97), Proof of Concept(0.94), Unproven (0.91). Please note that Not Defined has no influence over the calculations. The details of these values are specified in [37] and not detailed here to avoid redundant efforts.

b) **Remediation Level:** This metric holds significant importance for prioritization since it describes if a remediation, workaround or official patch has been issued for a vulnerability. The existence of an official fix decreases the temporal score. However for this study, this is one of our assumptions, that the official patch has been released. The possible values of this metric and associated score are Not Defined (1), Unavailable (1), Workaround (0.97), Temporary Fix (0.96), Official Fix (0.95). The details of these values are specified in [37] and not detailed here to avoid redundant efforts.

c) **Report Confidence:** This metric defines the authenticity and credibility of the vulnerability that is reported and its published technical details. Initially the vulnerability may be reported but the associated impacts or technical details of vulnerable component, its nature or information required to exploit might not be present. These might be produced after research. If the vulnerability is validated by the vendor or higher number of reputable sources, this metric shall have a higher score. The possible values of this metric and associated score are Not Defined (1), Confirmed (1), Reasonable (0.96), Unknown (0.92). The details of these values are specified in [37] and not detailed here to avoid redundant efforts.

4.5.3 Environmental Metrics Group

This group of metrics provide the relative CVSS score based upon the importance of vulnerable asset to the organization in terms of Confidentiality, Integrity and Availability. This metric is calculated by modifying the base metrics and take values depending upon the placement of asset in the infrastructure. Below are the metrics of this group.

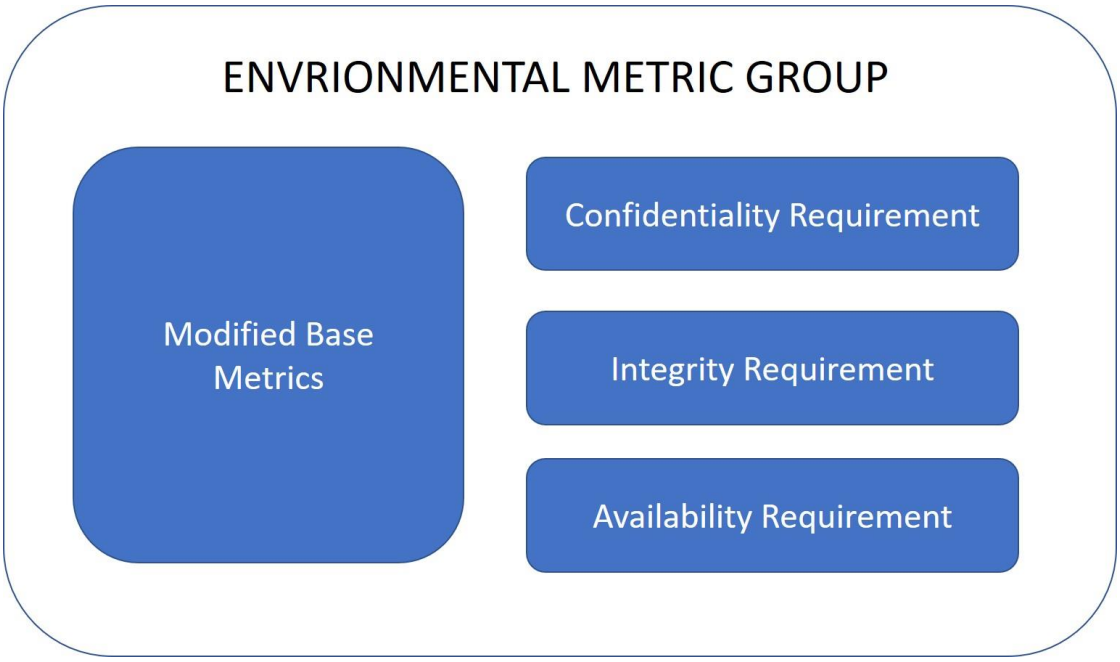


Figure 4-4 Environmental Metric Group

a) **Security Requirements (CR, IR, AR):** This metric gives the CIA value of the asset to the organization. The modified impact metrics discussed above are reweighted as per these metrics. The possible values of these metrics and their associated values are Not Defined (1), High (1.5), Medium (1), and Low (0.5). The details of these values are specified in [37] and not detailed here to avoid redundant efforts.
b) **Modified Base Metrics:** The metrics provision the ability to adjust the base metrics depending upon the current configurations in the environment and the placement of the asset such that the exploitability, impact or scope shall be altered by the placement and the configurations. It has no values of its own rather the modified values of the base metric.

The calculation of these scores and their equations are detailed in [37] and can be referred from there.

4.5.4 Enriching CVSS Scores for Asset Configurations

In this section, we define how we have calculated CVSS score for assets after completing the CVSS vectors from NVD and using asset security configurations. The CVSS scores in our solution are calculated using a Java API available at GitHub[39]. We have imported the asset security configurations from the user supplied values in the repository and the CVSS scores from NVD.

For example, the vulnerability with identified "CVE'2019-0708" from NVD has the base metrics set to below [40].


```
"cvssV3" : {
  "version" : "3.0",
  "vectorString" : "CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H",
  "attackVector" : "NETWORK",
  "attackComplexity" : "LOW",
  "privilegesRequired" : "NONE",
  "userInteraction" : "NONE",
  "scope" : "UNCHANGED",
  "confidentialityImpact" : "HIGH",
  "integrityImpact" : "HIGH",
  "availabilityImpact" : "HIGH",
  "baseScore" : 9.8,
  "baseSeverity" : "CRITICAL"
},
"exploitabilityScore" : 3.9,
"impactScore" : 5.9
```


Figure 4-5 Base Metrics for CVE-2019-0708

This is extracted from JSON feed downloaded from NVD. As seen from above, below are the base metrics set for the vulnerability.

- Attack Vector : Network
- Attack Complexity: Low
- Privileges Required: None
- User Interaction: None
- Scope: Unchanged
- Confidentiality Impact : High
- Integrity Impact : High
- Availability Impact: High
- Base Score: 9.8

We take this vulnerability as an example for prioritizing our assets. We shall discuss below how security configurations of our assets vary the score. Lets take two scenarios.

In scenario 1, the asset has highest security configurations and the second scenario has an asset with minimum configurations. This shall help us in analyzing maximum and minimum possible values for both assets.

a) **Scenario 1 (A1) :** Given below are the impact criticality metrics and the security configuration of this asset.

- Communication Internet : No
- Protected by Firewall : Yes

b) **Scenario 2 (A2):** Given below are the impact criticality metrics and the security configuration of this asset.

