



SEEK WISDOM, ELEVATE YOUR INTELLECT AND SERVE HUMANITY !



ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL AND COMPUTATIONAL
SCIENCES

SCHOOL OF INFORMATION SCIENCE

DESIGNING SYSTEMIC VULNERABILITY ASSESSMENT
FRAMEWORK FOR PENETRATION TESTING

By

Beimnet Bekele Degefu

March 2022



SEEK WISDOM, ELEVATE YOUR INTELLECT AND SERVE HUMANITY !



ADDIS ABABA UNIVERSITY
COLLEGE OF NATURAL AND COMPUTATIONAL
SCIENCES
SCHOOL INFORMATION SCIENCE

DESIGNING SYSTEMIC VULNERABILITY ASSESSMENT
FRAMEWORK FOR PENETRATION TESTING

By

Beimnet Bekele Degefu

March 2022

A THESIS SUBMITTED TO ADDIS ABABA UNIVERSITY IN
PARTIAL FULFILLMENT FOR THE DEGREE OF MASTER OF
SCIENCE IN INFORMATION SYSTEM AND SCIENCE

Advisor: Dr. Workshet Lamnew (Ph.D.)

ADDIS ABABA UNIVERSITY
COLLEGE OF NATURAL AND COMPUTATIONAL
SCIENCES

SCHOOL OF INFORMATION SCIENCE

DESIGNING SYSTEMIC VULNERABILITY ASSESSMENT
FRAMEWORK FOR PENETRATION TESTING

Beimnet Bekele Degefu

Name and signature of Members of the Examining Board

Workshet Lamnew (Ph.D.)

Advisor

Signature

Date

Examiner

Signature

Date

Examiner

Signature

Date

ABSTRACT

The main goal of this study is to develop a “systemic vulnerability assessment framework” that can improve the performance of penetration testing. This is done by defining a mechanism to quantify the “risk” that is associated with an identified “target node” with the “operating environment” in consideration.

The approach generally taken in this study utilized the “Design Science Research” paradigm. The main problems identified are; the lack of “numerically quantifying the risk of vulnerabilities” during penetration testing and an overwhelming amount of data that is produced during “vulnerability assessment” in penetration testing. Accordingly, to improve these aspects, data gathered from open-source vulnerability databases were utilized. Following, with the aid of provided CVSS scores and “analysis and literature review”, a framework was proposed. Finally, a script was developed based on the framework and tested using scanned synthetic data.

In the demonstration using synthetic data, it was found that the abstraction model proposed, significantly reduces the number of paths that need to be accessed. That is the abstraction taken by the framework needs to analyze only 61,258 paths formed by 499 weaknesses, as compared to several billion paths that can be formed using more than two weaknesses.

Furthermore, “related weaknesses” or “pattern” criteria found in CWE data can help testers prioritize paths formed together with the “chained score” and “vulnerability connectivity operator”. Additionally, the output of CVEs based on platforms can help “penetration testers “choose their automated tools based on the list generated; unlike traditional penetration testing processes that rely on automated testing tools to identify weaknesses.

Finally, the main limitation of this study is the frameworks’ inability to capture unregistered vulnerabilities. In addition, coming up with a more pragmatic approach that can reduce “analyzing paths” that do not have to be considered can improve the framework.

Keywords: CVSS, CWE, CVE, vulnerability, Vulnerability Assessment

DECLARATION

I, Beimnet Bekele Degefu, hereby declare that the work presented in this thesis titled “systemic vulnerability assessment Framework for penetration testing”, is an original work of my own and prepared under the guidance of my supervisor Dr. Workshet Lamnew. It has not been presented for any scholastic achievement in any University. All the sources of the materials used have been acknowledged duly.

Signature:

Beimnet Bekele Degefu

This thesis has been submitted for examination with my approval as university advisor.

Advisor’s Signature:

Workshet Lamnew(Ph.D.)

DEDICATION

This work is dedicated to my mother who has sacrificed all she had for my success. May God repay you and give you peace.

ACKNOWLEDGEMENT

First, I would like to thank my God, for giving me the strength and the chance to endure this frustrating and emotionally draining time of my life.

I would also want to thank my family who has suffered with me as I was conducting my research. Special thanks to my mother who has been a solid rock during this work with all the emotional support.

Additionally, I would like to express my gratitude to my examiners Dr. Dereje and Dr. Million for their honest and constructive feedback. Special thanks to Dr. Million's good-willed encouragement and insights that have boosted my morale to finalize this work.

Finally, I would like to thank my advisor Dr. Workshet Lamenew, who has guided me through this work and all those seminars he arranged which meant a lot to me. I would also like to acknowledge and express my gratitude to NIST, who has made vulnerability data publicly available freely. In addition, MITRE & FIRST have also provided detailed documentation and data about their work which were immensely helpful for this work.

TABLE OF CONTENTS

ABSTRACT.....	I
DECLARATION	II
DEDICATION	III
ACKNOWLEDGEMENT	IV
TABLE OF CONTENTS.....	V
LIST OF TABLES	VIII
LIST OF FIGURES	IX
LIST OF ACRONYMS	X
1 INTRODUCTION	1
1.1 Research Background	1
1.2 Research Motivation	1
1.3 Statement of the Problem.....	2
1.4 Research Questions	3
1.5 Objective of the Research	3
1.5.1 General Objective	3
1.5.2 Specific Objectives	3
1.6 Scope and Limitation of the study	3
1.7 Significance of the study.....	4
1.8 Operational Definitions.....	4
2 LITERATURE REVIEW	6
2.1 Vulnerability Assessment and Penetration Testing	6
2.1.1 Penetration Testing	6
2.1.2 Vulnerability Assessment	6
2.2 Risk Assessment	7
2.3 CVSS (Common Vulnerability Scoring System)	7
2.4 CWSS (Common Weakness Scoring System).....	8
2.5 CWE (Common Weakness Enumeration)	9
2.6 Penetration testing Standards (OWASP, NIST & PTES).....	10
2.6.1 OWASP.....	10
2.6.2 PTES	11
2.6.3 NIST.....	11
2.7 Related Works.....	12
2.7.1 Attack Graph-Based Risk Assessment.....	14

3 RESEARCH METHODOLOGY.....	18
3.1 Design Science Research	18
3.2 Problem Identified	19
3.3 Objectives of the solution.	19
3.4 Data source, collection, and Analysis	19
3.4.1 Data Collection for Demonstration and Evaluation.....	20
3.4.2 Data Collection for Framework Development.....	21
3.5 Missing Values.....	21
3.5.1 Analysis.....	21
3.5.2 Tools	21
3.6 Design and development of Framework	21
3.7 Demonstration and Evaluation.....	22
3.8 Communication.....	22
4 DATA ANALYSIS.....	23
4.1 Analysis.....	23
4.1.1 Summary of Data from Databases	23
4.1.2 NVD vulnerabilities statistics & summary	23
4.1.3 CVSS Score Analysis	27
4.2 Summary	28
5 DESIGN AND DEVELOPMENT OF FRAMEWORK	29
5.1 Graph-based “Vulnerability Assessment Framework”	29
5.1.1 Vulnerability Chaining.....	30
5.1.2 Exploitability and Impact vectors of the CVSS.....	31
5.1.3 Systemic Vulnerability Assessment Framework	32
5.1.4 Proposed of Framework.....	37
5.2 Discussion	37
5.2.1 Data Gathering	37
5.2.2 Processing Data.....	38
5.2.3 Data Presentation	39
6 DEMONSTRATION AND EVALUATION	40
6.1 Demonstration.....	40
6.1.1 Data for Demonstration.....	40
6.1.2 Processing Data and Flow Chart.....	41
6.1.3 Presentation.....	43
6.2 Evaluation	43

6.3 Discussion	44
7 CONCLUSION AND RECOMMENDATION.....	46
7.1 Conclusion	46
7.2 Recommendation	47
REFERENCE.....	49
APPENDIX A.....	52
APPENDIX B	68
APPENDIX C	76

LIST OF TABLES

Table 1 CWSS metric groups summary (taken from MITRE documentation)	8
Table 2 OWASP penetration testing stages	10
Table 3 Summary of Related works.....	16
Table 4 Summary of used data.....	23
Table 5 CVSS Attack Vector values definition (taken from CVSS specification) [20].....	30
Table 6 CVSS Privileges Required values definition (taken from CVSS specification) [20].	31
Table 7 CVSS Scope Vector values definition (taken from CVSS specification) [20].....	32
Table 8 Summary of Threat source to Target "Attack Vector" Chaining logic matrix	34
Table 9 CVSS version 3 Vectors metric value as per the specification.....	36
Table 10 Summary of scanned data used for demonstration	40
Table 11 CVES used for Demonstration	42
Table 12 Summary of Edges formed based on Table 11	42

LIST OF FIGURES

Figure 1 CVSS metric groups taken from specification [20]	8
Figure 2 top 25 weakness types of 2021 (taken from CWE website).....	10
Figure 3 PTES standard [13].....	11
Figure 4 Taken from NIST security testing standard [12].....	12
Figure 5 Agent based BDI model taken from Chu [24].....	13
Figure 6 Ensemble testing approach taken from Goel [6]	14
Figure 7 base risk assessment model taken from Aksu [8].....	15
Figure 8 Attack graph risk assessment model [8].....	15
Figure 9 Attack vector path formation (taken from Aksu) [8].....	15
Figure 10 Design Science Framework taken from Hevner [29]	18
Figure 11 DSR research methodology model by Ken Peffers [30]	19
Figure 12 CWE Id category found in NVD database	24
Figure 13 Correlation matrix heat map based on NVD database CVSS version 3 Base metric vectors (2002-2021).....	24
Figure 14 Severity counts of vulnerabilities from NVD database (2002-2021).....	25
Figure 15 Severity along Attack Integrity based on NVD data (2002-2201)	25
Figure 16 Severity along Attack vector based on NVD data (2002-2201).....	26
Figure 17 Severity along Scope based on NVD data (2002-2201).....	26
Figure 18 Severity along Privileges required based on NVD data (2002-2201)	27
Figure 19 Number of Vector counts that can be assume same severity scores	27
Figure 20 Number of Vector counts that can be assume same severity scores (when chaining two base metric vectors)	28
Figure 21 Threat source model based on the CVSS Attack vector.....	29
Figure 22 Threat and Target Abstraction based on CVSS Attack vector	33
Figure 23 Aspects and Comparison considered.....	36
Figure 24 Vulnerability risk Assessment framework for Penetration testing.....	37
Figure 25 Data processing metrics Calculation visulaization.....	38
Figure 26 Flowchart summarizing the data processing in the proposed framework.	41
Figure 27 Illustration of Table 12 results.....	43

LIST OF ACRONYMS

CAPEC	Common Attack Pattern Enumeration and Classification
CVSS	Common Vulnerability Scoring System
CVE	Common Vulnerability Exposure
CIA	Confidentiality Integrity Availability
CWE	Common Weakness Enumeration
FIRST	Forum of Incident Response and Security Teams
ICT	Information Communication and Technology
NIAC	National Infrastructure Assurance Council
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OPENVAS	Open vulnerability Assessment Scanner
OS	Operating System
OSVDB	Open-source Vulnerability Database
OSI	Open Systems Interconnect
OWASP	Open Web Application Security Project
PTES	Penetration Testing Execution Standard
RMF	Risk Management Framework
SDLC	Software Development Life Cycle
VAPT	Vulnerability assessment and Penetration testing

CHAPTER ONE

INTRODUCTION

This chapter discusses the background of the need for a “systemic vulnerability assessment framework” that can be used during ‘Penetration testing’. The focus is to devise a technique that will aid to optimize the performance of penetration testing from the point of “cost” and “time”, by putting some form of metrics in the “vulnerability assessment process”.

1.1 Research Background

To begin with the behavioral aspect of Security; Moore and Anderson [1] claim that many organizations fail because administrators do not recognize the cost of failure. This simply shows that the “understanding and awareness” of defenders; concerning the “consequence of failure due to security breach” is lacking. Therefore, reasoning that integrating economic analysis using most common attacks would aid policymakers with regard to “Information Security”.

Furthermore, Moore and Anderson [1] have put the following three main points as to why “system administrators are not fully aware of the cost of failure”:

- misaligned incentives (for instance banks are not as much affected by credential leakage as the customers)
- asymmetric information (that is people in the information security industry exaggerate harm when reporting) victims under-report incidents

Following, the economic damage of information security incidents, Pirounias, Mermigas, and Patsakis [2] research, among US big corporations, using their stock market value. The findings estimated the “cost of security breach” between \$168 and \$200 million for non-tech firms. This skyrockets even more for tech giants amounting to \$356-\$381 million. Consequently, following incidents regarding security in 2010/11, 60% of large corporations have begun giving more attention to “risk of security” [2].

Hence, given such predicaments, one of the most commonly practiced tasks to ensure security in organizations is performing “Penetration testing”. For instance, OWASP (Open Web Application Security Project) suggests regularly undertaking penetration testing in a focused manner [3]. In the news later by Bruce Schneier [4], the goal of security administrators should be to reduce the “window of opportunity”, that is narrowing the gap between the “vulnerability” being discovered and being patched.

Hence, given the trend of vulnerability discovery, which has shown no sign of slowing down since 2000 [5]; it is incumbent upon penetration testers to come up with a framework and methodology that can aid perform these tasks in a “timely”, “efficient” and “cost-effective” manner.

1.2 Research Motivation

The researcher’s main motivations can help to visualize the main problem the study focuses on. In addition, this motivation should have a scholarly basis. Thus, the main driving motivations can be summarized as follows:

- Due to the growing number of vulnerabilities; there is no single penetration testing automated tool that can discover all types of “vulnerabilities” [6]. For instance, Metasploit (popular exploitation tool version 6.0.20) contains 2081 exploits. Compared to more than 166,000 vulnerabilities recorded in NVD open-source database, which is less than 2% [5].
- Using different testing tools does not guarantee the discovery of weaknesses that can still cause damage [6]. Therefore, instead of starting testing for vulnerabilities from “scanning result” of tools, starting from open-source vulnerability databases that try to collect and publish weaknesses exhaustively can improve discovery.
- There is an overwhelming amount of data produced during the “vulnerability assessment” stage of “penetration testing” [7]. Hence, using standardized systems that compare weaknesses “numerically” can improve the penetration testing process; by helping testers prioritize.
- Finally, there are graph-based risk assessment models that numerically analyze paths using standardized scoring systems like CVSS [8][9]. Incorporating such models in the vulnerability assessment process can greatly improve the performance of the penetration testing process.

1.3 Statement of the Problem

Given the proliferation of IT in modern-day systems and the rate at which vulnerabilities are being discovered, information security concerns are growing in the industry [10][5][11], it is clear that “Information Security” is going to be an issue for a long time in IT environments.

Conducting a test on every “vulnerability in a system” is going to be challenging for two main reasons. The first reason is the “number of vulnerabilities discovered every year is increasing drastically” [5]. The second one is the number of devices to be considered even in small-scale networks complicating matters. This clearly shows that using automated tools for “assessment” and “testing” is not enough.

Therefore, it is clear that organizations have to come up with a security testing process that can be performed on regular basis. One of the main issues in traditional penetration testing standards is that; they leave the task of quantifying the “risk of the weakness” found in their devices completely to testers and do not necessarily assess the risk posed on targets with respect to the network as a whole [5][12][13]. So, the guidelines like PTES and NIST for security assessment generally recommend incorporating some kind of risk assessment process in the testing process [13].

Even more, most traditional “penetration testing” activities are conducted on nodes considered a target, which does not capture the “risk of vulnerabilities” with respect to the system as a whole (especially neighboring vulnerabilities). To use the term used by PVR Murthy, this can mean that “security coverage criteria or test adequacy criteria do not have a systematic basis” [14]. Consequently, even after verifying certain weaknesses exist in the target, it generally doesn’t give testers a means of comparing “level of risk” numerically, with the system as a whole in mind.

Hence, while performing some kind of vulnerability-centric risk assessment is recommended in the “penetration testing” standards, there is no “go-to” standardized framework proposed by scholars to be used [12][13]. Additionally, the risk of vulnerabilities is evaluated without considering weaknesses in the neighboring nodes. This clearly shows the need for a “systemic

vulnerability assessment” process that can improve the “vulnerability assessment” stage of penetration testing.

Therefore, this study aims to develop a “systemic vulnerability assessment framework”. The main desired attributes of the framework are two. The first desired attribute is adding the ability to consider a ‘large number of weaknesses. The second feature is to evaluate risk with weaknesses in ‘neighboring nodes in consideration’ and ‘numerically’.

1.4 Research Questions

Thus, in this study the following research questions are answered:

1. How can we develop a “systemic Vulnerability assessment” framework for penetration testing?
2. How can we adopt the “Common Vulnerability Scoring System (CVSS)” in penetration testing during “vulnerability assessment”?

1.5 Objective of the Research

1.5.1 General Objective

The general objective of this study is to develop a “systemic vulnerability assessment framework” for penetration testing that enables testers to quantify risk numerically. This helps defenders/testers prioritize based on damage posed.

1.5.2 Specific Objectives

The specific objectives of this study are the following:

- collecting available vulnerabilities from the most common vulnerability databases, mainly NVD and CVE
- Gathering sample data using Nmap(tool) scanners that can be used for testing the framework
- Cleaning up data from open-source vulnerability databases and performing analysis to extract insight that can aid the framework development process.
- Developing a “Systemic vulnerability assessment framework” that can be used by penetration testers before moving on to the exploitation stage.
- Developing a script that can demonstrate how the framework can be implemented and be used using synthetic data.

1.6 Scope and Limitation of the study

The scope of this study is formulating a “systemic vulnerability assessment framework” that is conducted during penetration testing. The following points can further describe the main aspects of this study.