- Exposed on Internet : Yes
- Protected by Firewall : No

Since the asset is protected by a firewall, cannot communicate with internet and resides within the local network, the exploitability of the vulnerability via internet is not possible and therefore the base metric Attack Vector (AV) is needed to be changed from Network to Adjacent as per the CVSS V3 Specification Guide [37]. This condition is represented by below pseudocode.

```
FOR EACH Vulnerable_Asset
    if CVSS_AttackVector is NETWORK OR ADJACENT
        if INTERNET_COMMUNICATION IS NOT ENABLED OR IS_PROTECTED_BYFIREWALL
            SET CVSS_AttackVector to NETWORK
        else
            SET CVSS_AttackVector to ADJACENT
```

Figure 4-6 Pseudocode for changing Attack Vector

This is valid for both scenarios. Attack Vector metric is adjusted based on network configurations of the asset. However this cannot be adjusted for a vulnerability for which Attack Vector is set to Logical or Physical since as per CVSS V3 Specification Guide [37] these vulnerabilities cannot be exploited over the internet.

After this, the adjusted score is calculated and assigned to each asset. Below are the modified values for each Scenario.

- Scenario 1 : Modified CVSS Score comes out to be 8.3 from initial value of 9.8 as suggested by NVD.

- Scenario 2 : Since initial CVSS vector and score is set to maximum, there is no change from initial value of 9.8

4.6 Total Asset Value and CVSS Scores

The end goal to produce a prioritized list and its satisfiability can only be tested if there is a quantifiable score against which the assets should be prioritized. For this, we have two metrics; one is asset value and two are the CIA requirements scores against each asset that we calculated in section 4.5.4.

a) **Asset Value:** This is the value input by the user which is expressed in a financial value is multiples of 1,000 US Dollars. This value is determined by organization as a result of its whole Risk Assessment and Business Impact Analysis exercise[41]. This generally comprises of cost of the asset, value of data residing on the server, potential loss of business and good-will of organization in monetary terms and the legal, statutory, obligatory and compliance consequences and fines that follow with the violation.
b) **CIA Requirements:** These requirements are defined by the user to define the sensitivity and criticality of data in terms of confidentiality, integrity and availability.

4.6.1 Calculation of Total Asset Value

As per ISACA Journal (Volume 3,2017) [42], "the worth of asset of the organization's information system is based on CIA security."

$$Total\ Asset\ Value = Asset\ Value\ x\ Weight\ of\ Asset$$

Where weight of the asset is defined as

$$Weight = Confidentiality + Integrity + Availability$$

All values of CIA are based on below values

- Low (1)
- Medium (2)
- High (3)

And the weights are displayed in below matrix.

Confidentiality		Low (1)			Medium (2)			High (3)		
	Integrity	**L**	**M**	**H**	**L**	**M**	**H**	**L**	**M**	**H**
Availability	**Low (1)**	3	4	5	4	5	6	5	6	7
	Medium (2)	4	5	6	5	6	7	6	7	8
	High (3)	5	6	7	6	7	8	7	8	9

Table 4-3 Weight of Assets, taken from [42]

As per the above matrix, an asset having value of 5,000 USD and CIA requirements of High, Medium and Low will have its total asset value calculated to be 30,000 USD.

4.6.2 Weighted Sum Model

Before we go ahead and combine our CVSS values and Total Asset Values that we calculated in section 4.5.4 and 4.6.1 respectively, lets look at weighted sum model and explain its significance in this research. In the field of decision theory[43], the most simple and well-know "Multi Criteria Decision Analysis (MCDA)" method for deciding upon choice within a number of alternatives is the Weighted Sum Model or WSM[44][45]. However, the values that are being added should be in similar units.

The weighted sum model is best suited for maximization decision; i.e. the decision criteria is a benefit criteria such that higher the values are the better it is. The same model has been incorporate for cyber security risk assessments by Kure, Islam and Razzaque in [46] and ISACA also refers to applying weights to risk scores for prioritization[42]. Generically, for m alternatives and n decision criteria, the decision criterion C is defined by the below equation

$$A_i^{WSM-score} = \sum_{j=1}^{n} w_j a_{ij}\,, for\ i = 1,2,3,\ldots\ldots,m$$

Equation 1 – Weighted Sum Model

Where w denoted the weighted importance, a is the valuation metric and A is the alternative evaluated in terms of decision criterion C.

4.6.3 Combining Total Asset Value and CVSS using Weighted Sum Model

In order to obtain the prioritization list against a testable metric, we define our metric using the weighted sum model. In our case, let 'a' denote the total asset value of each asset and 'c' denotes the CVSS score of the asset, then for 'm' number of assets, the alternative A evaluated against Criteria C which denotes maximum values under patching constraints 'n' is given by equation

$$A_i^{WSM-score} = \sum_{j=1}^{n} a_j c_{ij} \, , for \; i = 1,2,3, \ldots\ldots, m$$

Equation 2 – Weighted Sum for CVSS and Total Asset Value

The viable solution for our problem shall be valid for any A such that we can prove there does not exist any other A where the above equation produces a greater value under the given constraints n.

4.7 Solving prioritization problem using Z3 SMT Solver

As of this point , we have gathered the required information from the user and the vulnerability against which to prioritize, calculated modified CVSS scores for each asset as per their individual placement within the network, determined total asset value of each asset using the formulas and weights as published in ISACA journal, and combined total asset value with CVSS score using Weighted Sum Model.

The weighted sum model has given us the final metric against which we shall now have to obtain the prioritized list of assets. Before we start prioritizing, we shall have to establish constraints for the patching.

4.7.1 Gathering patching constraints

When performing an emergency patch management activity, most of the organizations must face some challenges and constraints. Below are the brief descriptions of these constraints.

a) **Time to Patch:** The most basic constraint here is time to patch one server. This shall include deployment of patch, doing any reboots to ensure efficiency of the patch and successful testing of the patch. This will be crucial in determining that the total patch time does no exceed the allowed maintenance downtime.

b) **Maintenance Downtime:** Since patching requires multiple reboots of servers and might lead to circumstances where one or more business critical applications and services are affected or fail to restart after patch deployment. This might need rollback of patches and again shall lead to outage of services and applications. That is why emergency patching activities often require a planned downtime window in which all stakeholders including but not limited to senior management, employees, clients, customers and vendors etc. are notified prior to the activity. This is done to inform them about potential unavailability and outage of affected applications and services during the maintenance window. Not only notification, but prior approvals also need to be sought from the relevant stakeholders. Due to operational loss to business, maintenance downtime window is usually very lesser in duration as compared to what is initially requested. Therefore, it is critical that patching activity does not exceed the planned downtime allowed limit.

c) **Simultaneous Patch Deployment:** The NIST standard for patch management[10] recommends deploying of patches using enterprise patch management solutions which push patches across entire infrastructure simultaneously. However, this is not the case with all organizations; 1) due to high cost of the solution both in terms of deployment and operational maintenance of the tool and 2) in some specialized environments where high availability is required. Therefore, this constraint is needed to be considered as to how many servers can be patched simultaneously. This value is determined by number of human resources available for deploying the patch.

d) **High Availability:** Organizations such as service providers which need maximum uptime for continuous business operations face another challenges of ensuring their mission critical assets remain available even during the maintenance downtime. This is done by ensuring redundancy within the infrastructure by keeping a secondary server online in the environment. The primary server provides all the services during routine operations and secondary server remains idle. However during an outage or technical faults on the primary server, the services are immediately shifted on to the secondary server to prevent outage and unplanned downtime while the relevant team investigates and fixes the issue. While deploying patches on the redundant servers, it is crucial that both primary and secondary servers are taken special care of and are not made part of the same patching cycle which might cause outage of business-critical services.

4.7.2 Using Z3 SMT Solver for finding solution

This section shall cover complete details on how we leveraged the use of SMT solvers. For this purpose we use Z3 which is a SMT Solver made by Microsoft Research upon SMTLIB 2.0 protocol for solving satisfiability problems of first-order logic which involves background knowledge[17]. SMTLIB is an international initiative with major contributions from authors and is endorsed by a large base within the research community worldwide. The main motivation behind SMTLIB was the goal to standardize the satisfiability problem solving along with the creation of a vast collection of benchmarks which can be used for the evaluation and comparison of systems like other libraries for satisfying propositional logic[47].

Our study involves using the Z3 implementation of SMTLIB for solving the constraint problem. In the following example we first establish constraints represented in logical equations that are solvable with Z3 and then we show the optimization available for producing optimal results.

We here leverage the Z3 Java API available online at GitHub. As per moment of writing this, the API has 126 contributors and more than 11,000 commits and is highly discussed in the research community[48]. It contains both pre-compiled binaries as well as the

compiled JAR file com.microsoft.Z3.jar. We have used the compiled JAR library for application of Z3 solver.

4.7.3 Establish constraints

As defined in section 4.7.1 we have received a few constraints from the user against which we have to provide an optimal solution. Our end goal is to find a solution that satisfies those constraints and does not produce any other solution which provides higher values of total risk than the one proposed by our solution.

4.7.3.1 Condition 1: Asset definition

The first step is that we need to define the values which an asset variable can hold. Let a be the asset belonging to list of all vulnerable assets A, this is represented by below pseudocode.

```
//Condition 1
FOR EACH VULNERABLE_ASSET a
    a >= 0 OR a <=1
```

Figure 4-7 Defining possible asset values

The above implies that an asset can either be 0 or 1 and cannot hold any other value. If asset is required to be patched under the proposed solution its value shall be equal to 1 otherwise zero. This is defined by below

```
FOR EACH VULNERABLE_ASSET a
    PATCHED(a) { a = 1; }
```

Figure 4-8 Assign values to patched asset.

4.7.3.2 Condition 2: Limiting maximum number of assets that can be patched

Now we shall ensure that the maximum number of assets shall remain less than the total number of vulnerable assets in a given solution. In other words, we are defining upper bounds within which SMT solver should remain and present the solution. Let a be the asset and A denote a set of n vulnerable assets (a_1, a_2….. a_n), the patching function P(a_i) is defined by below limit.

```
//Condition 2
ASSETS_PATCHED <= TOTAL_VULNERABLE_ASSETS
WHERE ASSETS_PATCHED = SUM OF ALL PATCHED(a);
```

Figure 4-9 Restricting number of patched assets.

4.7.3.3 Define patching time

Time to patch all assets is determined by number of assets, time required to patch each asset and number of assets that can be patched simultaneously. This is defined by

```
TOTAL_PATCHING_TIME =
(TIME_TO_PATCH_EACH_ASSET * ASSETS_PATCHED) / SIMULTANEOUS_PATCHED_ASSETS
```

Figure 4-10 Calculating Total Patching Time

Here ASSETS_PATCHED is the variable value which will be determined by the SMT solver to obtain a satisfiable model. Rest of the values will are provided by user in section 4.7.1. For example, if time to patch each asset is 5 minutes and 20 assets are to be patched while maximum of 2 assets can be patched simultaneously, then TOTAL_PATCHING_TIME = (20*5)/2 = 50 minutes.

4.7.3.4 Condition 3: Total Patching Time should not exceed Allowed Downtime

The third challenge is to ensure that number of assets that are allowed to be patched should be restricted such that their total patching time should not exceed maximum allowed downtime or maintenance window authorized by the stakeholders. This is determined by below

```
//Condition 3
TOTAL_PATCHING_ TIME <= ALLOWED_DOWNTIME
```

Figure 4-11 Restricting total patch time

4.7.3.5 Condition 4: Primary and Secondary assets must not be patched together.

This condition specifies the high availability constraint such that primary and secondary assets should not be patched in same cycle. If primary is being patched, secondary should be avoided and vice versa. This is represented by below.

```
//Condition 4
FOR EACH VULNERABLE_ASSET a
    if a_isSecondary = true
        SUM OF (a and PrimaryOf_a) <=1
```

Figure 4-12 Checking for high availability requirement.

As denoted in sub-section 4.7.3.1 each patched asset value is set to 1, so if both a and PrimaryOf_a are patched the sum shall be two which shall negate this condition hence shall not stay valid.

4.7.3.6 Defining Asset values for assets.

As represented in sub-section 4.6.3 by weighted sum model, we shall now assign these values to each asset as shown below.

```
FOR EACH VULNERABLE_ASSET a
    Value_Of(a) = Total_AssetValue_Of(a) * CVSS_ScoreOf(a);
```

Figure 4-13 Determine Value of Assets

The above denotes that the value of each asset is a weight product of its adjusted CVSS Score that we determined in sub-section 4.5.4 and the Total Asset Value that we established in sub-section 4.6.1. This is determined for all assets whether patched or not. For assets which are patched, the patched values are determined by below.

```
FOR EACH VULNERABLE_ASSET a
    PatchedValue_Of(a) = a * Value_Of(a);
```

Figure 4-14 Calculating value of patched asset

As per above, if asset is patched then a will have value of 1 and the value of asset shall be counted otherwise it will be multiplied by value of a which shall be 0 for non-patched assets. The cumulative value of all patched assets is determined by sum of patched value of each asset.

```
FOR EACH VULNERABLE_ASSET a
    TotalPatchedValue = SUM OF ALL PatchedValue_Of(a)
```

Figure 4-15 Total Patched Value which needs maximization.

4.7.4 Solving all constraints

Since we have defined all constraints now, we can put in all the values we have defined above into SMT solver and check if there is a solution that exists for a particular number of assets that can be patched satisfying the defined conditions. It is worth noting here, that the only variable value which SMT can modify here is the asset which we defined in sub-section 4.7.3.1. However, SMT is only allowed to use either 0 or 1 as we restricted in the same sub-section.

The goal here is two-fold.

- First we have to find maximum number of assets that can be patched in the given condition
- Secondly the proposed solution most select those assets with highest asset value as defined in sub-section 4.7.3.6.