- The work attempts to evaluate risk based on published in open-source vulnerability databases by NIST and CVE details. Data published ranging from 1999 E.C. to March of 2021 E.C.
- The CVSS based “Risk assessment processes” is the main focus of this study as it commonly accepted standard that attempts to gauge the severity of vulnerabilities. Additionally, CVSS is also widely utilized by open-source comprehensive vulnerability databases.

- Finally, penetration-testing standard methodologies like PTES and NIST are” used. Therefore, improving the way “vulnerability assessment” is conducted in “penetration testing” is the goal of this study. (Black box penetration testing is the main stay of this study).

The main limitation of this research is:

- This research does not intend to consider “zero-day” vulnerabilities, as it is another area of research that cannot be incorporated due to time constraints.
- As CVSS (Common Vulnerability Scoring System) is the core of the vulnerability assessment process in the study, addressing limitations of the CVSS (like ones faced in vulnerability chaining) is going to be beyond the scope of this study.
- Additionally, handling vulnerabilities with no score is not the main goal of this study. Thus, the vulnerabilities without a score are not going to be further processed.
- Additionally, performing and verifying the existence of a path identified in the list is not within the scope of the study. Consequently, there is no exploitation involved in this study.

1.7 Significance of the study

This research aims to make the process of vulnerability assessment in penetration testing more systematic and manageable, with respect to the target network in consideration and publicly known vulnerabilities. This study proposes a framework, which can aid security professionals to identify detrimental vulnerabilities with respect to the ‘system of interest’. hence:

- Network administrators can certainly use this framework to improve the security of their data centers.
- It can help defenders optimize their performance of “penetration testing”, by identifying weaknesses that pose more threat numerically.
- Security professionals also could utilize or improve the “systemic vulnerability risk assessment framework” developed to aid script development.
- Furthermore, it could motivate others to research “penetration testing” by adding/improving on the limitation of this proposed framework.
- Additionally, there can be further studies that could help quantify the loss from the “financial aspect of security”, based on the output of this research.

1.8 Operational Definitions

The following are frequently utilized keywords used to describe concepts in the “framework development” process.

Vulnerability- is any weakness found in software/hardware that an attacker can use to launch a successful attack.

Vulnerability assessment- is the process of scanning and identifying the possible weakness found in devices running in a network.

CVSS- generic framework proposed by NIAC; that helps security professionals compare vulnerabilities with respect to common attributes.

CWE- generic vulnerability classification system proposed by mitre.org to classify vulnerabilities based on their type. By type means the technical nature of the weakness. For instance, two different vulnerabilities found in two websites hosted using different platforms can have a “cross-site scripting attack” which has a CWE id of 79.

CVE- is a unique identifier given to published vulnerabilities found in software/hardware by mitre.org.

ZERO-DAY- refers to a vulnerability that is not known. Thus, giving defenders 0 days to take mitigation measures.

CHAPTER TWO

LITERATURE REVIEW

This chapter discusses basic concepts about penetration testing, vulnerability scoring systems, penetration testing standards & related works.

2.1 Vulnerability Assessment and Penetration Testing

2.1.1 Penetration Testing

Penetration testing can be defined as the process of running applications remotely to find security weaknesses (vulnerabilities) without knowing the inner working of the target [3][15]. Vulnerabilities are weaknesses that “an application has” [16]. These weaknesses are used by attackers to inflict some kind of damage to the target system [16].

Generally speaking, “penetration testing” can be categorized as External and Internal from the mode of operation perspective. External pen testing aims to test the target network outside of the local network [15]. It usually focuses on testing devices that are visible on the internet from the target network [15]. Internal Pen testing tests for security from an environment that is inside the network of the target system [15].

Another way of categorizing penetration testing can be “blind penetration testing”, “double-blind penetration testing” and “target penetration testing” regarding the “level of awareness of IT staff”. This kind of strategy employed could vary based on the specific objective of the test [15].

Additionally, based on scope and requirement, there is “black box”, “gray box” and “white-box” penetration testing [15].

In “Black box penetration testing”, the testers are given a minimal amount of information about the target; hence reconnaissance is given more emphasis to gathering information in this type of testing [15].

In “Gray box penetration testing”, testers are given partial information about the target [[15]. This type of testing tends to draw a clear role between testers and IT staff [15].

Finally, in “White box penetration testing”, complete information about the target is going to be provided, including codes and other design patterns [15]. This type of testing is usually costly and time-consuming.

2.1.2 Vulnerability Assessment

Vulnerability assessment within the context of penetration testing is the process of scanning the network of a system with the intent to identify possible software vulnerabilities in the system [16]. In this study, the general step taken to identify possible vulnerabilities is to use scanning results (OS version, service version, device version) to match with recorded data (from an open-source vulnerability database). It should be noted, that this method of identification normally generates many false positives.

2.2 Risk Assessment

Risk as defined by Webster's dictionary is "the possibility of loss or injury". Therefore, "risk assessment" can be seen as the process of evaluating the 'possibility of losses' under a predefined context.

Coming to risk assessment concerning Information security, NIST considers "risk assessment" as part of the "risk managing process" [17]. The "risk managing process" by NIST normally consists of "framing risk", "assessing risk", "responding to risk" and "monitoring risk" [17].

Furthermore, the main reason for conducting a "risk assessment" is to identify threats/weaknesses (which may or may not be software-based) that could cause potential harm, and to determine the likelihood of identified potential harm [17].

Coming to the context of this study, it should be noted there are many proposed "risk assessment frameworks". Comparing, evaluating, and choosing a preferred risk assessment model is not the goal of this study; as the path followed in the risk assessment is dictated by two main reasons. The reasons are the type of data at hand and the "risk assessment" to be conducted is for penetration testing (which orders the focus on a 'vulnerability-centric' risk assessment process).

2.3 CVSS (Common Vulnerability Scoring System)

Common Vulnerabilities Score System (CVSS) is a threat Modeling and Scoring framework developed by NAIC [18]. NAIC is a group of security consultants that advise US Homeland security [18]. Following its conception, FIRST (Forum of Incident Response and Security Teams) special interest group (formed in 1990) was selected as a custodian for CVSS by NAIC [18]. The goal of the study was to develop a 'vulnerability scoring' system that can be standardized and be used openly; by security professionals [18].

The CVSS has three categorical metric groups which are Base, Temporal and Environmental metrics [18][19].

The 'base metric group' refers to properties of the specific vulnerability that do not change over time [18][19][20]. For instance, if a vulnerability exists in a system, the way it can be utilized by perpetrators does not change over time [18][19][20]. The 'temporal metrics group' attempts to measure the nature of the vulnerabilities that could change over time. Finally, the 'environmental metrics group' aims to contextualize the damage the vulnerability can cause to the target network infrastructure [18][19][20]. The following diagram can aid in the visualization of the scoring system.

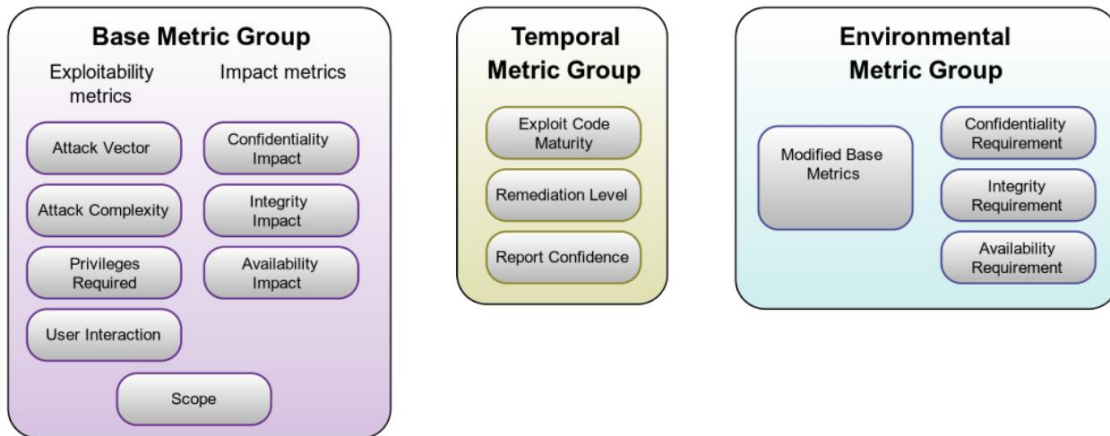


Figure 1 CVSS metric groups taken from specification [20]

In this study ‘CVSS version 3.1’ is used in the threat modeling and vulnerability assessment process for three main reasons. The first reason for choosing CVSS is that it is an accepted standard. Secondly, it is widely used by most security professionals and has publicly available documentation that can be adopted by “organizations vulnerability risk assessment process”. Finally, the fact that it quantifies risk numerically in a generic manner makes it more attractive to adopt in organizations. Specifically, in this study, in relation to penetration testing, the fact that its “open-source”, “widely used” and “attempts to put metrics to weaknesses” makes it ideal for the threat modeling process.

2.4 CWSS (Common Weakness Scoring System)

The Common Weakness Scoring System (CWSS) is a scoring system similar to the CVSS. It is part of the CWE project which is sponsored by the “U.S. Department of Homeland Security” [21][22]. The CWSS is also organized into three metric groups. These metric groups are the “base finding metric group”, “attack surface metric group” and “environmental metric group” [22].

The “base finding metric group” attempts to “capture the inherent risk of a weakness” using five vectors [22]. This metric group is mainly focused on the technical aspect of a vulnerability. The other “attack surface metric group” which has six vectors attempts to capture the difficulty an attacker must overcome to successfully exploit the weakness [22]. Finally, the “environmental metric group” uses five vectors to contextualize and capture characteristics of the weakness [22]. The table below taken from mitre.org can summarize the vectors.

Table 1 CWSS metric groups summary (taken from MITRE documentation)

Base Finding Metric group		
Vector	Summary	
Technical Impact (TI)	The potential result that can be produced by the weakness, assuming that the weakness can be successfully reached and exploited.	
Acquired Privilege (AP)	The type of privileges that are obtained by an attacker who can successfully exploit the weakness.	
Acquired Privilege Layer (AL)	The operational layer to which the attacker gains privileges by successfully exploiting the weakness.	
Internal Control Effectiveness (IC)	the ability of the control to render the weakness unable to be exploited by an attacker.	

Finding Confidence (FC)	the confidence that the reported issue is a weakness that can be utilized by an attacker
Attack Surface Metric group	
Vector	Summary
Required Privilege (RP)	The type of privileges that an attacker must already have to reach the code/functionality that contains the weakness.
Required Privilege Layer (RL)	The operational layer to which the attacker must have privileges to use the weakness in an attack.
Access Vector (AV)	The channel through which an attacker must communicate to reach the code or functionality that contains the weakness.
Authentication Strength (AS)	The strength of the authentication routine protects the code/functionality that contains the weakness.
Level of Interaction (IN)	the actions that are required by the human victim(s) to enable a successful attack to take place.
Deployment Scope (SC)	Whether the weakness is present in all deployable instances of the software, or if it is limited to a subset of platforms and/or configurations.
Environmental Metric group	
Vector	Summary
Business Impact (BI)	The potential impact on the business or mission if the weakness can be successfully exploited.
Likelihood of Discovery (DI)	The likelihood that an attacker can discover the weakness
Likelihood of Exploit (EX)	the likelihood that, if the weakness is discovered, an attacker with the required privileges/authentication/access would be able to successfully exploit it.
External Control Effectiveness (EC)	the capability of controls or mitigations outside of the software that may render the weakness more difficult for an attacker to reach and/or trigger.
Prevalence (P)	How frequently this type of weakness appears in software.

The main advantage of the CWSS is its flexibility. This means it gives security professionals a way to score weaknesses without being reactive like the CVSS. For instance, the CVSS always assumes the worst case; meaning exaggerated scores for weaknesses with unknown values. In this particular study, the CWSS is not chosen due to its complexity and accessibility of the scores in vulnerability databases.

2.5 CWE (Common Weakness Enumeration)

The CWE (Common Weakness Enumeration) is a community-developed framework for vulnerability categorization [23]. The categorization of weaknesses list is mainly generated in relation to software or hardware weakness type [23]. This is distinct from the CVSS in that, different vulnerabilities with different CVSS scores can have identical CWE id. This mainly happens because CWE tries to make categorization of weaknesses based on their technical taxonomy [23].

Another, data that is provided by the CWE community is data showing the relationship among identified categories [23]. In this study, the focus is on the list of “related patterns” and “related weaknesses”. Additionally, the CWE community attempts to present the relationship between weaknesses in several different ways; but delving into the details and technicalities of the CWE

or CAPEC is not the goal of this study. Attached below are the top 25 vulnerabilities of 2021 taken from mitre.org's website.

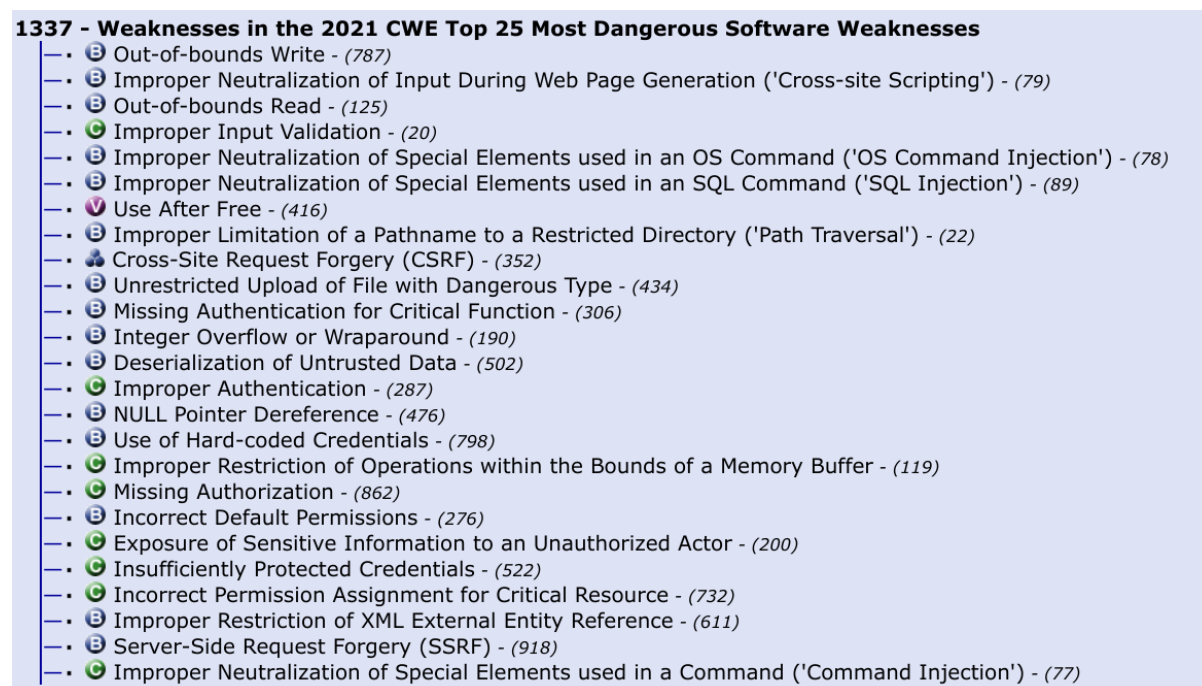


Figure 2 top 25 weakness types of 2021 (taken from [CWE website](#))

2.6 Penetration testing Standards (OWASP, NIST & PTES)

2.6.1 OWASP

OWASP normally crams the penetration testing process in two phases. The first stage is passive reconnaissance, which includes attempting to understand the application at hand by analyzing different types of data gathered by tools [3]. Following, it has a set of recommended tests that should be undertaken to ensure that the “web application” at hand does not have certain vulnerabilities [3]. This stage is termed the “active mode phase” in the ‘OWASP version 4’ testing guide [3]. The following table is a summary of OWASP recommendations in phase 1 and phase 2.

Table 2 OWASP penetration testing stages

Phase 1 Passive Mode	Phase 2 Active mode
trying to understand the applications <ul style="list-style-type: none"> • for instance, analyzing the HTTP/HTTPS request responses • banner grabbing 	<ul style="list-style-type: none"> -information gathering -Configuration and Deployment management testing -Identity Management testing -Authentication & Authorization testing -Session & input validation testing -Error handling testing -cryptography testing -Business Logic testing

2.6.2 PTES

The PTES standard generally consists of Intelligence gathering, Threat Modeling, Vulnerability analysis, Post Exploitation & Reporting [13]. The following figure below can help to visualize the PTES testing guidelines. It is important to note that the guideline recommends things to be considered as they are not meant to be binding as some can be added/omitted depending upon the specific circumstances.

Aside from recommendations regarding prearrangement interactions with the target network owner, which intends to sort out 'scope', 'legal' & 'other requirements' of the penetration testing process; the steps are somehow similar to the other testing frameworks.