When the conditions are solved, Z3 returns the first possible solution which satisfies all the conditions. This value is in fact the minimum value however we are only interested in maximum values. This is achieved and solved by Z3 optimization[49].

4.7.4.1 MAX-SAT problem and Z3 Optimization

In the field of computational complexity theory, the MAX-SAT or Maximum Satisfiability problem is defined as determination of maximum number of objectives which hold a Boolean formula true for all values of the clauses[50].

Z3 allows optimization of solution either by single objective or combining multiple objectives. Since we have multiple objectives here as stated above, we shall be leverage Z3's combined objective optimization. This allows maximization of minimization of the objectives based on below strategies[51].

a) **Lexicographic Combinations:** This is the default optimization for objectives O_1 and O_2 for the constraint F using a lexicographic combination. For constraint F, this technique finds a Model M such that the pair $<M(O_1),M(O_2)>$ is lexicographically maximized.

b) Pareto Fonts: This technique produces multiple objectives for set of Models $M_1 \ldots M_m \ldots M_n \ldots$ such that either $M_m(O_1) > M_n(O_1)$ or $M_m(O_2) > M_n(O_2)$ and $M_m(O_1) < M_n(O_1)$ or $M_m(O_2) < M_n(O_2)$ holds valid at the same time.

c) Boxes: These are specified for independent objectives such as $M_1(O_1)$ has maximal value of O_1 and $M_2(O_2)$ has the maximal value of O_2.

In our case, we try to find a model such that our Model M has the maximum values for both; 1) the number of assets patched and 2) the assets patched have maximum asset value for the given set of constraints F. In short O_1 denotes Maximize(Assets) while object O_2 denotes Maximize(Asset Value).

The Z3 Optimizer for Java is part of com.Microsoft.Z3.jar library file and is available in the form of a java class. It supports all forms of optimization as discussed above such as Lexicographic combinations, Pareto Fonts and Boxes[52].

The below Java code shows the optimization for Z3 where asset and asset value both are optimized.

```
Optimize.Handle maxasset = opt.MkMaximize(asset);
Optimize.Handle maxTAV = opt.MkMaximize(TAV);
```

Figure 4-16 Getting maximized solution using Z3 Optimization.

This concludes our design and implementation of the research. The experimentation and values produced by this solution shall be discussed in the following chapter.

5 EXPERIMENTATION AND ANALYSIS

5.1 Introduction

In this chapter, we put our developed solution "Asset Vulnerability Management" to test and determine its efficacy in producing the desired results. To do this, we first detail down the tools, experimentation environment that we shall use, the testing methodology and various scenarios that are used and the visual representation of the interpreted results. This chapter is broken down into below sections.

- Section 5.2 Specification of the testing environment
- Section 5.3 Experimentation Methodology
- Section 5.4 Analysis and Representation of acquired results

5.2 Specification of Testing Environment

All test runs of our solution "Asset Vulnerability Management" are executed on the system with below hardware and software specifications:

5.2.1 Hardware Specifications

The testing environment has an eight core Intel Core i7 processor running at 2.8 GHz and a 16 GB RAM. This is run on a physical machine and no virtual environments are used.

5.2.2 Software Specifications

The testing environment is running 64-bit version of Microsoft Windows 10 release 1903 at the time of testing. The installed Java is 64-bit version of Java 8 update 211. The environment also has the Java Z3 library (libz3java.dll) added in system path at "C:\Program Files (x86)\Common Files\Oracle\Java\javapath" for execution of the Z3 library and API.

5.3 Experimentation Methodology

We now put our solution to test. This is done by taking a sample of 25 assets and establish constraints and observe how our solution behaves. The goal is to determine if all constraints are properly handled, and there is no other valid solution that exists for the sample which produces either number of assets patched or value of the asset patched higher than the one produced by our solution. To validate accuracy, we shall not vary the vulnerability against which the testing is being performed. For this we shall use vulnerability with ID "CVE-2019-0708" from the NVD[40]. Below is our sample of 25 servers.

5.3.1 Testing Sample

The sample asset table is shown below.

Asset Tag	**Name**	**Role**	**C**	**I**	**A**	**Value**
A1	Authentication Server	PRIMARY	2	3	3	25
A2	Web Server	PRIMARY	2	1	3	80
A3	File Server	PRIMARY	3	3	2	12
A4	DNS Server	PRIMARY	2	3	3	15
A5	DNS Server 2	SECONDARY	2	3	2	10

A6	Database Server	PRIMARY	3	3	1	45
A7	Training Server	PRIMARY	1	1	1	10
A8	Database Server 2	SECONDARY	3	3	3	35
A9	Email Server	PRIMARY	2	3	3	150
A10	Moodle Server	PRIMARY	1	2	2	25
A11	ERP Server	PRIMARY	3	3	3	700
A12	DHCP Server	PRIMARY	2	3	3	105
A13	PROXY Server	PRIMARY	1	2	3	90
A14	Staging Server	PRIMARY	1	1	1	125
A15	Voice Server	PRIMARY	1	1	3	110
A16	Code Repo Server	PRIMARY	3	3	2	240
A17	PROXY Server 2	SECONDARY	1	2	2	85
A18	Email Server 2	SECONDARY	2	3	2	150
A19	Monitoring Server	PRIMARY	1	1	1	80

A20	Ticketing Server	PRIMARY	2	3	2	75
A21	Backup Server	PRIMARY	2	3	1	105
A22	Payroll Server	PRIMARY	3	3	1	330
A23	Voice Server	SECONDARY	1	1	2	100
A24	Audit Server	PRIMARY	1	1	1	85
A25	ERP Server 2	SECONDARY	3	3	2	700

Table 5-1 Asset Table (Scenario 1)

Tag	Internet Allowed	Email Allowed	Protected By Firewall	High Availability Required	Redundant Asset	Adjusted CVSS
A1	FALSE	TRUE	FALSE	FALSE		8.8
A2	TRUE	FALSE	FALSE	FALSE		8.8
A3	FALSE	FALSE	FALSE	FALSE		9.8
A4	TRUE	FALSE	TRUE	TRUE	A5	8.8
A5	TRUE	FALSE	FALSE	TRUE	A4	8.8
A6	FALSE	FALSE	FALSE	TRUE	A8	8.8
A7	TRUE	FALSE	FALSE	FALSE		8.8
A8	FALSE	FALSE	FALSE	TRUE	A6	9.8
A9	TRUE	TRUE	FALSE	TRUE	A18	9.8