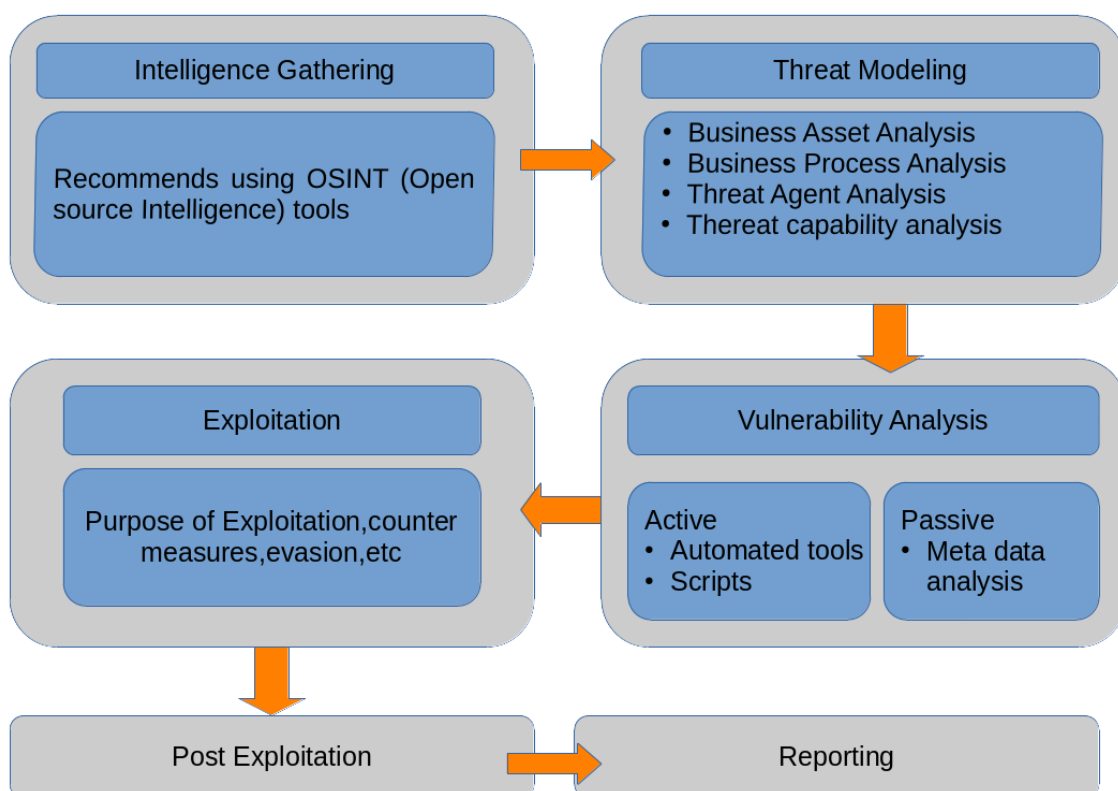


Figure 3 PTES standard [13]

2.6.3 NIST

NIST security testing guideline has a planning stage that sorts out rules of engagements and what kind of tests to be checked. Following this, the execution stage contains vulnerability validation and exploitation [12]. Finally, in the post-execution stage, mitigation measures and reporting are compiled [12]. The diagram attached below, taken from NIST 800-115 manual could aid the visualization of the testing process. In the picture below, 'the planning', 'discovery' and 'attack stage' have overlapping activities.

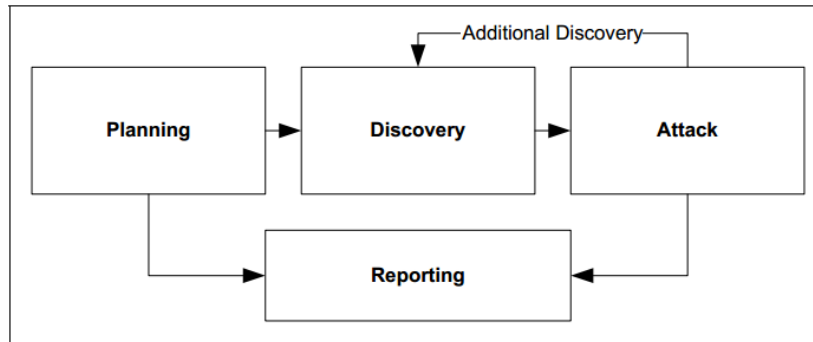


Figure 4 Taken from NIST security testing standard [12]

2.7 Related Works

Automated Penetrating testing was proposed by Ge Chu from the University of Liverpool using the BDI (Belief desire and intention) model to make the process more adept to the dynamic and complex environment of the target system [24]. The problem BDI model attempts to enable current penetration testing approaches is to help the process deal with ‘dynamic’ and ‘complex’ environments of operations [24]. In other words, current approaches that utilize vulnerability-scanned results with mapped tools cannot deal with the current dynamic environments.

The Belief sets normally consist of information gathered during reconnaissance [24]. Therefore, results found using scanning tools for information gathering like, types of services in the target; OS of the platform & so on are included in the belief set.

Following, based on the belief sets, desired ways of penetrating the system are chosen (which includes choosing an appropriate agent). It is important to note here that not all of the possible options in the desired set are used [24].

Finally, based on the intention of the agent, options from desire sets are executed using a plan which could trigger a possible improvement of the ‘belief set’ or finalizing a report [24]. A flow chart borrowed from Ge Chu can sum up the process implemented in the ‘BDI’ penetration testing process.

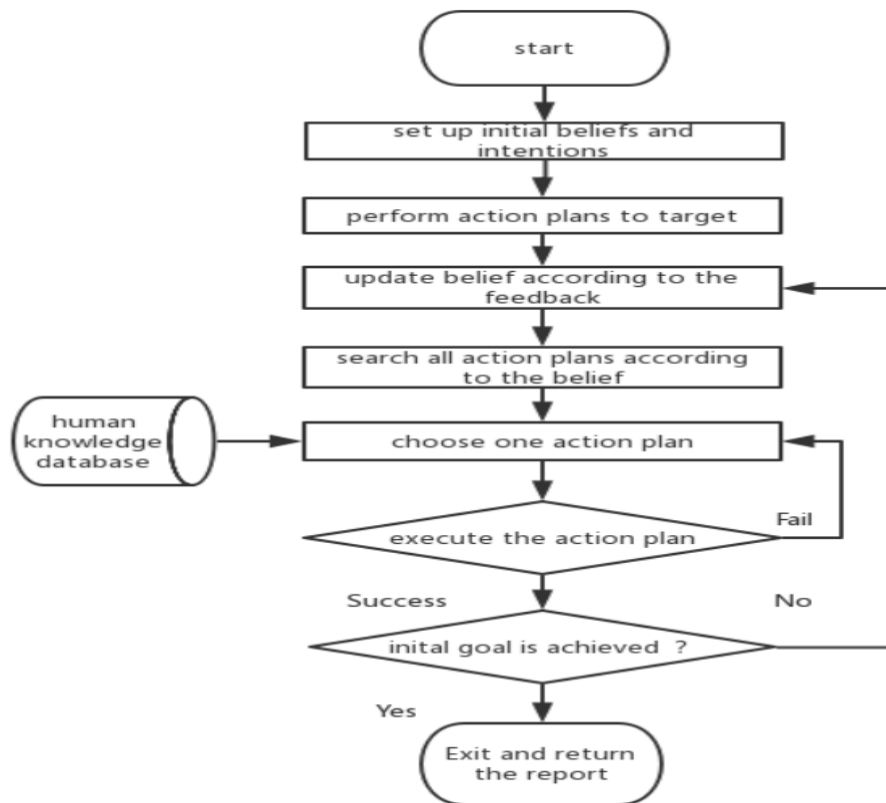


Figure 5 Agent based BDI model taken from Chu [24]

Another proposal which was made by Jai Narayan Goel and His Colleagues was the use of automated tools in vulnerability assessment using an ‘ensemble method’ [6]. In their research published in 2016 in the stage of vulnerability assessment in the penetration testing processes, they used many automated tools (open/propriety) to identify and check for weakness in the target system [6]. Figure 3 below, taken from their research can sum up the process they followed. It is important to note that the weight is a value assigned based on an ‘accuracy measurement’ technique recommended through ‘bench marking penetration testing tools’ from “literature” [6].

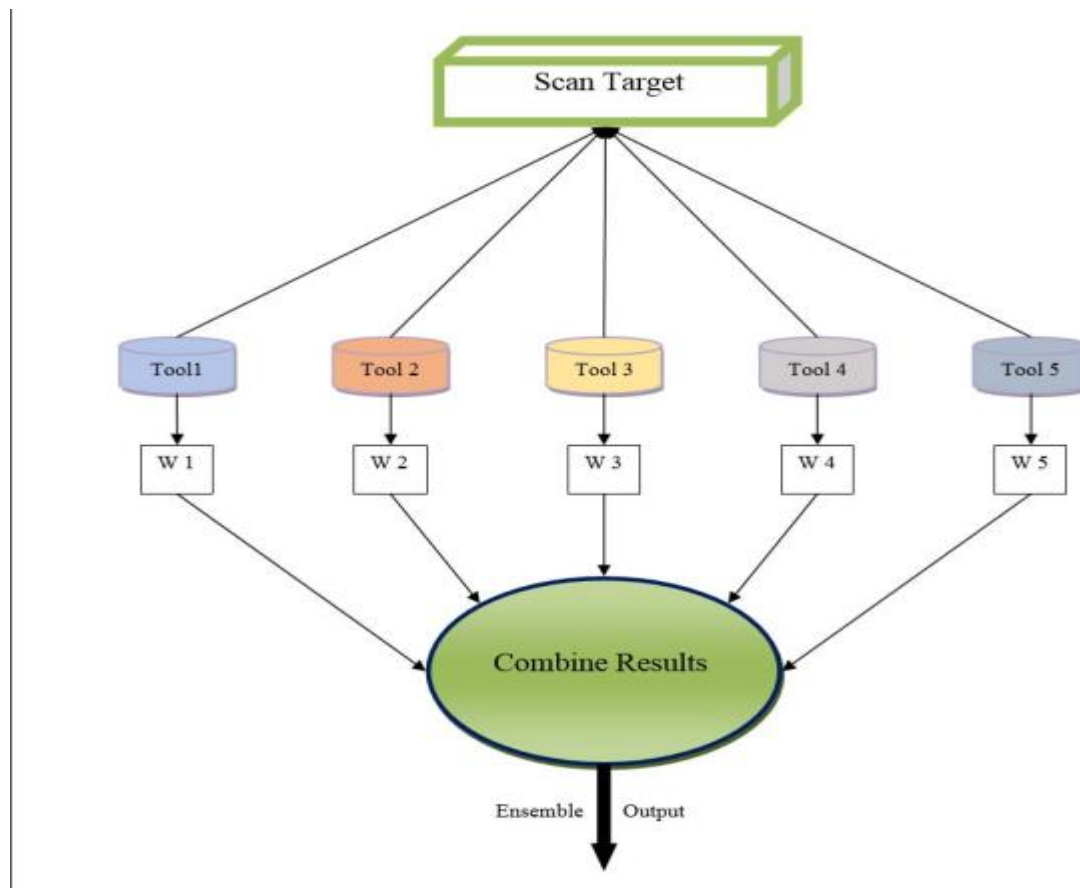


Figure 6 Ensemble testing approach taken from Goel [6]

Finally, the approaches taken by Ge Chun and Jai Goel recognize that vulnerability-scanning tools do not always discover weaknesses that need to be “considered” in a system during ‘penetration testing’. On the other hand, both approaches encourage performing different sets of vulnerability assessment tests to boost the discovery of most weaknesses.

2.7.1 Attack Graph-Based Risk Assessment

Attack graph-based risk assessment, specifically with regard to Cybersecurity is not a new thing [25][26]. Starting with the work of Mohammed Alhomidi, tried to solve the problem of “lack of methodology” that can be used to explore a “graph-based risk assessment model” [27].

The approach of Alhomidi relied on the genetic algorithm as a means of exploring risky paths [27]. The genetic algorithm approach is finding optimum solutions for a “path of attack” through ‘breeding’, ‘crossover’ and ‘mutation’ [27]. The approach used vulnerability attack likelihood and expected loss as a means of weighting a path [27].

The study by Alhomidi, Aksu, M. Ugur, and his colleagues (in 2017), further expounded the work by integrating the CVSS version 2, with an attack graph-based risk assessment [27][8]. The main problem Aksu tries to solve is that “risk assessment frameworks” by ISO and NIST are ‘high-level frameworks’; meaning their focus is mainly on auditing rather than quantifying risk [8].

The approach by Aksu consisted of two risk assessment models: These are “Base Risk Assessment Model” and the “Attack Graph-Based Risk Assessment Model” [8].

The base assessment model comprises four components, which are ‘assets’, ‘vulnerabilities’, ‘likelihood’, and ‘impact’ [8]. The broken line threat source box below is to depict that base risk assessment does not have a threat source component [8].

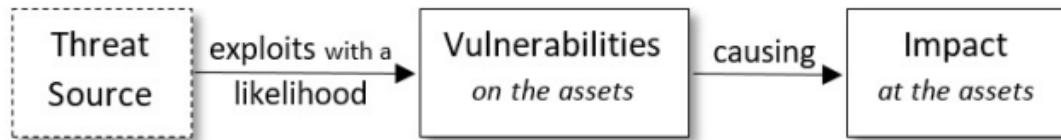


Figure 7 base risk assessment model taken from Aksu [8]

On the other hand, the “Attack Graph Risk Assessment Model” adds two more components to the “base risk assessment model” which are threat source and attack path [8]. The main goal of using the ‘attack graph risk assessment model’ was to quantify risks of the “attack paths” using vulnerability CVE with the CVSS [8]. Additionally, attack graphs show the “risk” of multiple weaknesses being chained together in an attack.

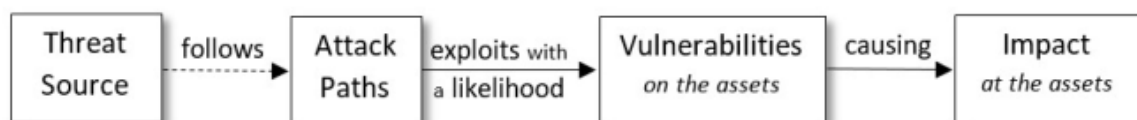


Figure 8 Attack graph risk assessment model [8]

The model considered assets as any computer found in the network (which are considered as vertex), vulnerabilities as weaknesses found on the assets (as defined by the NVD database) [8]. Then, used scanning tools like OPENVAS to identify vulnerabilities in the assets [8]. After, combining with NIST special publication 800-3rc1 recommendation of likelihood estimation and version 2 of CVSS scoring, they calculated the weights of (probabilities of a path along vertex) [8][27][28]. Fig attached below can aid the visualization of the path formation.

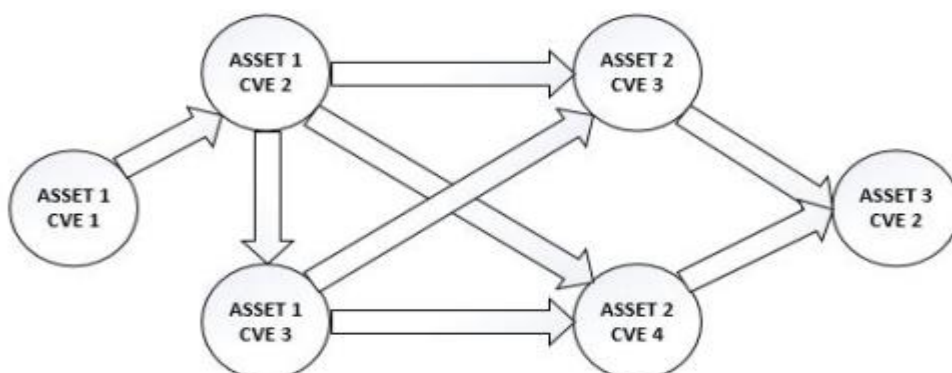


Figure 9 Attack vector path formation (taken from Aksu) [8]

The threat source and target component addition on the attack-based graph risk assessment by the author aids to simplify the risk assessment process. Additionally, the algorithms’ ability to

capture sequential and parallel attack paths within the network also helps security professionals in their plight of defending against exploration [8].

Finally, the work by Zhang Heng-Wei identified a problem that managers face during risk assessment. The main problem identified was managers were not able to identify key vulnerabilities efficiently to improve the safety of organizations.

The approach taken by Zhang (2016) took a similar approach to Aksu. The main difference occurs in how probability is calculated. Zhang used “vulnerability connectivity” and “vulnerability value” based on CVVS base vector values.

One of the drawbacks of the vulnerability-centric attack graph-based assessment is that, when the ‘number of nodes’ in question grows, processing vulnerabilities becomes daunting to handle. Mainly because a single vertex can contain several CVE weaknesses that could form edges with neighboring nodes. This means ‘possible paths’ to explore will grow drastically as the number of nodes in question rise.

Additionally, as the number of nodes in consideration goes up, identifying each combination of sequential/parallel ‘attack path’ even within the node is going to be a time-consuming task that impairs the “vulnerability risk analysis” problem. Additionally, it is going to get even more complicated, as exploring cyclic paths is going to be a challenge.

Finally, the main goal of this study is to develop a “systemic vulnerability assessment framework” for penetration testing on a chosen target with respect to its operating environment. While, being able to identify the path might be helpful, in reality, it is a challenging task to perform on time. Correspondingly, from the “penetration testing” point of view, might be unnecessary as well; since identifying the final weakness exploited and taking mitigation measures could deter attackers sufficiently.

.

Table 3 Summary of Related works

No	Author	Problem	Approach	Finding
1	Ge Chu (2019)	-the inability of penetration testing methodology to deal with dynamic and complex environments	-using an Agent-based penetration testing approach to improve performance by extracting from multiple domain experts	-similar output as traditional penetration testing
2	Jai Narayan Goel	-improve penetration testing cost -no single tool can discover vulnerabilities that need to be assessed	-improve penetration testing cost by using ensemble “vulnerability assessment and penetration testing process” using open-source tools	-demonstrated using open-source automated tools with an ensemble approach can outperform proprietary tools

3	Prashant S. Shinde (2016)	-overview of a vulnerability assessment process	-using automated vulnerability analysis with penetration testing	-vulnerability assessment generates an overwhelming data -also, false positives
4	M Ugur Aksu (2017)	-the inability of risk assessment frameworks (NIST and ISO) to quantify risk	-Propose graph-based Risk Assessment Methodology using the CVSS version 2	-can be more efficient as CVSS version 2 gets complete
5	Zhang Heng Wei (2016)	-to enable managers, to improve safety by identifying detrimental vulnerabilities	-quantify the risk of chained vulnerabilities using the connectivity operator (using CVSS version 2)	-using the connectivity operator to quantify is credible

CHAPTER THREE

RESEARCH METHODOLOGY

This chapter discusses the approach taken to develop a “systemic vulnerability assessment” process for penetration testing using the mental model proposed by Ken Peffer in the Design science research model.