A10	FALSE	FALSE	TRUE	FALSE		8.8
A11	FALSE	FALSE	TRUE	TRUE	A25	9.8
A12	FALSE	FALSE	TRUE	FALSE		8.8
A13	TRUE	FALSE	FALSE	TRUE	A17	9.8
A14	FALSE	FALSE	TRUE	FALSE		8.8
A15	TRUE	FALSE	FALSE	TRUE	A23	8.8
A16	FALSE	FALSE	TRUE	FALSE		8.8
A17	TRUE	FALSE	FALSE	TRUE	A13	9.8
A18	TRUE	TRUE	FALSE	TRUE	A9	8.8
A19	TRUE	FALSE	TRUE	FALSE		9.8
A20	FALSE	FALSE	TRUE	FALSE		8.8
A21	FALSE	FALSE	TRUE	FALSE		9.8
A22	FALSE	FALSE	TRUE	FALSE		8.8
A23	TRUE	FALSE	FALSE	TRUE	A15	8.8
A24	FALSE	FALSE	TRUE	FALSE		9.8
A25	FALSE	FALSE	TRUE	TRUE	A11	8.8

Table 5-2 Asset Configuration Table

5.3.2 Assigning CVSS Scores

CVSS score for vulnerability is obtained form NVD which is 9.8. For those servers which either are not allowed to communicate on internet or are protected by a firewall have their Attack Vector and CVSS scores reduced to 8.8 as shown in section 4.5.4.

5.3.3 Assigning Constraints

Now we establish below constraints for patching. We are expected to produce a solution which lets us know how many assets can we patch under the below set of constraints?

- Time to patch each server = 15 minutes.
- No. of assets patched simultaneously = 3
- Maximum allowed patching time = 80 minutes

5.3.4 Obtain the solution

We now evaluate the results produced by our solution.

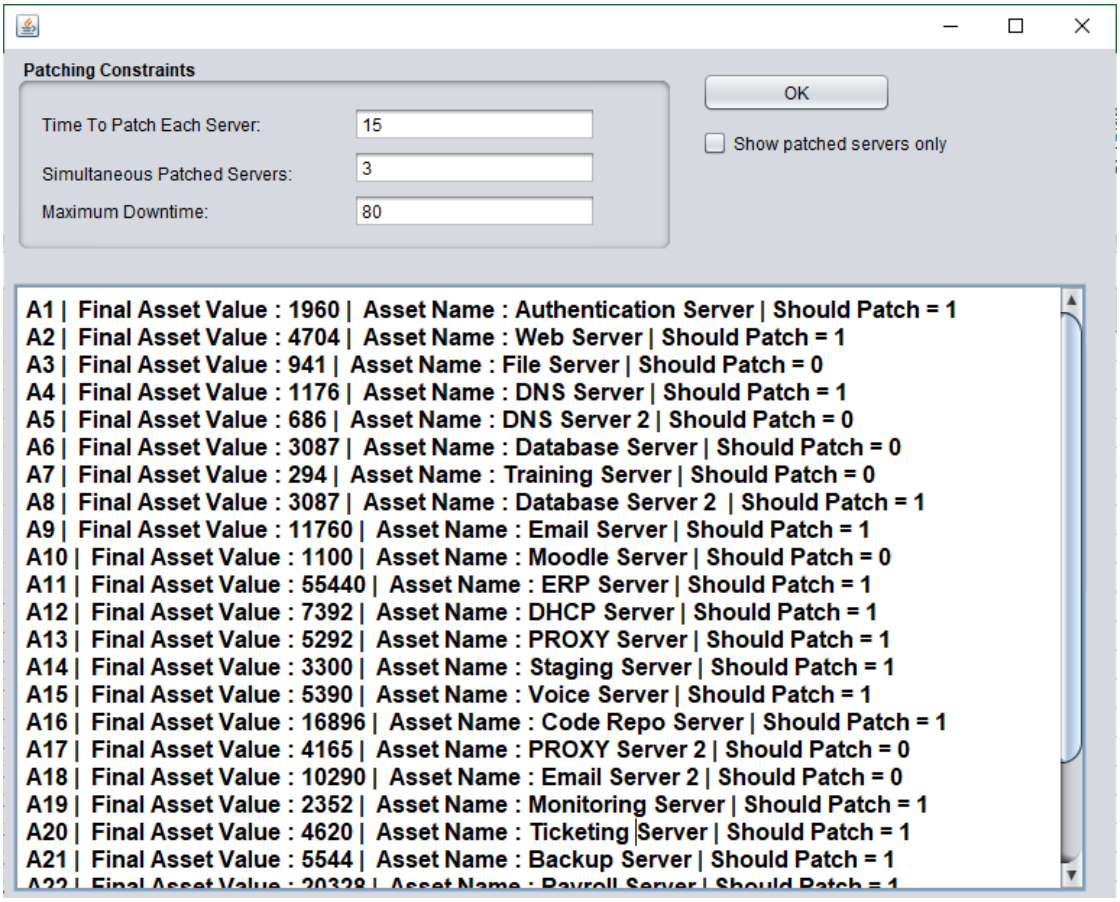


Figure 5-1 Produced Output

As per the above solution, the assets against which value of "Should Patch" is set to 1 are proposed to be patched while the other ones set to 0 should not be patched. Let us calculate the values as we describe per the equations. We take A1 for example :

Let AV denote Asset Value, W denote Weight, TAV denote Total Asset Value, C denote Confidentiality, I denote Integrity, A denote Availability, FAV denote Final Asset Value then

```
AV = 25 for A1 from Table 5-1.
W(C+I+A) = Moderate (2) + High (3) + High(3) = 8.
TAV(200) = AV(25) * W(8) (From Section 4.6.1)
FAV(1760) = TAV(200) * Adjusted_CVSS (8.8) (Section 4.6.2 Weighted Sum Model)
```

Figure 5-2 Calculating Final Asset Value

Similarly all other Final Asset Values can be calculated for rest of the assets. As we have observed the calculation of values, let us now observe how the constraints have been met one by one.

5.4 Analysis and Representation of results

After analyzing the constraints, the sample set resulted in below proposed solution where "Should Patch" implies if the asset should be patched against the vulnerability in the mentioned cycle. If it is set to 1, the asset should be patched, otherwise no. Lets now analyze if our solution meets the set of constraints.

Asset Tag	Final Asset Value	Asset Name	Should Patch
A1	1960	Authentication Server	1
A2	4704	Web Server	1
A3	941	File Server	0
A4	1176	DNS Server	1
A5	686	DNS Server 2	0
A6	3087	Database Server	0
A7	294	Training Server	0
A8	3087	Database Server 2	1

A9	11760	Email Server	1
A10	1100	Moodle Server	0
A11	55440	ERP Server	1
A12	7392	DHCP Server	1
A13	5292	PROXY Server	1
A14	3300	Staging Server	1
A15	5390	Voice Server	1
A16	16896	Code Repo Server	1
A17	4165	PROXY Server 2	0
A18	10290	Email Server 2	0
A19	2352	Monitoring Server	1
A20	4620	Ticketing Server	1
A21	5544	Backup Server	1
A22	20328	Payroll Server	1
A23	3920	Voice Server	0
A24	2244	Audit Server	1
A25	49280	ERP Server 2	0

Table 5-3 Final Output Table

5.4.1 Number of patched assets

Our solution proposed that under the given set of constraints we can patch 16 out of 25 servers which. This is given by the formula in Figure 4-10 and 4-11. Putting values into the formula, we get

$$Total\ Patching\ Time = \frac{(16\ x\ 15)}{3} = \frac{240}{3} = 80\ minutes$$

Equation 3 – Patched Assets

Since the maximum allowed downtime is 80 minutes the proposed solution has to abide by the constraint that total patching time should not exceed maximum allowed downtime as shown in Figure 4-11. Therefore this constraint has been validated for the proposed solution since exceeding more than 16 assets shall exceed total patching time from the allowed limit and thus 16 is the maximum value of number of assets that can be patched. This is visualized by below graph.

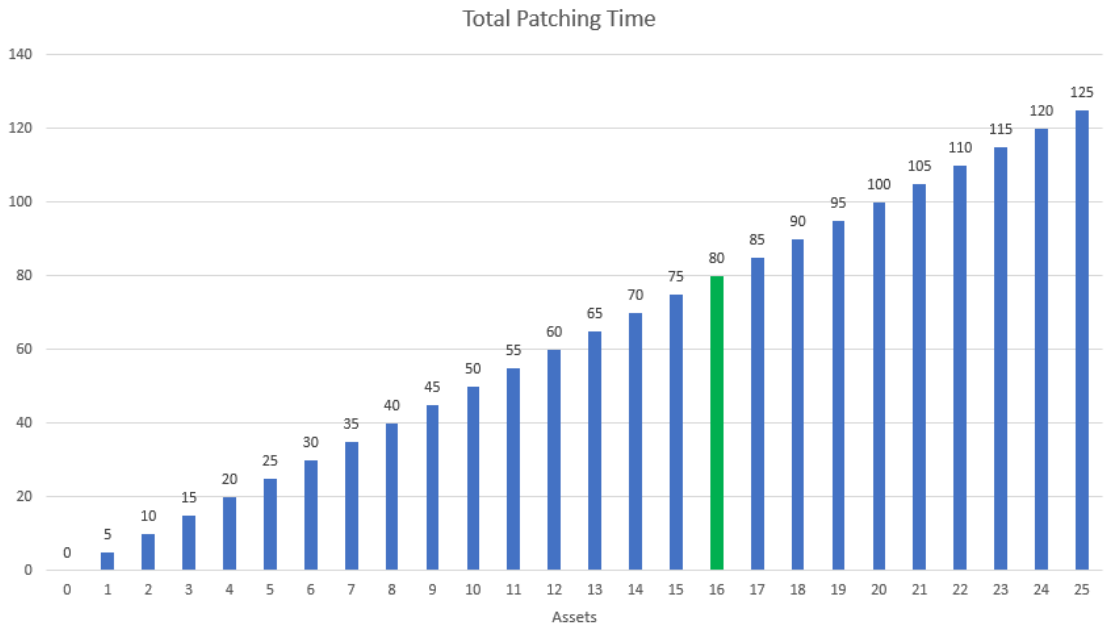


Figure 5-3 Graph showing Total Patching Time

5.4.2 Primary and Secondary asset must not be patched together

Our sample set has primary and secondary assets denoted by name “Asset Name” and “Asset Name 2”. As we see from our resultant set, there is no such “Asset Name” and its secondary “Asset Name 2” both patched in the same cycle. This validates are

constraint that primary and secondary assets were not patched together. This is represented by the below graph.

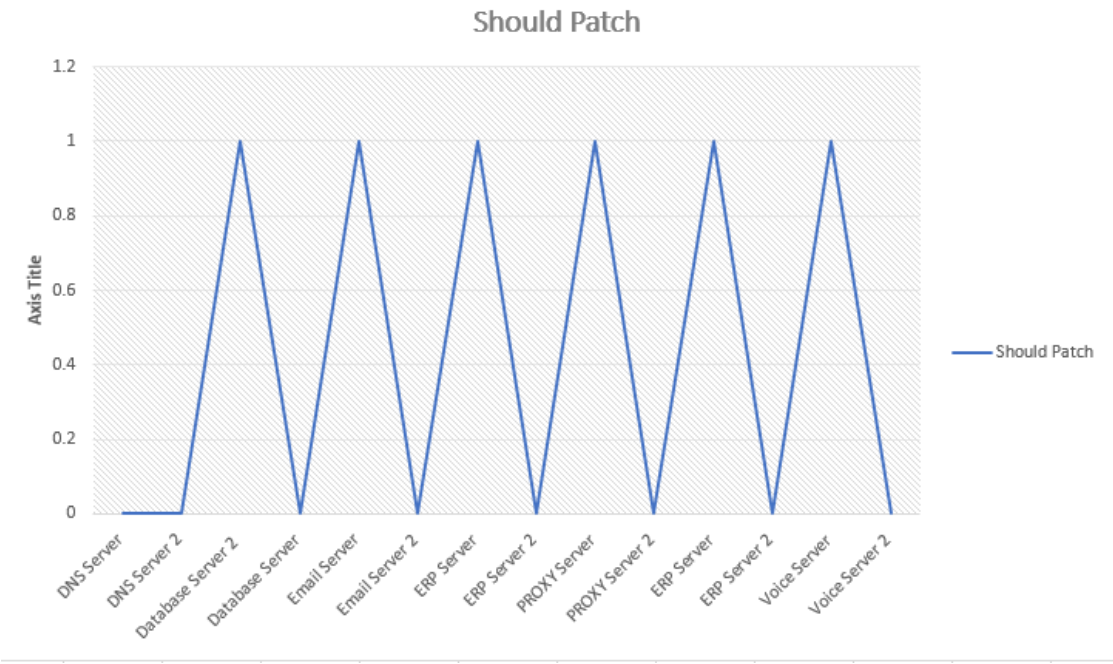


Figure 5-4 Graph showing patching of primary and secondary servers

5.4.3 Assets with maximum Final Asset Value are patched

While achieving maximum number of assets, the solution must also prioritize assets such that the patched assets must have maximum final asset value. If we sort our final results in descending order (highest final asset value on top), we get below table.

Asset Tag	Final Asset Value	Asset Name	Should Patch
A11	55440	ERP Server	1
A25	49280	ERP Server 2	0
A22	20328	Payroll Server	1
A16	16896	Code Repo Server	1
A9	11760	Email Server	1
A18	10290	Email Server 2	0
A12	7392	DHCP Server	1
A21	5544	Backup Server	1
A15	5390	Voice Server	1
A13	5292	PROXY Server	1
A2	4704	Web Server	1

A20	4620	Ticketing Server	1
A17	4165	PROXY Server 2	0
A23	3920	Voice Server	0
A14	3300	Staging Server	1
A6	3087	Database Server	0
A8	3087	Database Server 2	1
A19	2352	Monitoring Server	1
A24	2244	Audit Server	1
A1	1960	Authentication Server	1
A4	1176	DNS Server	1
A10	1100	Moodle Server	0
A3	941	File Server	0
A5	686	DNS Server 2	0
A7	294	Training Server	0

Table 5-4 Final table sorted in descending order

If we ignore the secondary redundant assets highlighted in grey in above table, we can easily visualize that there is no asset which is to be patched such that its asset value is higher than the one marked as not to be patched. Thus, this shows that the proposed solution is validating all patching constraints and conditions.