3.1 Design Science Research

Regarding research conducted in Information systems ‘Behavioral’ and ‘Design science’ have dominated the discipline [29]. The Behavioral science paradigm is normally reactive, which means it explains and attempts to solve problems assuming the conditions of human behavioral laws are stable. Hence, it fails to recognize additional “behavioral laws” spawned by the introduction of an IT artifact [29]. On the other hand, the Design science realm attempts to solve that problem by anticipating possible unforeseen changes that are instigated due to the “introduction of the IT artifact” [29]. The picture below can best summarize the framework; taken from Hevner’s 2004 journal article.

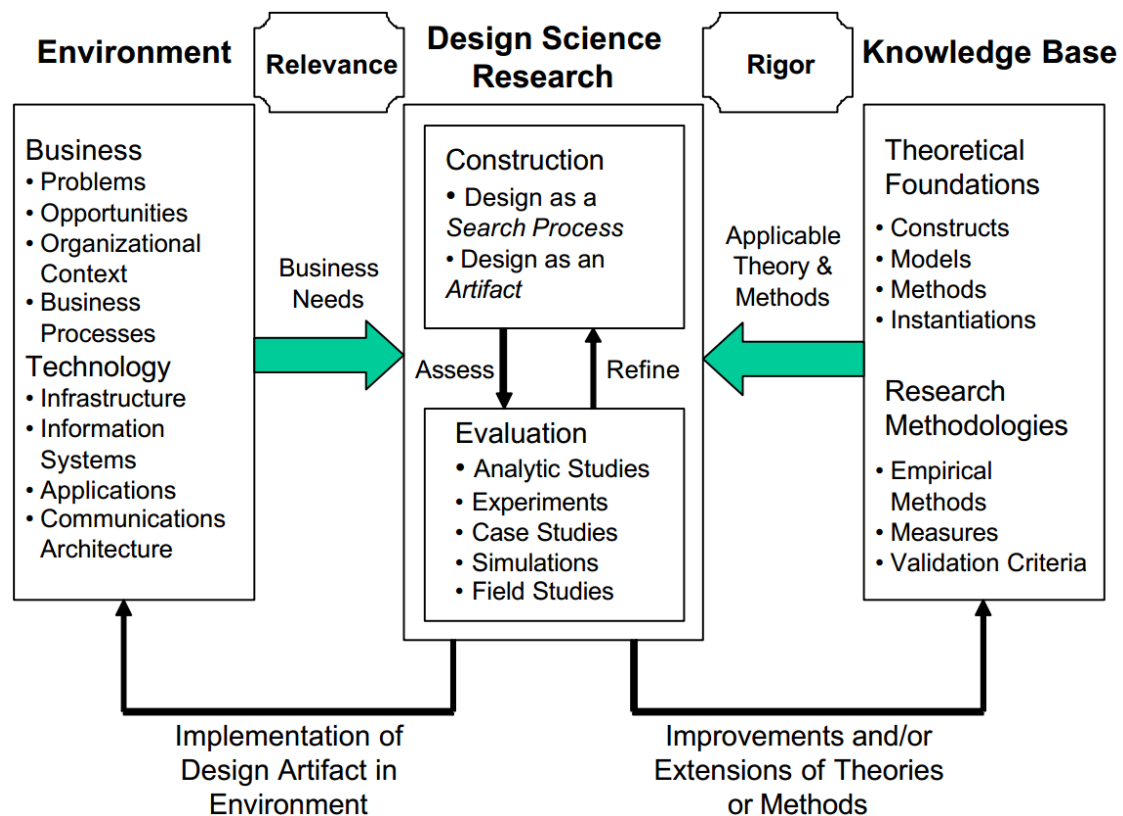


Figure 10 Design Science Framework taken from Hevner [29]

Here in this study, the framework is considering the ‘systemic vulnerability assessment framework’ for penetration testing as an artifact, which operates with a given target environment.

Thus, the methodology to follow in this study uses the mental process model for the design science paradigm by Ken Peffers [30], which can be visualized in the figure below.

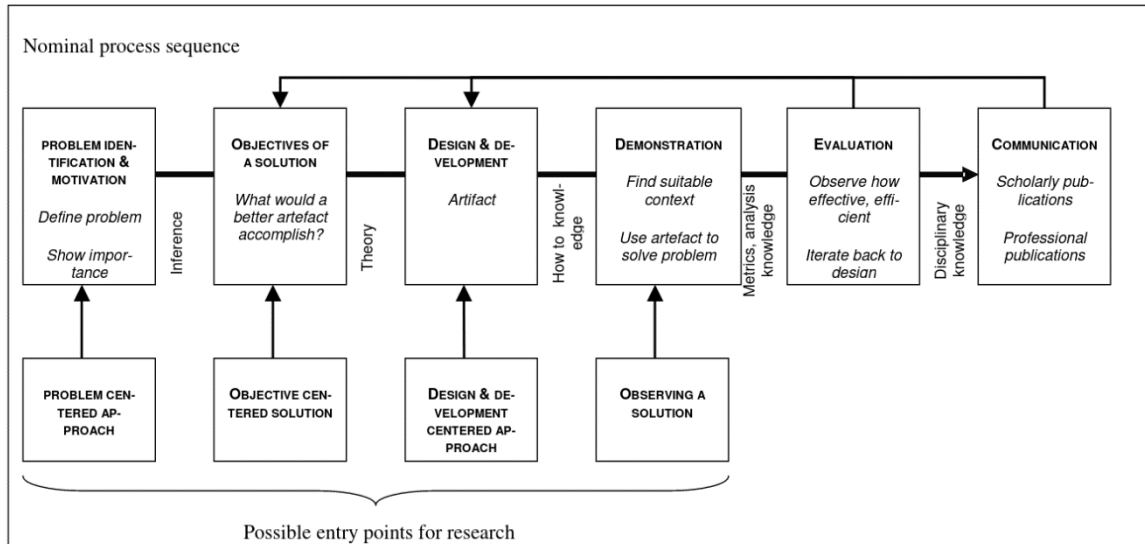


Figure 11 DSR research methodology model by Ken Peffers [30]

3.2 Problem Identified

The main problem identified in traditional “vulnerability assessment and penetration testing” process is:

- The drastically growing number of vulnerabilities decreases traditional testing process efficiency. This means, merely using “automated vulnerability assessment testing tools” is not enough as compared to vulnerabilities that need to be checked.
- Traditional penetration testing “vulnerability assessment” processes do not capture the risk of vulnerabilities with respect to weaknesses found in neighboring nodes.
- Traditional vulnerability assessment frameworks and standards don’t have go-to standards that compare “risks of vulnerabilities” numerically.

3.3 Objectives of the solution.

The main objective of the study is to design a “systemic vulnerability assessment framework”. The main desired features of the framework are-

- Ability to consider all known “vulnerabilities” found in the “platforms” running. The platforms in this context refer to the types of software/hardware that are being utilized by the “system of interest”.
- Ability to quantify “risk of vulnerability” on target with respect to neighboring node weaknesses.
- improve performance of penetration testing process in terms of time and cost (by formulating a framework to process large amounts of data generated by “vulnerability assessment data”)

3.4 Data source, collection, and Analysis

The data collection consists of two types of datasets. The first type of dataset collected is collected from vulnerability databases. The second type of dataset is scanned data gathered from scanning different nodes from different networks in Addis Ababa. This synthetic dataset is used for demonstration. The first dataset collected from the databases is used in the data

analysis before framework development. In addition, it is also used as a ‘source of vulnerability’ for the framework.

3.4.1 Data Collection for Demonstration and Evaluation

3.4.1.1 Sampling

Given the type of information gathered about the “vulnerabilities” on the NVD database; information about the type, product, and version number are recorded. Thus, findings during reconnaissance can be used for “vulnerability identification”.

Therefore, knowing what kind of information is needed and the holistic approach employed in this study; one must tactically sample nodes, which should be considered. Generally, during data collection, there are two ways of choosing a sample, namely probability sampling and purposive sampling [31]. In probabilistic sampling, it is assumed every element in the population has an equal chance of being randomly selected, whereas in purposive sampling “description rather than generalization” is intended [31].

In this particular case, a mixed-up approach of the aforementioned sampling techniques are used. The main reason for sampling is to find the description of the kind of “system”, “services” and “devices” in the network. Therefore, using sampling using a probabilistic approach only could skew the data, as scanning two similar devices running the same services will only be a waste of valuable time. On the other hand, purely scanning targeted sections could omit some devices that could be meaningful. Therefore:

1. Form cluster of similar devices together found during passive reconnaissance (using ping sweep for instance). This is where purposive sampling technique is employed. Therefore, IP addresses considered unique devices like “router”, “firewall” and “server” are grouped.
2. Following, randomly select a node from a group of clusters found during step one and perform “service” and “version” scanning on all ports, OS & device detection scanning.

To summarize, the main goal of the “sampling process” in the above targets is to get the majority of unique devices in the network before going on to the active “reconnaissance process”.

3.4.1.2 Reconnaissance

Reconnaissance (passive/Active) is one of the most common steps that is practiced in ‘traditional penetration testing’ [32][13]. It usually comes before enumeration and exploitation [32][13]. As the goal of this study is to design a “systemic vulnerability assessment framework” (within the context of ‘black box’ penetration testing), information is gathered through “passive and active reconnaissance”.

In the passive reconnaissance, phase the main information to be salvaged are-

- Identification of IP ranges
- router IP addresses
- Web server’s hostnames (IP addresses)
- Firewalls
- DNS servers

- Email Servers

Then, once the above information about the devices is gathered; the active reconnaissance session attempts to categorize the devices into groups and take a sample to perform “detection”, “version detection” and “services” on selected devices.

3.4.2 Data Collection for Framework Development

The vulnerability data in this study are gathered from “[NVD](#)” and “[cvedetails](#)” databases. These datasets contain CVSS scores for each identified CVE ids.

3.5 Missing Values

Finally, it should be known that the CVSS version 3 was introduced in 2015. This means, that most of the CVEs in the NIST NVD do not have CVSS3 Scores. Thus, in the event of CVE ids with no version 3 score, version 2 scores of the corresponding CVE id are going to be used. In cases of “privileges required” and “user interaction” vectors, which are new to version 3, the worst-case scenario value is assumed. In addition, CVE details database, version 2 scores were converted to version 3.

In cases where version 3 or version 2 scores do not exist, those CVE ids found are presented separately.

3.5.1 Analysis

The analysis of this study consists of two statistical analysis sections. The first analysis depends on data gathered from an open-source vulnerability database. The second one utilizes synthetic data collected from different networks in Addis to get real-time reconnaissance data. It should be noted, the reconnaissance datasets are not specific to any particular network, but the results can truly be representative of devices that could be contained in real-time networks.

3.5.2 Tools

The tools listed below are the main ones used for data collection, analysis & design of the framework.

- Scrapy is an open-source web crawling framework developed in python. Scrapy to scrap data relevant to the list of vulnerabilities published by MITRE Corp (nonprofit organization) or NVD.
- Nmap is also a free network scanner created by Gordon Lyon. It uses python and Lua scripting for additional features
- Jupyter notebook(python) along with several relevant python data processing libraries is utilized
- Other tools used are dictated based on the vulnerabilities deemed relevant to the target.

3.6 Design and development of Framework

Based on the above-discussed core aspects of the problem, a “systemic vulnerability assessment framework is proposed with the following main points in mind: -

- reviewed literature
- main objectives of the frameworks
- extracted insight from data analysis

3.7 Demonstration and Evaluation

The demonstration of the framework was performed using synthetic data (data from reconnaissance). Additionally, ‘python script’ written based on the framework is utilized to demonstrate “how to use the framework”.

The evaluation mainly consists of statistical and logical arguments based on the output of the use case result.

3.8 Communication

Finally, the results of the finding are “reported” and “presented” in a form of a thesis to the “school of Information Science”.

CHAPTER FOUR

DATA ANALYSIS

This chapter discusses the result of data analysis performed on datasets from the vulnerability databases. Moreover, a simple analysis of the CVSS score outcome based on the base vector variations is also processed and summarized.

4.1 Analysis

4.1.1 Summary of Data from Databases

The table below shows the summary of data collected from two different vulnerability database sources.

Table 4 Summary of used data

Data Source	
NVD Database	CVE Details
-ranging from 2002 E.C. -2021 E.C. -total 166,903 vulnerabilities -79011 vulnerabilities with CVSS scores and CWE ID	- ranging from 1999 E.C.-2020 E.C. -total 139,550 vulnerabilities -84,717 identified CWE ids -119,000 CVSS version2 scores
After merging scores based on CVE id	
-total scored vulnerabilities after converting to version 3 152,987 (91.67% of the total vulnerabilities) -total number with identified CWE id is 124,431(74.5% vulnerabilities NVD)	

4.1.2 NVD vulnerabilities statistics & summary

NIST maintains National Vulnerability Database (NVD) along with their corresponding CVSS scores. The database has several lists of published weaknesses starting from 2002. In their database, the organization has formally adopted the CVSS as a scoring framework for published vulnerabilities. In the coming sections, a summary of the CVSS score vectors distribution insights is discussed.

Analysis based on CWE id shows nearly half of seven types of the categories cover 49.91% of the vulnerabilities. This shows that from the point of taxonomy (technical attributes), there is at least a 50% chance that an attacker will utilize a vulnerability that has one of these seven CWE ids. The number of CWE ids of vulnerabilities found in the database can be summarized as in the fig below.

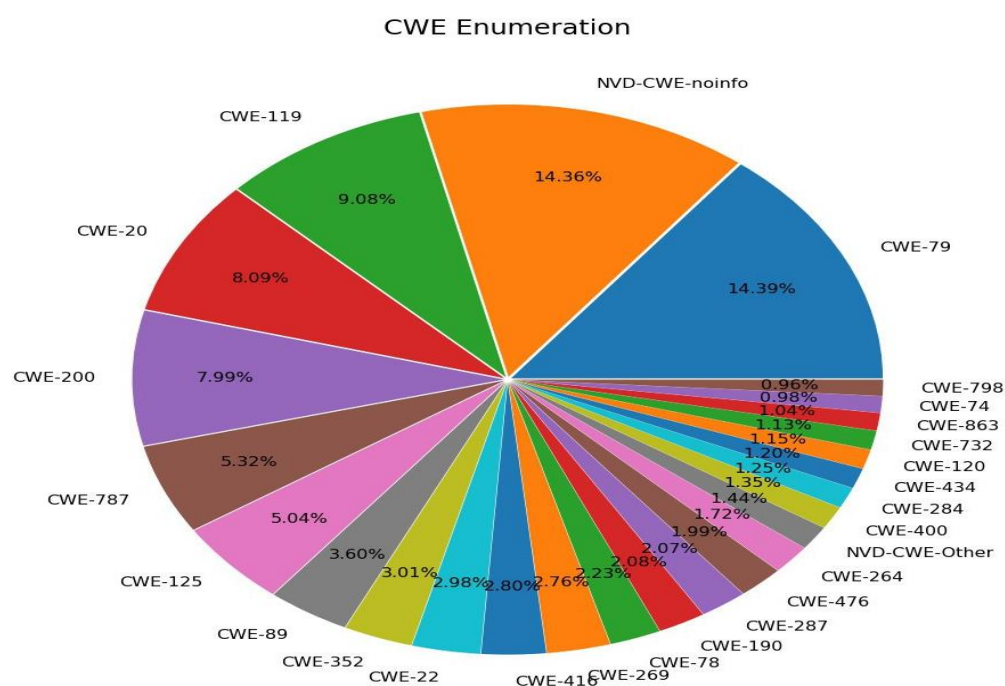


Figure 12 CWE Id category found in NVD database

Starting from Impact metrics values: It is seen that the CVSS gives more weight to “Impact metrics” than the “Exploitability metrics” when scoring. Further generation of a correlation matrix shows the “Integrity” vector is moderately positively correlated with both “Confidentiality” and “Availability” metrics; with Pearson correlation values 0.54 and 0.51 (alpha values less than 0.05 on both occasions). The figure below is a summary of the correlation matrix.

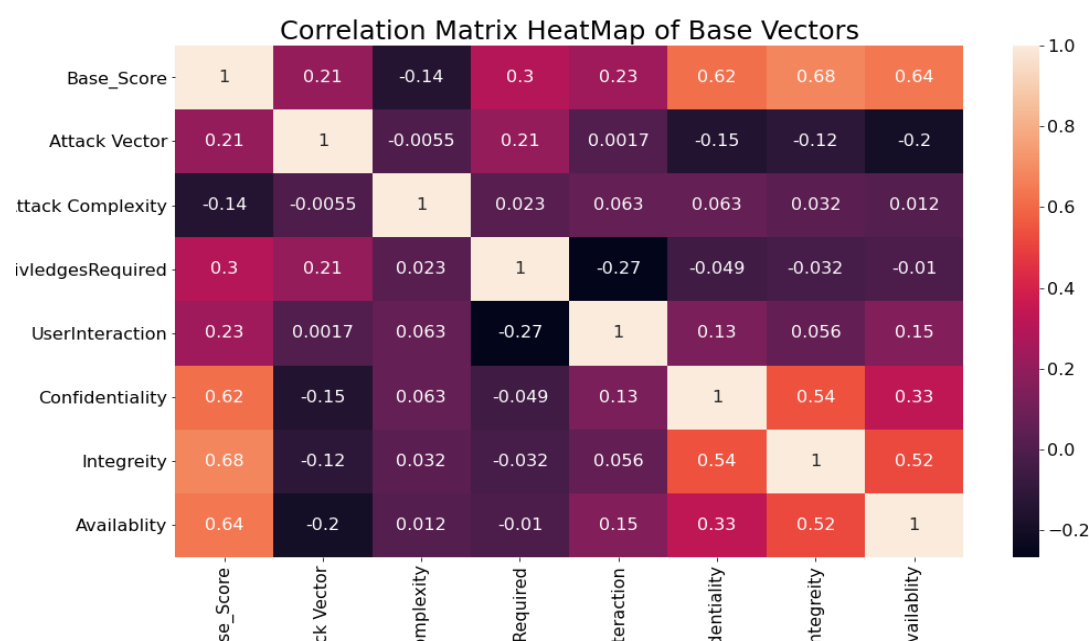


Figure 13 Correlation matrix heat map based on NVD database CVSS version 3 Base metric vectors (2002-2021)

Another useful visualization is the distribution of “level of severity” vulnerabilities found in the database. FIRST org rating based on CVSS version 3 scores have categorized severity into four categorical groups. The groups are, low (base score <4), medium (four<bases core <7), High (7< base score <9) and critical (base score >9). Distribution of severity based on these, the ratings show that more than 80% of vulnerabilities have a severity “high” or “medium”

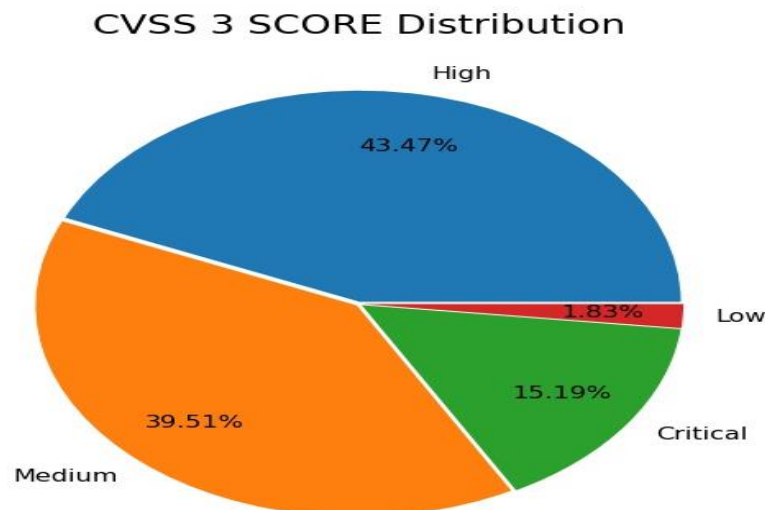


Figure 14 Severity counts of vulnerabilities from NVD database (2002-2021)

impact. Additionally, vulnerabilities with an impact low account only for less than 2% of the vulnerabilities. The pie chart above shows severity distribution based on the CVSS base score of Vulnerabilities.

Another interesting way of visualizing the “severity” of the score is with regard to the CVSS vector values, mainly the “Attack Vector”, “Scope”, “Privileges required”, and “Integrity” metrics. The graph below shows the categorical distribution of severity along Integrity.

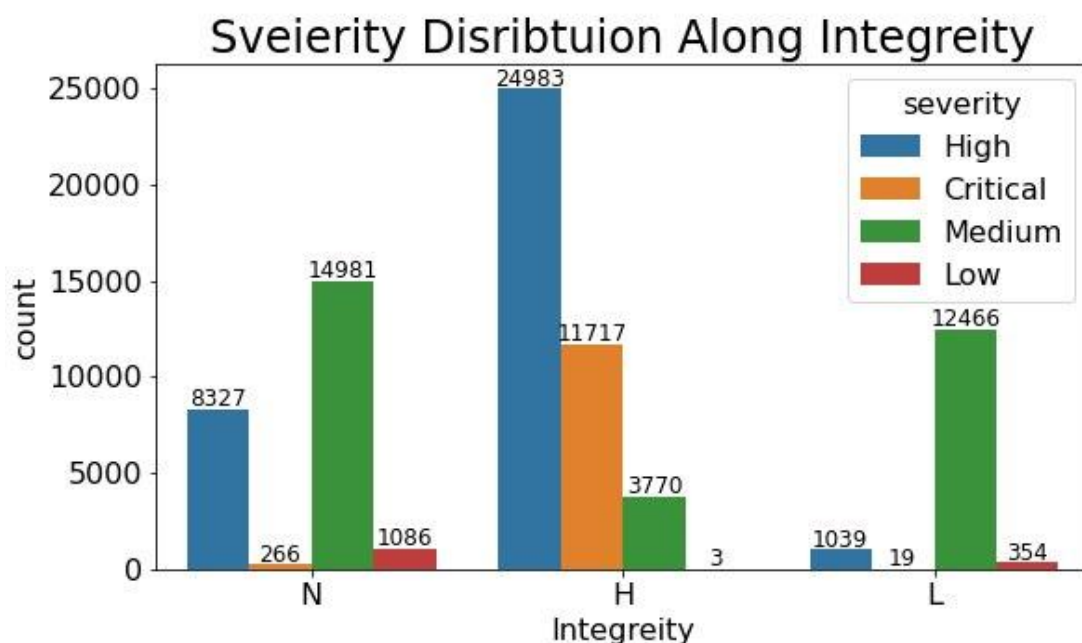


Figure 15 Severity along Attack Integrity based on NVD data (2002-2201)

The above graph shows the higher the impact metrics score, the higher the “base score “of a vulnerability. Additionally, the distribution of severity of “vulnerabilities “based on “attack vector” shows that more than 96% of weaknesses can be launched from either the internet or need “local access (like from ssh)”. The figure below shows this distribution.

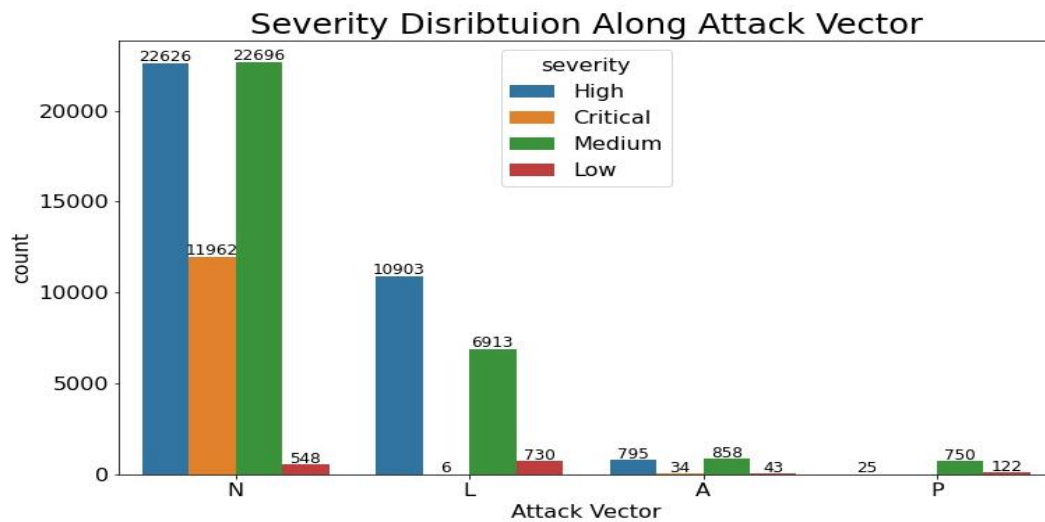


Figure 16 Severity along Attack vector based on NVD data (2002-2201)

The other main metrics considered are “Privileges required” and “Scope”. It is seen from the severity count graph that most CVEs with scope value “C (changed)”, have base scores, which are above four. On the other hand, more than half of the CVEs with a “PR” value of “N” have a base score greater than four. The graphs shown below are severity counts with respect to “scope” and “privileges required “vector value variations.

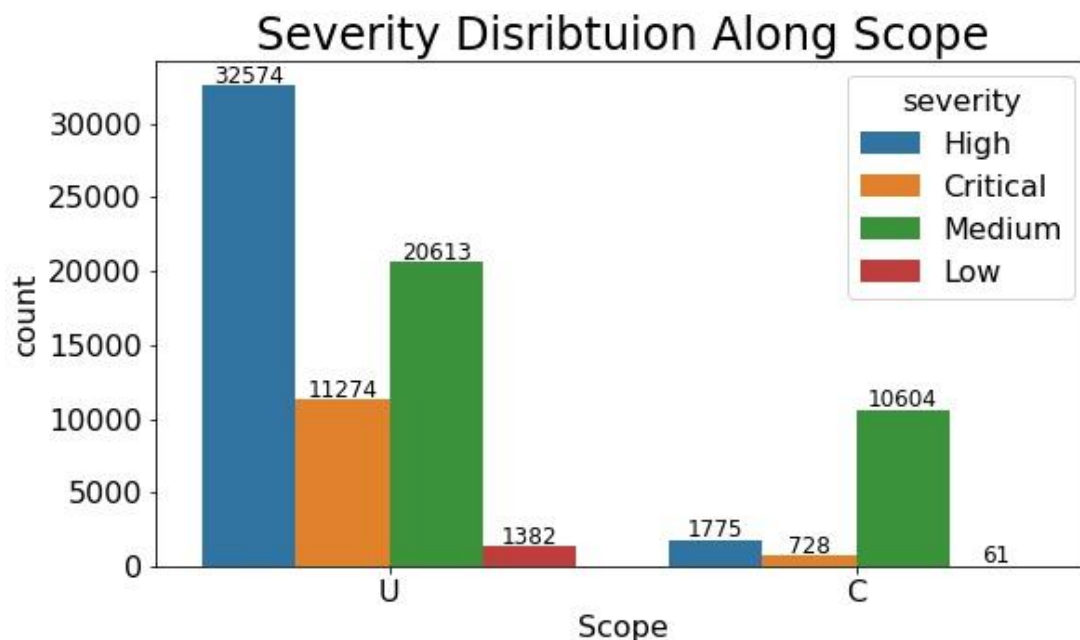


Figure 17 Severity along Scope based on NVD data (2002-2201)

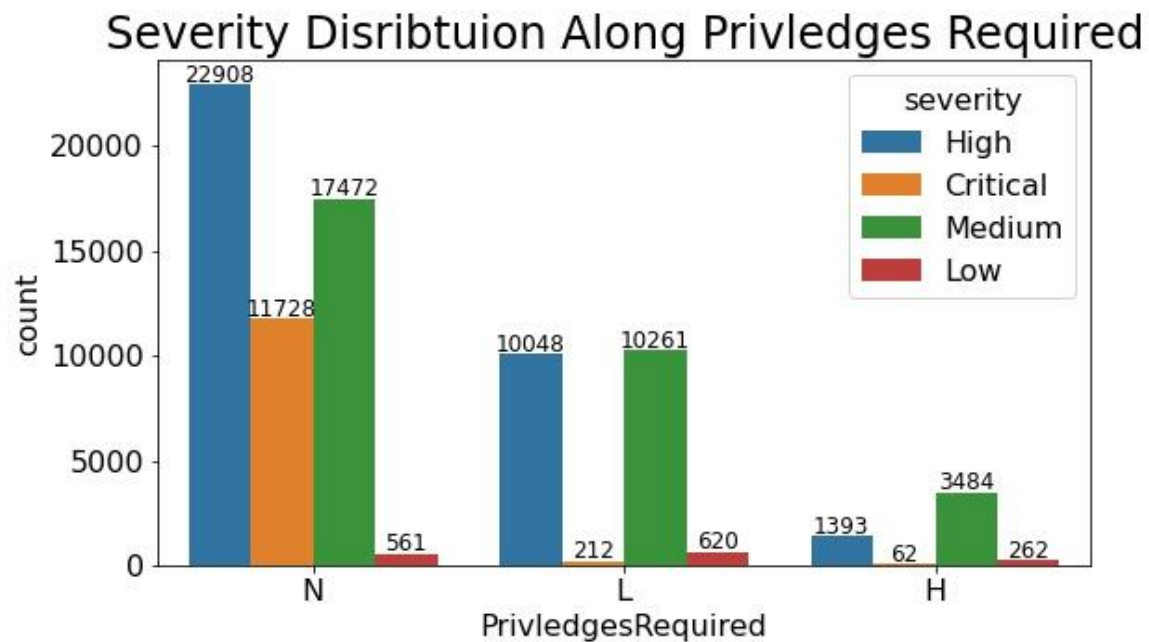


Figure 18 Severity along Privileges required based on NVD data (2002-2201)

4.1.3 CVSS Score Analysis

The section discusses analysis based on the CVSS “base vectors” possible outcomes. Given the way the CVSS generates severity vectors, there can only be 2592 unique “base vector” values. Additionally, based on the formula provided by CVSS to convert these vectors to metrics, there can only be 84 possible numeric outcomes. This means there are several “base vector” values that can share the same score. The figure below shows the number of vector combinations that can assume the same numeric CVSS severity score.

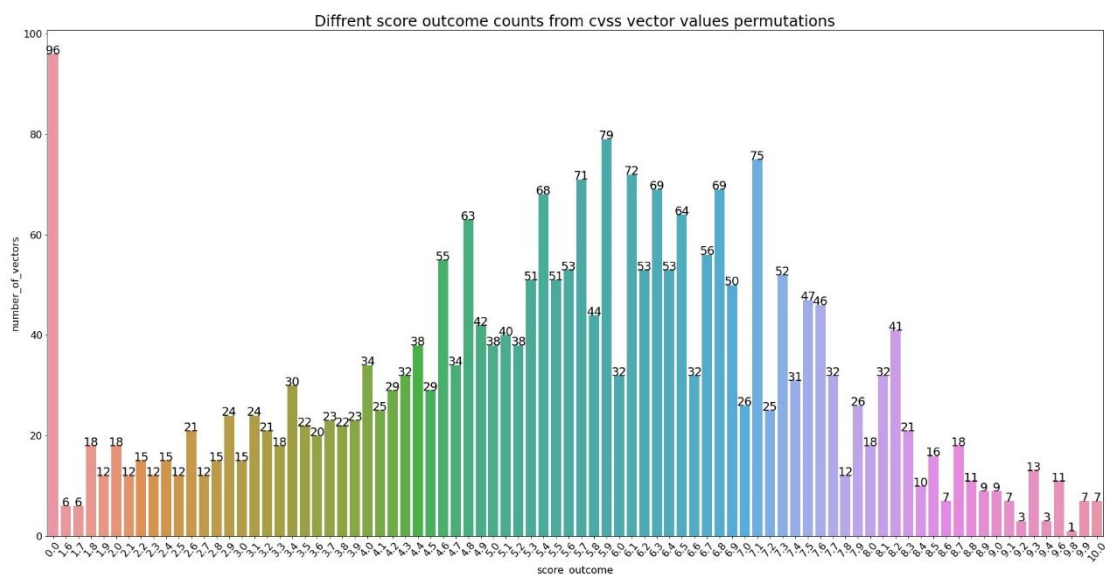


Figure 19 Number of Vector counts that can be assume same severity scores

Another interesting insight can be found by generating the sample space of vectors for the event of chaining only two vulnerabilities. Generating frequency distribution of the graph for similar

numeric outcome vectors shifts the mode to the right. This clearly shows the severity of a path that can be formed using ‘combinations of vulnerabilities likely to get severe. Hence, this can be useful insight because it shows that analyzing possible “attack paths” during risk assessment is not fundamentally necessary. This is mainly because the likelihood of the “severity” being high will rise as the number of vulnerabilities being used together grows. The figure below shows, the pool of CVSS scores after being chained using two chains.

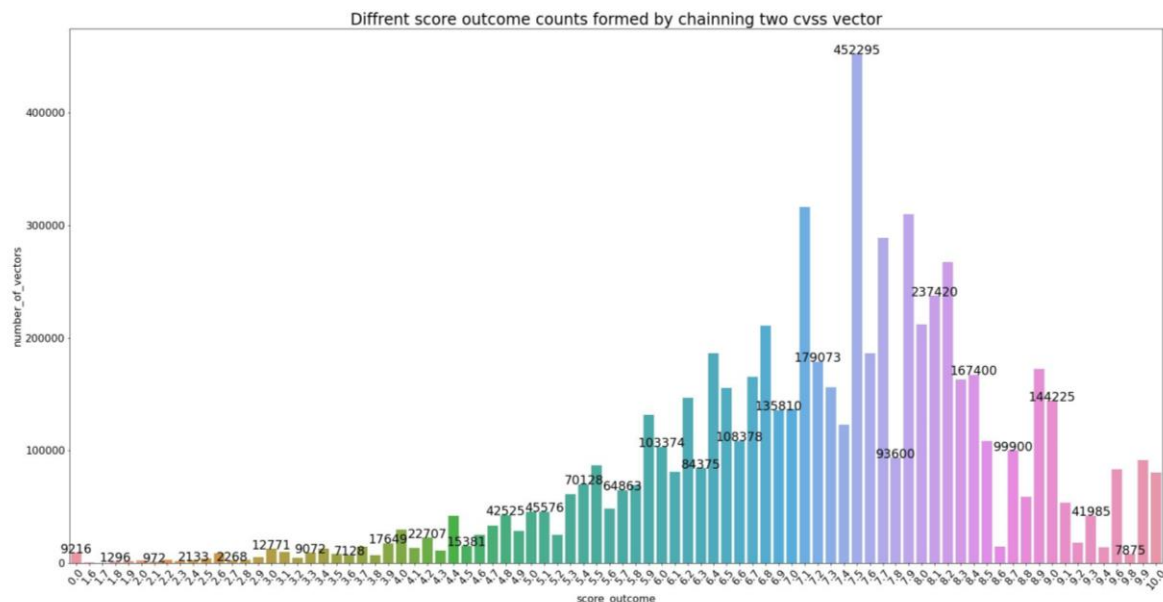


Figure 20 Number of Vector counts that can be assume same severity scores (when chaining two base metric vectors)

4.2 Summary

This chapter discussed data analysis performed before proposing a framework. Thus, the main points to be taken from the data analysis can be summed up as follows:

- The majority of known vulnerabilities have similar taxonomy. This is with respect to their CWE id
- Most severe vulnerabilities are launched from the internet or through gaining local access.
- The CVSS mainly emphasizes on “impact metrics” vectors. That is the system gives more weight to attacks that breach the CIA triad.
- Finally, the scoring system analysis shows that during the event an attack uses “multiple vulnerabilities”, the severity is more likely to have a score of more than 4.