6 CONCLUSION AND FUTURE PROSPECT

The incorporation of Management Information Systems (MIS) into organization and enterprises business processes have made the management of records and their retention easier and have helped organizations in transitioning towards a paperless environment. The intelligent processing and storage of data in the form of huge cluster of databases, the growing capabilities of data mining and data warehousing, big data concepts have made the storage, processing and retrieval of billions of data records efficient, manageable and in smallest possible time. Not only the private sector, but public sector has been investing huge amounts in leveraging the MIS for record management of their citizens and all the information pertaining to them[53].

Similarly, one of the major challenges that organizations face is with the expansion and growth of their business, the size of their infrastructure grows tremendously as well. As it continuously grows, the management and assurance of its security become a more arduous task. Cloud Service Providers, online social media giants, worldwide instant messaging application providers have a huge infrastructure comprising of not only thousands but tens of thousands servers and endpoints which have to be managed for operational efficiency as well as security in such a manner that it does not interfere with the routine business operations.

As the development of operating system and applications grows, more and more features are incorporated. Every new update or version upgrade by vendor aspires to infuse new valuable features into their product in order to make a preferable selling point in the market. However, these continuous developments often overlook the key security and privacy features which are later disclosed in form a vulnerability advisory or 'exploit in-the-wild' news. Thus organizations are forced to release security updates and patches continuously to mitigate the issues.

The rich repository of private and confidential data residing in these systems have made it more appealing and lucrative for data thieves and actors with intent to gain unauthorized access to this data and sell it in the black market for dollar value. This is complemented by a huge variety and number of security vulnerabilities and bugs affecting the applications. The security researchers and bug bounty professionals are working days and nights to find these security loopholes and report securely to manufacturers before malicious actors start exploiting them.

This leads to the roll out of emergency patches and updates in the software and platforms by manufacturers after which enterprise are left with a tedious and crucial task of applying these patches in their environment. While the importance of a security patch cannot be ignored, but the risk that it might affect production and business operations always looms over the heads of enterprise managers and thus have to be handled carefully.

Considering the challenges faced by organizations for emergency patch deployments, this study addresses those challenges and proposes a viable method for deploying patches amidst those constraints and challenges. The study prioritizes the enterprises most valuable assets and ensures that they are catered to first and then the others. It allows the organizations to input their organization contextual controls and determine the risk to assets specific to their own organization instead of generic risk. This enables the system administrators and cyber security personnel to direct their efforts in the right direction. Since time is of the essence while deploying emergency patches, it is

imperative that the right assets are patched in the right order. This ensures maximum risk reduction in the given set of organizational constraints.

As for future enhancements, this study can be extended to suggest workarounds and alternatives for zero-day vulnerabilities for which official patches are not available. This technique is termed as virtual patching. For instance a vulnerability whose exploitation is possible over network stack and uses a particular layer 3 port, the possibility of blocking the communication over that port might lead to mitigation of risk completely without deploying an official patch. Off course before doing this, business requirements are needed to be determined to ensure production does not get affected.

REFERENCES

[1] NIST, "Guide for Conducting Risk Assessments SP800-30rev1," *NIST Spec. Publ. 800-30 Revis. 1*, no. September, p. 95, 2012.

[2] "Security Risk Management Considerations for Small & Medium-Sized Business." [Online]. Available: https://resources.infosecinstitute.com/security-risk-management-considerations-small-medium-sized-business/#gref. [Accessed: 04-Jul-2019].

[3] "What is Application Threat Modeling? DREAD & STRIDE Explained | NETPEAS - Cyber Security." [Online]. Available: http://www.netpeas.com/blog/what-is-application-threat-modeling. [Accessed: 04-Jul-2019].

[4] National Institute of Standards and Technology, "Risk Management Framework for Information Systems and Organizations A System Life Cycle Approach for Security and Privacy," p. 164, 2018.

[5] N. C. S. D. CSD, "NIST SP 800-53 Revision 3, Recommended Security Controls for Federal Information Systems and Organizations," *NIST SP 800-53 Revis. 3*, pp. 1–237, 2014.

[6] S. Landau, M. R. Stytz, C. E. Landwehr, and F. B. Schneider, "Overview of Cyber Security: A Crisis of Prioritization," *IEEE Secur. Priv. Mag.*, vol. 3, no. 3, pp. 9–11, May 2005.

[7] "Risk Management Concepts and the CISSP (Part 1)." [Online]. Available: https://resources.infosecinstitute.com/category/certifications-training/cissp/domains/security-and-risk-management/cissp-risk-management-concepts/#gref. [Accessed: 05-Jul-2019].

[8] "Risk Appetite, Risk Tolerance, and Residual Risk Definitions." [Online]. Available: https://www.logicmanager.com/erm-software/knowledge-center/best-practice-articles/risk-appetite-risk-tolerance-residual-risk/. [Accessed: 05-Jul-2019].

[9] F. M. Nicastro, "Security Patch Management," *Inf. Syst. Secur.*, vol. 12, no. 5, pp. 5–18, 2003.

[10] NIST SP 800-40, "Creating a Patch and Vulnerability Management Program Recommendations of the National Institute of Standards and Technology (NIST)," *Nist Spec. Publ.*, vol. 2, p. 75, 2005.

[11] "NVD - Home." [Online]. Available: https://nvd.nist.gov/. [Accessed: 05-Jul-2019].

[12] "Homepage | CISA." [Online]. Available: https://www.us-cert.gov/. [Accessed: 05-Jul-2019].

[13] P. M. P. Mell, K. S. K. Scarfone, and S. R. S. Romanosky, "Common

Vulnerability Scoring System," *IEEE Secur. Priv. Mag.*, vol. 4, no. 6, pp. 85–89, 2006.

[14] C. P. Gomes, H. Kautz, A. Sabharwal, and B. Selman, "Chapter 2 Satisfiability Solvers," Elsevier, 2008, pp. 89–134.

[15] C. Barrett and C. Tinelli, "Satisfiability modulo theories," *Handb. Model Checking*, pp. 305–343, 2018.

[16] L. de Moura and N. Bjørner, "Z3: An Efficient SMT Solver," Springer, Berlin, Heidelberg, 2008, pp. 337–340.

[17] "Z3 @ rise4fun from Microsoft." [Online]. Available: https://rise4fun.com/z3. [Accessed: 05-Jul-2019].