CHAPTER FIVE

DESIGN AND DEVELOPMENT OF FRAMEWORK

This chapter discusses the main concepts utilized in the framework development and discussion of the framework.

5.1 Graph-based “Vulnerability Assessment Framework”

Earlier, it has been described that an “Attack graph-based risk assessment” employed a vulnerability/asset-centric approach for risk assessment by Aksu. Therefore, the whole risk modeling process consisted of identifying possible probabilistic paths from the ‘threat source’ to the ‘target’. In the coming sections, concepts of the Attack vector, Scope, Impact, and Vulnerability chaining used in the framework are discussed.

In the CVSS, to take into account from where any exploitation attack originated, the authors used the Attack vector. The attack vector (AV) broadly categorizes attacks instigated to be from “Network/Internet(N)”, “LAN(A)”,” Local(L)” or “physical(P)”, from the OSI model point of view. Correspondingly, when scoring individual vulnerability, the assumption is that the possible number of attackers for weaknesses with “attack vector” value “network” is greater than a weakness with value “physical” [20][33]. Consider the table below, which can aid the visualization process.

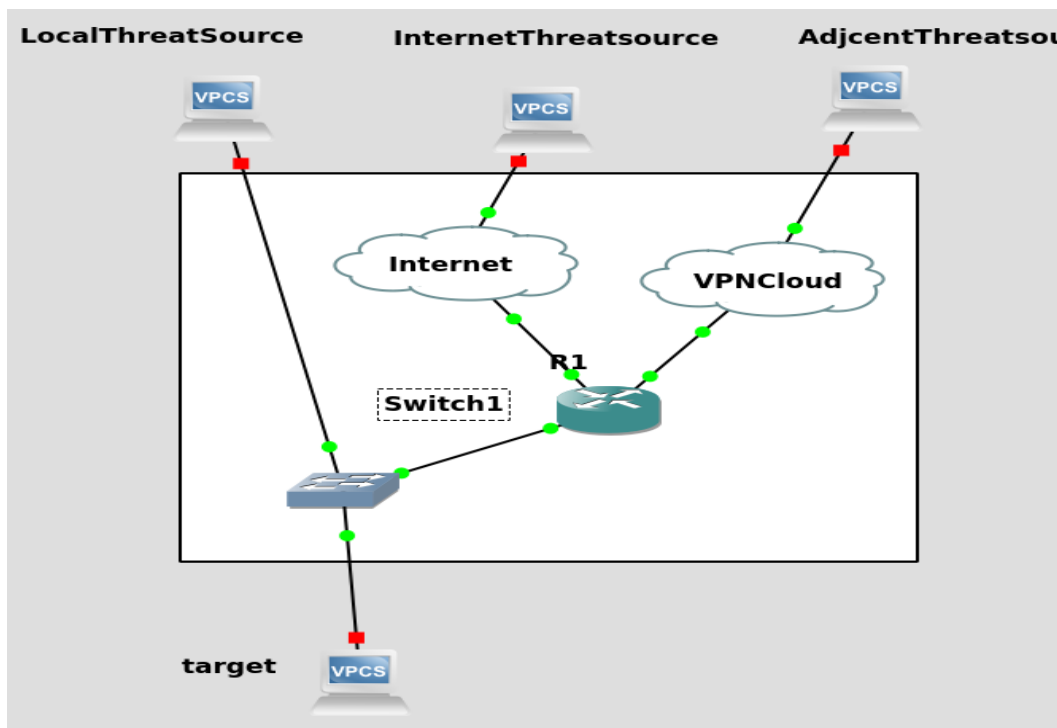


Figure 21 Threat source model based on the CVSS Attack vector

Additionally, it should be noted that “attack vector “values “local (L)” and “physical (P)” are values somewhat similar. The slightly nuanced difference between them being, a “weakness” assuming local value probably requires some kind of user interaction, or remote access (like

ssh), to launch a successful attack [33]. Below is the description of the metrics from the CVSS specification guide.

Table 5 CVSS Attack Vector values definition (taken from CVSS specification) [20]

Metric Value	Description
Network (N)	The vulnerable component is bound to the network stack and the set of possible attackers extends beyond the other options listed below, up to and including the entire Internet. Such a vulnerability is often termed “remotely exploitable” and can be thought of as an attack being exploitable <i>at the protocol level</i> one or more network hops away (e.g., across one or more routers). An example of a network attack is an attacker causing a denial of service (DoS) by sending a specially crafted TCP packet across a wide area network (e.g., CVE-2004-0230).
Adjacent (A)	The vulnerable component is bound to the network stack, but the attack is limited <i>at the protocol level</i> to a logically adjacent topology. This can mean an attack must be launched from the same shared physical (e.g., Bluetooth or IEEE 802.11) or logical (e.g., local IP subnet) network, or from within a secure or otherwise limited administrative domain (e.g., MPLS, secure VPN to an administrative network zone). One example of an Adjacent attack would be an ARP (IPv4) or neighbor discovery (IPv6) flood leading to a denial of service on the local LAN segment (e.g., CVE-2013-6014).
Local (L)	The vulnerable component is not bound to the network stack and the attacker’s path is via read/write/execute capabilities. Either: <ul style="list-style-type: none"> • the attacker exploits the vulnerability by accessing the target system locally (e.g., keyboard, console), or remotely (e.g., SSH); or • the attacker relies on User Interaction by another person to perform actions required to exploit the vulnerability (e.g., using social engineering techniques to trick a legitimate user into opening a malicious document).
Physical (P)	The attack requires the attacker to physically touch or manipulate the vulnerable component. Physical interaction may be brief (e.g., evil maid attack ¹) or persistent. An example of such an attack is a cold boot attack in which an attacker gains access to disk encryption keys after physically accessing the target system. Other examples include peripheral attacks via FireWire/USB Direct Memory Access (DMA).

5.1.1 Vulnerability Chaining

Vulnerability Chaining is a concept added to CVSS version 3 as per the recommendation of the SIG [20][33]. It tries to model circumstances where multiple vulnerabilities are used together to launch a successful attack [20][33]. It is more similar to the work of M Ugur Aksu, who attempted modeling the “chaining” through an attack-graph risk assessment technique. The main difference is Aksu used CVSS version 2 and provided a probabilistic way of calculating an attack path. On the other hand, the CVSS version 3 chaining leaves the responsibility of verifying/quantifying whether two vulnerabilities can be chained together on the analyst using the CVSS [33].

Additionally, when performing chaining, the CVSS team recommends assuming the worst-case scenario [33]. Consequently, taking a vector value with the “highest risk” is to be taken from the values being compared. For instance, if two CVE with “attack vector” values N (Internet) and P (Physical) are to be chained, the new attack vector value should assume the N (Internet). Additionally, using CWE (Common Weakness Enumeration) in tandem with CVSS could also be used to aid the chaining process, thus the chain formed is likely to be formed if those CWE ids are related in some way.

It is clear from above that, the vulnerability chaining provided by CVSS version 3 lacks a clear way of chaining, as the burden of proof relies completely on the analyst. Additionally, using it to refine a list of vulnerabilities to check upon, verifying if vulnerabilities can be chained together is not feasible, especially when analyzing paths that involve multiple vulnerabilities. Therefore, coming up with some assumptions based on score distribution in the analysis to gauge the likelihood of chaining could simplify reducing the load of data that is formed by graph-based vulnerability risk assessment.

Furthermore, the score analysis has shown that, as the number of vulnerabilities that are involved in a path increases, the severity increases accordingly. This means, there is no need to assess several step paths since longer chains are likely to have a severe impact. Correspondingly, measuring to what extent the weakness contributes to a path by averaging chained score variation caused by a weakness in a single step can help gauge the effect of a CVE. The single step in this context refers to the chain that is formed by only two vulnerabilities.

5.1.2 Exploitability and Impact vectors of the CVSS

Taking into consideration the “worst-case scenario” recommendation in chaining CVE, this study is going to add more refining criteria using insights from the score distribution of the metrics from the NVD vulnerability database. In addition, focus given to the base metrics values of the CVSS as the characters of the vectors can help develop intuitions while chaining vulnerabilities.

The main exploit-ability vectors metrics considered in the chaining process are the “Privileges Required”, “Attack Vector”, “Attack Complexity” and the “Scope” Metrics. Mainly, because the “vulnerability connectivity” operator by Zhang uses these vectors to calculate probability.

Starting with privileges required metrics in the CVSS; it is a vector, which can assume three distinct values. It is a vector that attempts to capture the minimum privileged that is required to successfully exploit the known vulnerability [19][20][33]. The values that it can assume are “None”, “Low” and “High” representing situations that require “minimum or guest”, “user” and “admin” privileges respectively [33].

Table 6 CVSS Privileges Required values definition (taken from CVSS specification) [20]

Metric Value	Description
None (N)	The attacker is unauthorized prior to attack, and therefore does not require any access to settings or files of the the vulnerable system to carry out an attack.
Low (L)	The attacker requires privileges that provide basic user capabilities that could normally affect only settings and files owned by a user. Alternatively, an attacker with Low privileges has the ability to access only non-sensitive resources.
High (H)	The attacker requires privileges that provide significant (e.g., administrative) control over the vulnerable component allowing access to component-wide settings and files.

Additionally, in the CVSS, a CVE with a lower PR vector value is likely to get a higher score. Moreover, the “vulnerability connectivity” operator in the chaining process emphasizes “High” and “Low” values, as these privileges are used as a pretense to exploit other vulnerabilities. This can simply be reflected in CVSS scores after chaining as well.

The other Exploitability metric considered is the scope metric. This metric is an addition to CVSS version 3[19]. The main attribute it captures is if the defined vulnerability affects the machine beyond the “components security scope” [[19][33]. A Linux OS (Ubuntu) running on windows virtual machine can be a good example. Thus, if a weakness in the Linux OS virtual machine lets attackers execute commands on the windows machine, then the scope has changed (which usually manifests in the PR vector). Below is the formal definition of scope metrics values from FIRST.org.

Table 7 CVSS Scope Vector values definition (taken from CVSS specification) [20]

Metric Value	Description
Unchanged (U)	An exploited vulnerability can only affect resources managed by the same security authority. In this case, the vulnerable component and the impacted component are either the same, or both are managed by the same security authority.
Changed (C)	An exploited vulnerability can affect resources beyond the security scope managed by the security authority of the vulnerable component. In this case, the vulnerable component and the impacted component are different and managed by different security authorities.

5.1.3 Systemic Vulnerability Assessment Framework

Now considering “threat source”, “target”, “Exploitability”, “Impact metrics” and “vulnerability chaining”; the coming sections discuss the concepts about the framework developed along with assumptions.

The desired features of the proposed framework need to address the three main objectives. The first objective is to reduce the probability of missing existing critical vulnerabilities on the desired target. The other two features needed are the framework should be able to analyze the risk of existing vulnerabilities (on the target) with respect to weakness in neighboring nodes numerically.

To meet the first desired objective, the proposed framework attempts to utilize vulnerability databases. As a means for identifying “vulnerabilities in a target”, the framework attempts to use data gathered about the platforms from reconnaissance.

To address the other two features, the attack “graph-based risk assessment” model from Aksu was utilized in tandem with CVSS “Attack Vector” threat modeling scheme. Additionally, the “vulnerability Connectivity” by Zhang and CWE relation data were used as a means for building intuition for risk.

Starting with the concept of, “threat source”, the approach employed in this study is Aksus’ approach in tandem with the concepts of “Attack vector” of the CVSS. Since the main objective of this study is to come up with a “systemic vulnerability assessment framework” using the CVSS, forming several path chains among the vulnerabilities is not performed.

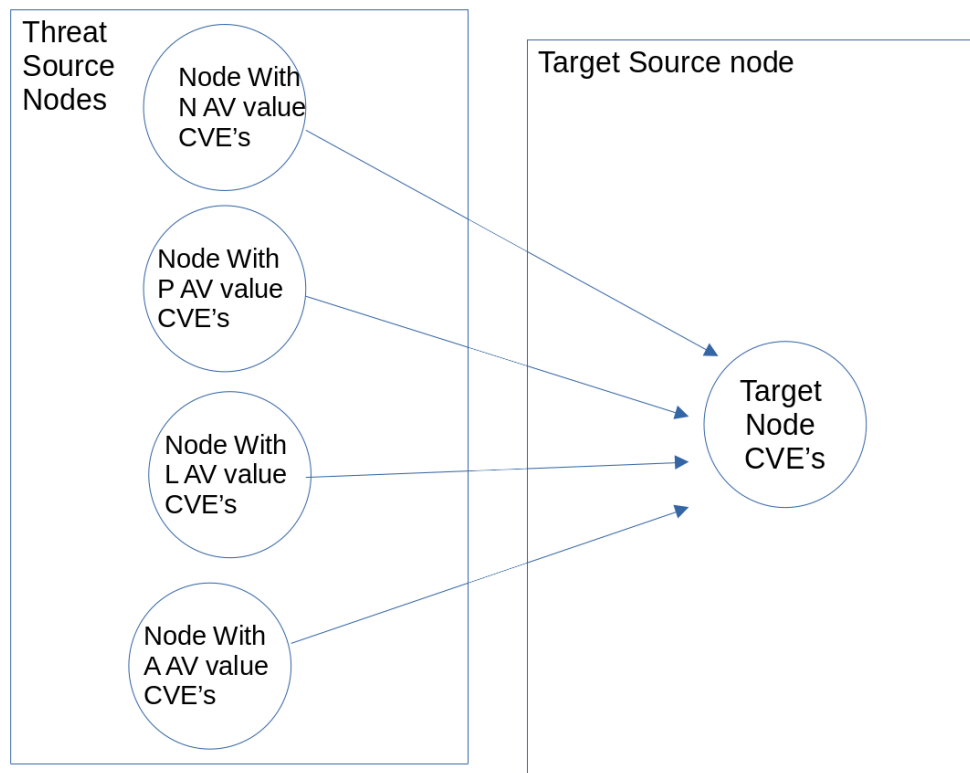


Figure 22 Threat and Target Abstraction based on CVSS Attack vector

The first abstraction to be considered in the coming model is going to use the CVSS “Attack Vector” values. Therefore, using the AV values, even in large-scale networks can abstract their network threat source into four. See the figure above; hence, most assumptions are made with respect to weakness in the target node. It is also possible to have multiple target nodes.

In the model proposed above, the whole network is abstracted to five nodes. The nodes in the threat source are considered devices (vertices) that contain all the possible vulnerabilities found in the network with corresponding attack vector values. Moreover, the list of vulnerabilities contains, CVEs identified in the target, as it is desired to include the possibility of chaining vulnerabilities found within the target. In addition, analyzing the chain formed with neighboring nodes assesses risk with surrounding weaknesses in mind: which is one of the desired features.

The abstraction above also reveals the last CVE weakness that is needed to be used by the attacker, before exploiting a CVE inside the target. It is seen from the model above that, paths followed to get to the CVE weaknesses within the “threat source node box” cannot be captured. From the perspective of protectors, specifically in the context of “penetration testing”, one does not need to know the paths that perpetrators have followed, as the goal is to identify the risk on the target “as quickly as possible and take mitigation measures”. Additionally, identifying the

“vulnerability” that can form most edges is more expedient. This is because several risky paths are formed using a few vulnerabilities.

Following, the abstraction in the above model, the next question that rise is “what criteria do we use to identify possible chaining CVEs to give priority for testing?” and “how can one compare the paths numerically”.

To start with, the first question given the lack of data concerning chaining between vulnerabilities, an assumption is made to help eliminate improbable chaining. Thus, to check if the two CWE IDs of chained weaknesses have some relation with each other from their provided data of “related weakness” and “related patterns”. Therefore, priority must be given to validating these paths if found in the CAPEC “related patterns” data.

Additionally, another assumption to be used giving priority using the “Attack Vector” value of transitions from threat to target. So, the assumptions taken can be summarized in the table below. It should be noted that this proposed logic below originated based on the core definition of the vectors and analysis of vulnerability data.

Table 8 Summary of Threat source to Target "Attack Vector" Chaining logic matrix

Threat Node AV vector Value	Target Node AV vector Value	Required	Justification
Network	Physical	No	-2% of total weakness account to Adjacent -by definition physical weakness can only be a beginning of an attack path - about 23 percent of vulnerabilities have an AV score value of L, of which the majority have of which their score is high or medium - the link between adjacent to adjacent is omitted because there is less than a 2% chance of omitting weakness above ‘medium’ CVSS score -no direct edge can be formed to weaknesses in other nodes unless using other weaknesses in the same node.
Network	Network	Yes	
Network	Adjacent	Yes	
Network	Local	Yes	
Local	Physical	No	
Local	Network	Yes	
Local	Adjacent	Yes	
Local	Local	Yes	
Adjacent	Physical	No	
Adjacent	Network	Yes	
Adjacent	Adjacent	Yes	
Adjacent	Local	Yes	
Physical	Physical	No	
Physical	Network	No	
Physical	Adjacent	No	
Physical	Local	No	

Lastly, to answer the question “how can one compare the paths”; the CVSS Exploitability and base score are going to be used to quantify. So, the probabilistic weight of the edge connecting two vertices is the “vulnerability connectivity” operator as defined by Zhang [9]. Furthermore, the CWE “relation count” and “average chained score” can shade the likelihood of edge formation based on the CWE id combination.