[18] "Slides · Z3Prover/z3 Wiki · GitHub." [Online]. Available: https://github.com/Z3Prover/z3/wiki/Slides. [Accessed: 05-Jul-2019].

[19] Flexera, "Global Trends in Vulnerabilities," *Security Review - 2018*, 1386. [Online]. Available: https://www.flexera.com/media/pdfs/research-svm-vulnerability-review-2018.pdf. [Accessed: 05-Jul-2019].

[20] Charles Cooper, "WannaCry: Lessons Learned 1 Year Later | Symantec Blogs," 2018. [Online]. Available: https://www.symantec.com/blogs/feature-stories/wannacry-lessons-learned-1-year-later. [Accessed: 04-Jul-2019].

[21] "NVD - CVE-2017-0144." [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2017-0144. [Accessed: 04-Jul-2019].

[22] K. A. Farris, A. Shah, G. Cybenko, R. Ganesan, and S. Jajodia, "VULCON: A System for Vulnerability Prioritization, Mitigation, and Management," *ACM Trans. Priv. Secur.*, vol. 21, no. 4, p. TODO, 2018.

[23] R. Ann Miura-Ko and N. Bambos, "SecureRank: A risk-based vulnerability management scheme for computing infrastructures," *IEEE Int. Conf. Commun.*, pp. 1455–1460, 2007.

[24] U. Kumar and C. Joshi, "Quantifying Security Risk by Critical Network Vulnerabilities Assessment," *Int. J. Comput. Appl.*, vol. 156, no. 13, pp. 26–33, 2016.

[25] C. Wu, T. Wen, and Y. Zhang, "A revised CVSS-based system to improve the dispersion of vulnerability risk scores," vol. 62, no. March, pp. 2018–2020, 2019.

[26] Q. Liu and Y. Zhang, "VRSS: A new system for rating and scoring vulnerabilities," *Comput. Commun.*, vol. 34, no. 3, pp. 264–273, Mar. 2011.

[27] A. A. Younis and Y. K. Malaiya, "Comparing and Evaluating CVSS Base Metrics and Microsoft Rating System," *Proc. - 2015 IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS 2015*, no. 1, pp. 252–261, 2015.

[28] I. Standard, "INTERNATIONAL STANDARD ISO / IEC / IEEE Systems and software engineering — engineering," *Iso/Iec/Ieee 29148:2011(E)*, vol. 2011, pp. 1–94, 2011.

[29] "Custom Software Requirements Specification Document." [Online]. Available:

https://belitsoft.com/blog/software-requirements-specification-document-example-international-standard. [Accessed: 26-Jul-2019].

[30] "What is Research- Definition, Types, Methods & Examples." [Online]. Available: https://www.questionpro.com/blog/what-is-research/. [Accessed: 06-Jul-2019].

[31] "The basics of writing a statement of the problem for your research proposal | Editage Insights." [Online]. Available: https://www.editage.com/insights/the-basics-of-writing-a-statement-of-the-problem-for-your-research-proposal. [Accessed: 06-Jul-2019].

[32] "Asking and Answering the 5 W's and H Questions | Thoughtful Learning K-12." [Online]. Available: https://k12.thoughtfullearning.com/minilesson/asking-and-answering-5-ws-and-h-questions. [Accessed: 06-Jul-2019].

[33] R. V. Labaree, "Research Guides: Organizing Your Social Sciences Research Paper: 5. The Literature Review."

[34] "dbdiagram.io - Database Relationship Diagrams Design Tool." [Online]. Available: https://dbdiagram.io/d. [Accessed: 07-Jul-2019].

[35] C. Frühwirth and T. Männistö, "Improving CVSS-based vulnerability prioritization and response with context information," *2009 3rd Int. Symp. Empir. Softw. Eng. Meas. ESEM 2009*, pp. 535–544, 2009.

[36] "NVD - Data Feeds." [Online]. Available: https://nvd.nist.gov/vuln/data-feeds. [Accessed: 07-Jul-2019].

[37] T. Base and T. Base, "Common Scoring System v3 . 0 : Specification Document Vulnerability," pp. 1–21.

[38] "CVSS v3.0 Specification Document." [Online]. Available: https://www.first.org/cvss/v3.0/specification-document. [Accessed: 07-Jul-2019].

[39] "GitHub - stevespringett/cvss-calculator: A Java library for calculating CVSSv2 and CVSSv3 scores and vectors." [Online]. Available: https://github.com/stevespringett/cvss-calculator. [Accessed: 08-Jul-2019].

[40] "NVD - CVE-2019-0708." [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-0708. [Accessed: 04-Jul-2019].

[41] "How to Perform a Business Impact Analysis." [Online]. Available: https://www.cleverism.com/business-impact-analysis/. [Accessed: 08-Jul-2019].

[42] C. C. Shemlse Gebremedhin Kassa, "IT Asset Valuation, Risk Assessment and Control Implementation Model." .

[43] S. O. Hansson, *Decision Theory: A Brief Introduction*. .

[44] E. Triantaphyllou, *Multi-criteria decision making methods : a comparative study*. Kluwer Academic Publishers, 2000.

[45] Wiki, "Weighted Sum Model." [Online]. Available: https://en.wikipedia.org/wiki/Weighted_sum_model. [Accessed: 08-Jul-2019].

[46] H. Kure, S. Islam, and M. Razzaque, "An Integrated Cyber Security Risk Management Approach for a Cyber-Physical System," *Appl. Sci.*, vol. 8, no. 6, p. 898, 2018.

[47] C. Barrett, A. Stump, and C. Tinelli, "The SMT-LIB Standard – Version 2 . 0," pp. 1–12.

[48] "GitHub - Z3Prover/z3: The Z3 Theorem Prover." [Online]. Available: https://github.com/Z3Prover/z3. [Accessed: 09-Jul-2019].

[49] "Z3 Optimization | rise4fun." [Online]. Available: https://rise4fun.com/Z3/tutorial/optimization. [Accessed: 09-Jul-2019].

[50] B. Borchers and J. Furman, "A Two-Phase Exact Algorithm for MAX-SAT and Weighted MAX-SAT Problems," *J. Comb. Optim.*, vol. 2, no. 4, pp. 299–306, Dec. 1998.

[51] N. Bjørner, A. D. Phan, and L. Fleckenstein, "νZ-an optimizing SMT solver," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9035, pp. 194–199, 2015.

[52] "z3/Optimize.java at master · Z3Prover/z3 · GitHub." [Online]. Available: https://github.com/Z3Prover/z3/blob/master/src/api/java/Optimize.java. [Accessed: 09-Jul-2019].

[53] W. Dorotinsky and J. Watkins, "Government Financial Management Information Systems," in *The International Handbook of Public Financial Management*, London: Palgrave Macmillan UK, 2013, pp. 797–816.