The “vulnerability connectivity” operator as defined by Heng Wei Zhang is the probability a perpetrator will use a vulnerability in an adjacent node to launch a successful attack [9]. Their work mainly utilized CVSS version 2[9]. Their main argument is, vulnerability connectivity is related to “information accessibility”, “host connectivity” and “availability”. These traits are analogous to CVSS version 2 vectors AV (Access vector), Au (Authentication user, which is replaced by PR in version 3), and AC (Attack Complexity) respectively. The formula used by Zhang is:

$$\frac{AU}{AU + \sqrt{AC \times AV}}$$

Formula 1 Zhang Vulnerability Connectivity

Therefore, since in this study CVSS version 3 is being utilized, the AV vector representing host connectivity is going to be replaced by PR. Thus, changing the formula to

$$\frac{PR}{PR + \sqrt{AC \times AV}}$$

Formula 2 Modified version 3 Vulnerability connectivity

Hence, to answer the question “how can one compare the paths”; the CVSS Exploitability and base score are going to be used to quantify. Therefore, the probabilistic weight of the edge connecting two vertices is the multiplication of the “vulnerability connectivity” of each node as defined by Heng Wei Zhang. The figure attached below can be the summary of the process followed.

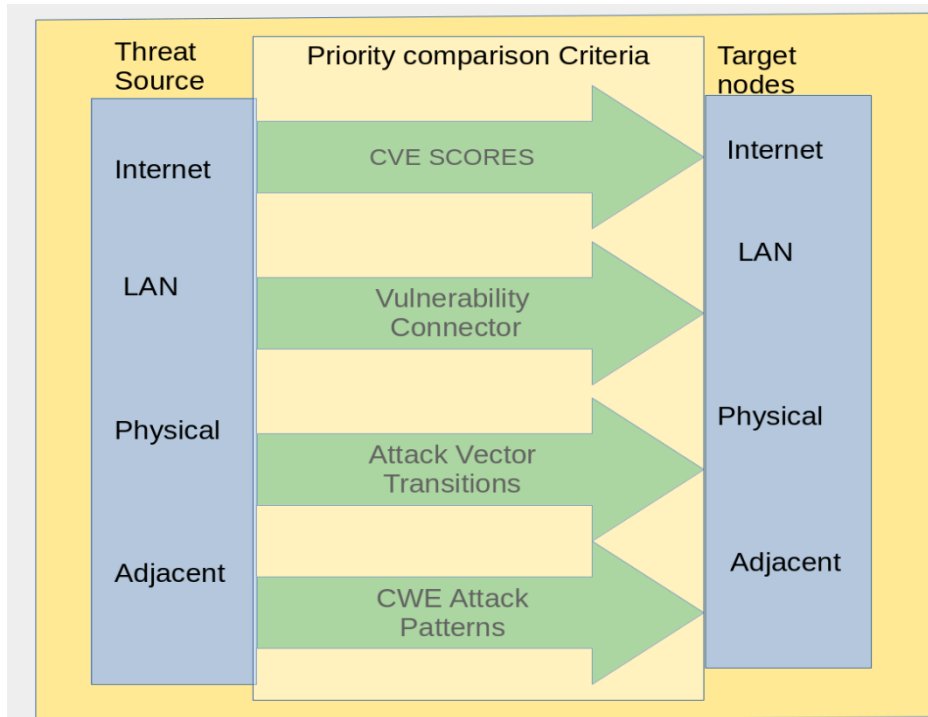


Figure 23 Aspects and Comparison considered

Finally, the focus of this study was to identify the first possible path that can be formed with a vulnerability in the “target node” from the “threat source”. The main assumption here is, if the analysts can identify the first possible chain that can be formed with a weakness in the target node, then they can use that to take mitigation measures.

Additionally, data provided by CWE related “patterns” and “weaknesses” are used to shed light on the likelihood of edge formation between the weaknesses. Finally, the table below shows the metric values of the CVSS version 3 vectors used for calculating the “vulnerability connectivity” operator as per the specification of CVSS 3.1.

Table 9 CVSS version 3 Vectors metric value as per the specification

Attack Vector				Attack Complexity	
N(Network)	A(LAN)	L(Local)	P(physical)	H(High)	L(Low)
0.85	0.65	0.55	0.22	0.77	0.44
Privileged Required					
Scope Changed(“C”)			Scope Unchanged(“U”)		
None(H)	Low(L)	High(H)	None(H)	Low(L)	High(H)
0.85	0.68	0.5	0.85	0.62	0.27

5.1.4 Proposed of Framework

The first phase generally consists of gathering data about the target from an open-source database. The second phase consists of identifying possible weaknesses, vulnerability chaining, calculating the probability of chaining and CVSS score. Finally, based on the probability (vulnerability connectivity), CVSS chained score, and related weakness, presenting the vulnerabilities with CVE-ID. The proposed framework for vulnerability assessment for penetration testing is as in the fig below.

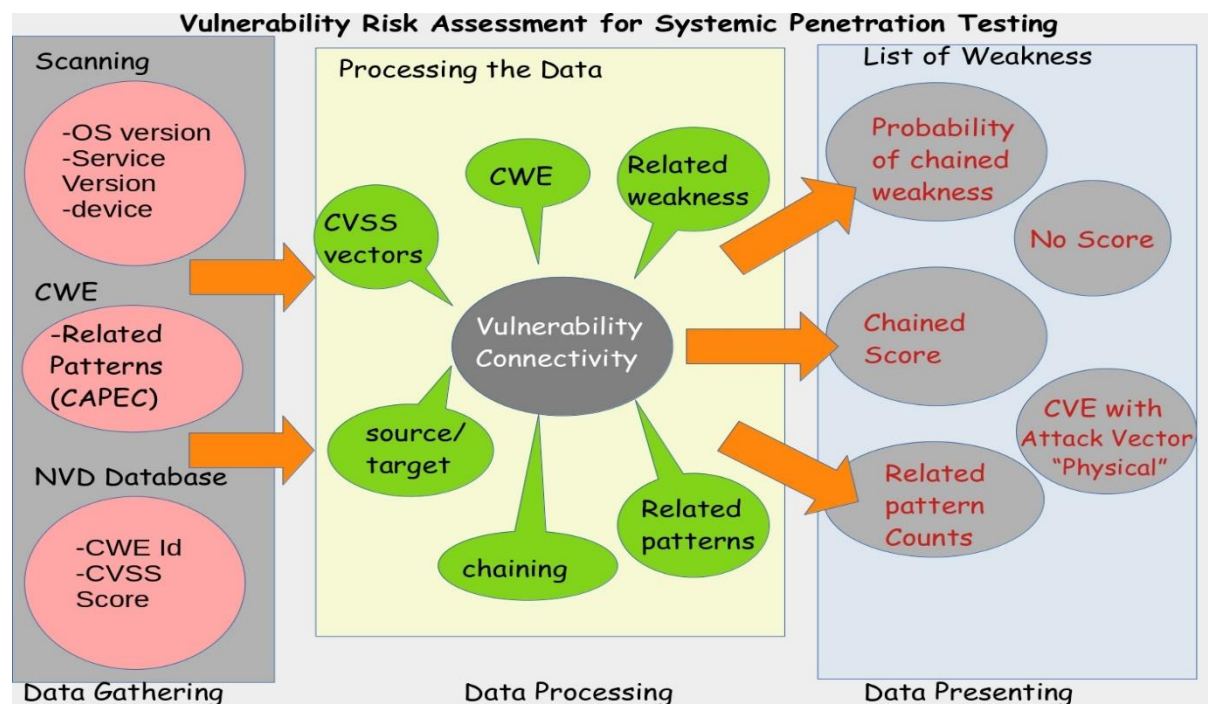


Figure 24 Vulnerability risk Assessment framework for Penetration testing

5.2 Discussion

This study proposed a “systemic vulnerability assessment framework” for penetration testing, which has three basic components. The first components mainly focused on data gathering. The second component is focused on processing gathered data. Finally, the processed output data is presented. The discussion of each component is presented in the coming sections below.

5.2.1 Data Gathering

The data gathering component of the framework is mainly focused on gathering data from different sources. Correspondingly, three main types of data are needed, which are “vulnerability list from open-source databases”, “data about most common attack patterns” and “reconnaissance data” from scanning the “system of interest”.

The data from open-source vulnerability databases need can contain different information about the recorded CVE. The main reason for gathering data from different vulnerability databases is to improve coverage of considered vulnerabilities that cannot simply be assessed with the use of a combination of “vulnerability assessment tools”. Additionally, the CVSS scores and CWE id can help perform numeric analysis.

The Other data about “related attack patterns” about vulnerabilities can help build an intuition about edges formed based on their CWE id. These data can help testers/defenders identify a vulnerability that is found in the most known “attack patterns”.

Finally, the main purpose of data from reconnaissance’s to identify the type of platforms(software/hardware) running in the nodes of the “system of interest”. Thus the “output of reconnaissance” (OS detection, service detection, etc. ...) is used to filter out possible existing vulnerabilities gathered by open-source vulnerability databases.,

5.2.2 Processing Data

The “Data processing” component of the framework mainly attempts to summarize the processing using three numeric metrics. This component consumes data from “Data gathering”. The main numerical calculation made are “average connectivity”, “average CVSS chained Score” and “Relation count”.

The “average Connectivity” is the mean of “vulnerability connectivity”. The statistical mean is calculated for each CVE using edges formed with vulnerabilities in the “threat source” (see the abstraction figure 21). This value measures the likelihood of path formation of a given CVE with respect to neighboring vulnerabilities. The “average CVSS score” is calculated similarly, but utilizes the CVSS vulnerability chaining recommendation. The average value of a given CVE “CVSS score” reflects how much it would affect the severity of an “attack path”.

Finally, the “relation count” is a frequency measure. It counts the number of times the CVEs weakness enumeration id was found in an identified “Attack Pattern” (CAPEC). Consequently, the higher the value, the more likely the CVE is part of a more severe attack path.

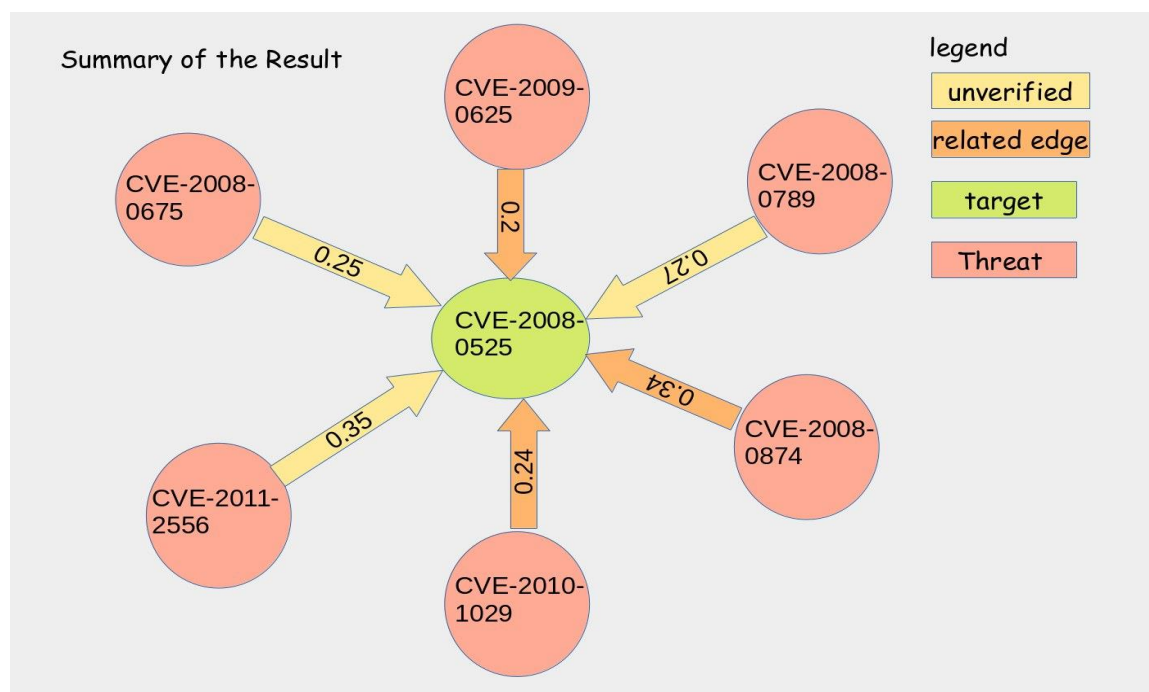


Figure 25 Data processing metrics Calculation visulaization

To sum up, consider figure 24 above which shows how the metrics described above are calculated for a single CVE in a target node. The green node represents a CVE in a target node, while the dark pink nodes represent CVEs in neighboring nodes (threat source). The dark orange edges are to delineate the existence of the “related pattern” in CAPEC mitre.org data. Thus, both the “average connectivity” and “average CVSS score” are calculated as the mean of edges formed as in the figure above. The “relation count” value is the number of dark orange edges formed by the target CVE.

5.2.3 Data Presentation

The data presentation component normally presents the processed result for each CVE in the target node. The results can be grouped into three, which are:

- CVEs in the target with attack Vector Value “Physical” along with CVSS severity score if it exists. Note that these CVEs cannot be chained with any neighboring weaknesses as the “attack Vector” value by definition requires the attacker to have physical access to successfully use this weakness.
- CVEs in the target without a score (which is beyond the scope of this study due to time constraints)
- CVEs with a score along with “average connectivity,” average CVSS score” and “relation count” metrics value.

Thus, with the result in mind, penetration testers can move on to the exploitation and reporting stage of penetration testing. One advantage of this result is it gives penetration testers a chance to numerically compare the severity of possibly existing vulnerabilities found in the “system of interest”. Additionally, unlike the traditional penetration testing process, it gives testers the chance to choose “automated testing tools” based on existing CVEs.

CHAPTER SIX

DEMONSTRATION AND EVALUATION

This chapter discusses the demonstration and evaluation of the proposed framework. The demonstration section uses synthetic data gathered from different networks in Addis Ababa. The Evaluation of the proposed framework is logically discussed from the perspective of desired features of the framework.

6.1 Demonstration

In the “Design Science Research” process, the demonstration stage involves some kind of experimentation to demonstrate how the proposed framework can be utilized. Thus, concerning this study, the following sub-sections discuss the synthetic data for demonstration and how it can be processed using the proposed framework.

6.1.1 Data for Demonstration

The table below is a summary of synthetic data gathered from different networks in Addis. This data is gathered using Nmap scanner, using Black box penetration testing reconnaissance methodology. Additionally, the summarized result presented here is the summary of the platforms identified and IP addresses. These results are used to identify a possible vulnerability that could exist in scanned devices.

Table 10 Summary of scanned data used for demonstration

Nodes Considered Threat source in a LAN	
10.9.30.1/10.1.0.89/10.1.0.22/10.1.41.14(group of IP addresses)	
platforms	-Cisco 2900-Series -IOS 12.1 -Cisco IOS 12.x -Cisco 3700-Series
10.90.10.29/10.90.10.28(group of IP addresses)	
platform	-Ubuntu -OpenSSH 7292
10.6.1.1(group of IP addresses)	
Platform	-Fortinet -OpenBSD 4.X
10.133.193.243/1.1.1.2(group of IP addresses)	
Platforms	-unidentified
Node Selected as Target	
10.90.10.76	
platforms	-OpenSSH -OpenSSH 6.7p1 -Apache httpd 2.4.10 -Linux 3.x -Linux 4.x -Debian

The other data used provided by mitre.org is “CWE compressive View” data which is not presented here due to the size of the data. This data can be found at

<https://cwe.mitre.org/data/downloads.html>. Additionally, the raw output of scanned synthetic reconnaissance data can be found in Appendix B.

6.1.2 Processing Data and Flow Chart

The processing stage of data gathered mainly includes the calculation of “average connectivity”, “average CVSS score” and “relation count”. In Addition, a list of CVE ids without CVSS score and “attack vector” value “physical” are filtered. The figure shown below can aid in visualizing the processing stage of the proposed framework.

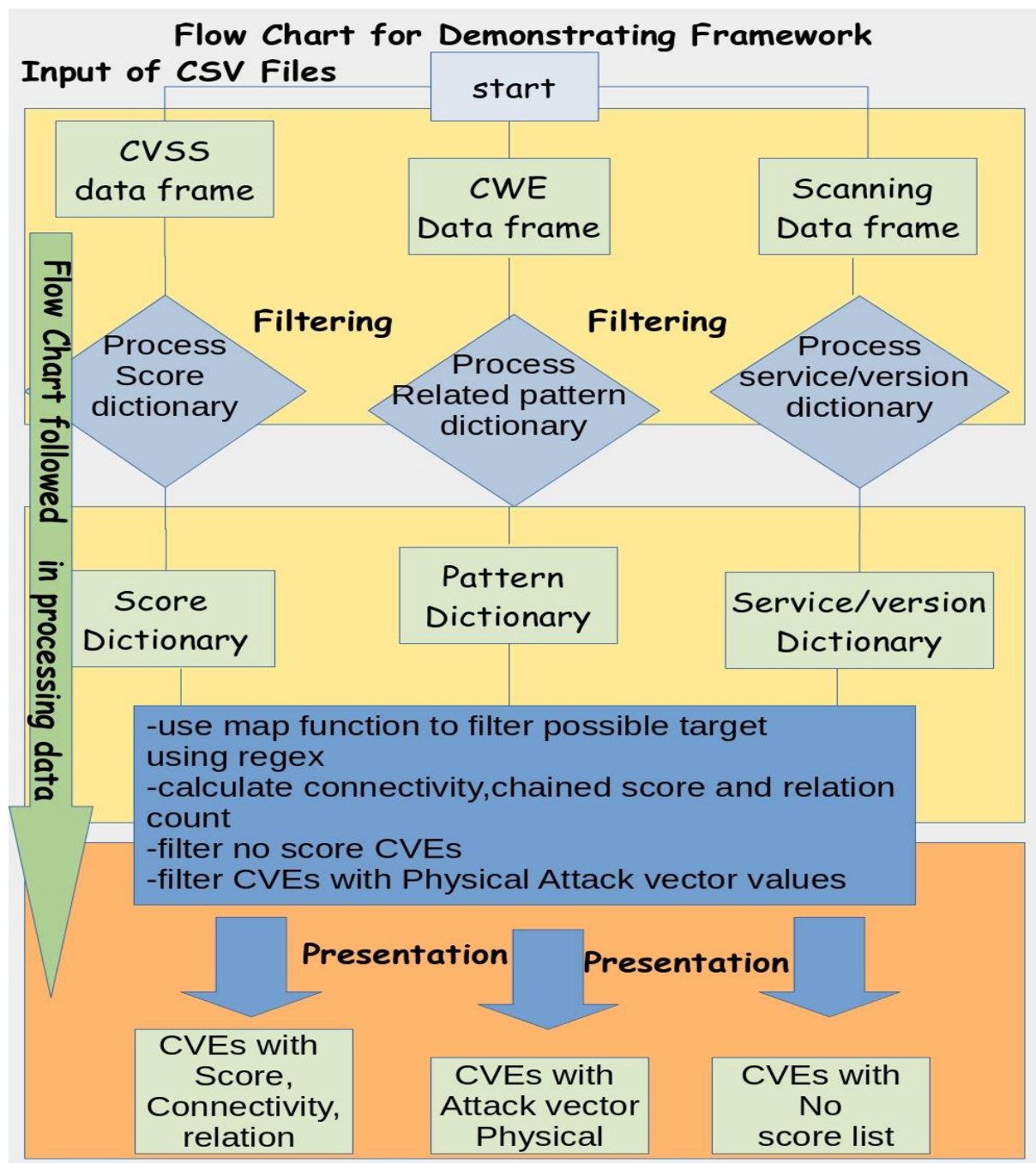


Figure 26 Flowchart summarizing the data processing in the proposed framework.

As it can be seen from the flow chart, the whole process is going to utilize data from three different sources. The identified platforms are used to identify CVEs in “threat” and “target” nodes. Following, “average connectivity”, “average CVSS score” and “relation count” are calculated. The following tables/figure below demonstrates how these values are calculated using five CVEs once the weaknesses are identified using “platforms”. Two of the CVEs are from the “threat” node while three are CVEs in the target.

Table 11 CVES used for Demonstration

CVEs with score and metric values to demonstrate the calculation					
CVE_id	CVSS Vector	CVSS Score	CWE -Id	Connectivity	location
CVE-2001-0361	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:	4.8	310	0.51	Target
CVE-2016-4484	CVSS:3.0/AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	6.8	287	N/A	Target
CVE-1999-0373	No Score	N/A	noid	N/A	Target
CVE-2002-2239	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	7.5	20	0.58	Threat
CVE-2005-0197	CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	6.5	16	0.61	Threat

The table below is a summary of edges formed with one of the CVE’s

Table 12 Summary of Edges formed based on Table 11

Calculations for CVE id CVE-2001-0361 from above table				
Edges	Chained Vector	Chained Scores	Chained Connectivity	Relation in attack pattern based on CWE id(yes/No)
Edge 1(with CVE-2002-2239)	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/C:L/A:H	8.6	0.29	Yes
Edge 2(with CVE-2005-0197)	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/C:L/A:H	8.6	0.311	No
	Average Score	8.6		
	Average connectivity		0.3	
	Relation count			1

The figure provided below further illustrates the details of the edges.

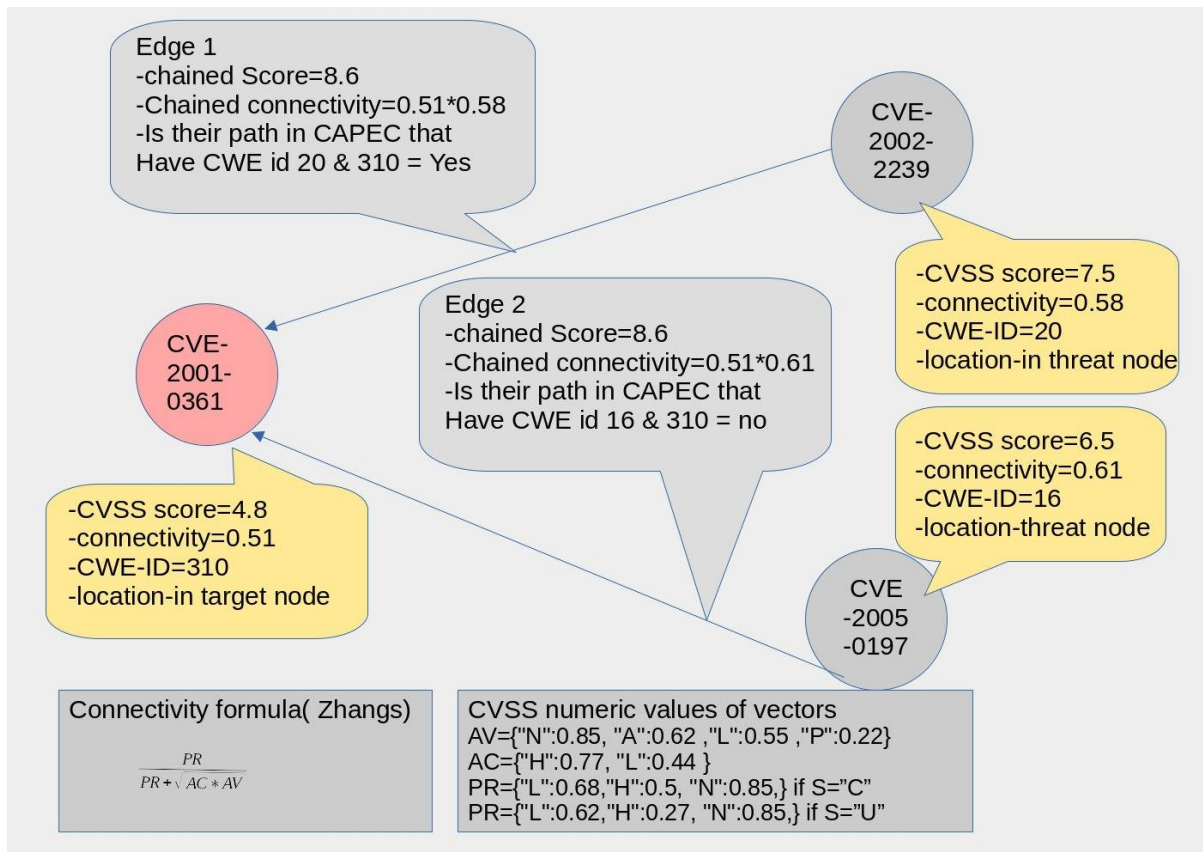


Figure 27 Illustration of Table 12 results

In table 11, CVEs not utilized are inserted to show edge cases. The CVE with Attack Vector value is not chained together with any neighboring node, because by definition an attacker needs physical access to the target node to use that weakness. Additionally, the other edge case, CVE with no score, is shown as is it as there is no means of comparing it with other CVEs based on the vectors.

6.1.3 Presentation

The final output of the proposed result was processed using python script and presented in form of a table. The sample result of the table is presented in Appendix C due to the size of the table. Additionally, the complete result of the tables, codes can be found at <https://github.com/bushubeke/systemic-vulnerability-risk-assessment>.

6.2 Evaluation

The Evaluation of the proposed framework is argued logically concerning the three main objectives of the proposed framework. The three main objectives of the framework are “minimizing missing of existing vulnerabilities”, “adding a feature in the vulnerability assessment process that quantifies risk” and improving the “vulnerability assessment process” in terms of time and cost.

To start with the first objective, the main goal was to minimize the chance of not discovering existing vulnerabilities. This problem is usually imposed due to reliance on “vulnerability assessment tools”. Consequently, in the proposed framework, to solve this predicament, it was

proposed to use “open-source vulnerability databases” in tandem with “output of reconnaissance”. In this proposed solution, it is expected to increase the number of vulnerabilities to be discovered, mainly because the approach goes straight to the “source of vulnerabilities” that most of these “vulnerability assessment tools” utilize.

The second and third desired features of the framework are, the “ability to numerically compare vulnerabilities” with respect to neighboring nodes and “making the graph-based attack path risk assessment process” more manageable for penetration testing. Additionally, improving the performance of the penetration testing process in terms of time and cost is included in the third objective. To enable “vulnerability assessment in penetration testing” analyze the risk of a vulnerability with respect to neighboring weaknesses, incorporating graph-based risk assessment was necessary. This does enable testers to capture the risk of weakness with respect to neighboring nodes. Additionally, the proposed numeric values can indirectly help defenders gauge the likelihood of a “vulnerability” being used together with neighboring weaknesses.

Finally, “path-based” assessment needed to be improved is it generates a loads number of paths just between a few vulnerabilities. For instance, in the demonstration based on the “platforms”, 281 vulnerabilities were identified in the target. In addition, 31 weaknesses were found in 3 different nodes and 187 in one. Now, if we can consider the number of possible paths that can be formed using the above-described nodes (5 nodes); the permutation using the vulnerabilities yields 1,565,427,677 possible paths. This is without considering cyclic paths and is formed using only 499 identified CVEs. The same path formation using the abstraction by the framework needs to analyze only 61,258. This does mean less process, thus consuming fewer resources and less time to analyze.

Additionally, it enables penetration testers to choose “vulnerability assessment and testing” tools” based on the output of these results. This means the purchase of tools that might not be relevant for testing might be avoided. Additionally, numeric results would enable testers to invest(time/resource) in weaknesses that have high severity impact, which is not employed in traditional testing processes.

6.3 Discussion

The results from the demonstration and evaluation of the framework can be seen in the light of context coverage, effectiveness, and efficiency.

In terms of context coverage, the proposed framework brings an additional dimension to “vulnerability assessments” conducted in traditional penetration testing. The added dimension brought by the framework is a graph-based analysis of weaknesses. The approach taken gives testers a means to visualize the relationship between weakness within the networked devices of systems of interest.

Additionally, it gauges information availability and the possible impact of weakness before exploitation indirectly using a standardized scoring system. It has a huge advantage, unlike the traditional penetration testing process, because there is no standardized way of comparing weaknesses in the testing process. Thus, penetration testers normally are accustomed to identifying weaknesses, verifying the existence of the weakness, and reporting what they were able to do with weaknesses. This approach, while it can help in improving security, it is not a systematic approach; as testers could be investing resources by analyzing vulnerabilities

that are not as damaging to the system of interest. Therefore, having awareness about information availability and impact does add quality to the assessment process.

Furthermore, while the proposed framework adds overhead to the traditional penetration testing process, it does improve test coverage and attempts to minimize missing “critical vulnerabilities” by using comprehensive opens source vulnerability databases. This aspect of the framework comes with a drawback, as it means considering more weaknesses.

In terms of effectiveness, the proposed graph-based vulnerability assessment from the result of the demonstration and mathematical analysis of the CVSS scoring: the fact that the scores get skewed to the right as the number of chains increase(see figure 19 &20) show that the frameworks approach is more effective unlike fully connected analysis of attack paths. It should be noted that this does not mean it guarantees absolute security if employed in the testing process.

Even more, the output of the framework uses numerical values to measure information availability and impact. This gives testers an advantage where time and resources are constraints, as it gives testers a chance to prioritize based on the numerical outputs.

Traditionally, you might argue what is the point of performing the analysis if there is a chance of leaving weaknesses during the verification stage. The answer to that can be taking a risk having some level of awareness about the possible damage is much better than taking a risk without having no awareness about the possible damage.

Finally in terms of efficiency, while there is a clear indication that some aspects of the “penetration testing vulnerability assessment process” are improved by the framework; further research needs to be done. Further works on “how utilizing the framework improved the efficiency of security” are needed. In addition, feedback from users and user satisfaction could better the framework, as this can be considered as the first of many iterations to improve the vulnerability assessment process in penetration testing.

CHAPTER SEVEN

CONCLUSION AND RECOMMENDATION

This chapter discusses the conclusion and findings of this study based on the demonstration. Moreover, based on the finding, further study is presented.

7.1 Conclusion

The main goal of this study was to develop a “systemic vulnerability assessment framework” for penetration testing. The main problems it attempts to solve were:

- The rising number of vulnerabilities discovered. This has made depending on a few automated tools not as effective as desired due to missing vulnerabilities.
- Traditional penetration testing “vulnerability assessment process” does not analyze the effect of a vulnerability with respect to neighboring weaknesses.
- Traditional penetration testing vulnerability assessment does not compare the risk of vulnerabilities numerically.

Therefore, with the above problems in mind, the framework proposed using open-source vulnerability databases to identify possible existing vulnerabilities during vulnerability scanning. Hence, not relying on “vulnerability assessment tools” as in traditional penetration testing methodology to identify weaknesses. Additionally, it proposed using graph-based vulnerability-centric risk assessment to enable testers to measure the effect of weakness when being utilized together with CVSS.

The approach taken in the framework was tested using synthetic data. The nature of synthetic data is real-time OS and Service detection scanning results gathered from different networks in Addis Ababa. This data can be taken as a sample of real-time networks, as the output generated by scanning real-time single networks are similar. The reason for labeled synthetic data here is to indicate the devices scanned do not belong to any single organization's network.

Furthermore, findings after using synthetic data results reflect the following main points:

- the plan of only identifying only the last attack path that can be formed during chaining has significantly reduced the number of paths that need to be handled by generating all possible paths. As described in the evaluation, only 61,258 need to be analyzed as compared to evaluating paths formed using several nodes.
- The results enable penetration testers to choose their testing tools based on the output. This is an advantage over starting with vulnerability scanning tools which can only consider a few CVEs. For instance, it was mentioned earlier Metasploit has exploits that account for less than 2% of known vulnerabilities.
- The “relation count” metric along with “average connectivity” helps to indirectly gauge the likely hood of a vulnerability being used in an attack. This is because these values gauge the availability of information about the particular weakness. Thus, the higher the values the more likely an attacker will find an “automated tool” or information (how to use it to exploit) about the weakness.
- The “vulnerability connectivity” operator along with the “chained score” can help in reducing unnecessary “resource allocation” (by prioritizing).

- The data-gathering(scanning) stage of the framework results consisted of some false positive service and Os detection outputs. This means, there is a greater chance of wasting resources by running non-existent weaknesses.
- Performing exploitation was not part of this study. This means how much effective the framework was in weeding out false-positive vulnerabilities during the identification process is still unknown.

7.2 Recommendation

The main goal of this study was to improve introduce a systematic approach to the vulnerability assessment of penetration testing. Considering the process and the output of the demonstration, the following are the main recommendations:

- Further studies can be conducted with regards to improving how weakness without a score can be handled. It should be noted that weaknesses without scores are likely vulnerabilities with less information about them on the internet. This means they are less likely to be utilized by penetration testers who utilize automation tools, but that does not mean possible impact by them is not critical. Therefore, in the process of scoring this weakness, attempting to quantify the possible impact is more profitable.
- The Scope of this study was bound to published vulnerabilities. This means it did not consider zero-day vulnerabilities which could still have an unpredictable impact and information availability. Although it is a known fact that handling zero-day vulnerability is a challenge, a pragmatic approach that can consider such weaknesses could greatly improve the penetration testing process and the proposed framework.
- Due to the challenging nature of finding(gathering) data about relations, a more pragmatic model of weeding out false positives could improve the penetration testing process.
- The CVSS which is dynamically maintained by first.org continually. Therefore, upgrading the model by using CVSS vector recommendations could add insight for further improvement of the framework.
- Graph-based assessments by the framework did not attempt to analyze and generate all possible attack paths. This is because analyzing complete path formation for penetration testing is not practical. Therefore, another practical approach that can utilize complete path formation and verification can be one direction to explore as well.
- The research conducted utilized CVSS which was accepted as standard recently. This means the data that is gathered normally are not structured. Therefore, preparing structured and standardized datasets for experimentation and exploration can help ease the difficulties faced by researchers in their plight to improve security.
- Due to time and resource constraints, performing the exploitation stage was not part of the evaluation process. Thus, performing exploitation using the proposed framework can help measure how efficient it is as compared to the traditional testing process.
- Finally, this study purely relied on the Base score of the CVSS to perform the analysis, mainly because the experiments resemble “Black Box penetration testing”. Therefore, it is recommended to use the environmental metrics to get a more realistic picture of the system. This means, using the framework in tandem with a “Gray Box Penetration

testing methodology” can improve the performance of the framework in terms of time, cost and effectiveness as well.

REFERENCE

- [1] Moore, Tyler, and Ross Anderson. 2011. "Economics and Internet Security : A Survey of Recent Analytical , Empirical and Behavioral Research." ." Accessed December 5, 2020, <https://dash.harvard.edu/handle/1/23574266>.
- [2] Pirounias, Sotirios, Dimitrios Mermigas, and Constantinos Patsakis. 2014. "The Relation between Information Security Events and Firm Market Value, Empirical Evidence on Recent Disclosures: An Extension of the GLZ Study." *Journal of Information Security and Applications* 19 (4–5): 257–71.
- [3] Leaders, Project, Matteo Meucci, and Andrew Muller. n.d. "OWASP Testing Guide Version 4," ." Accessed December 7, 2020, <https://owasp.org/www-pdf-archive/OTGv4.pdf>
- [4] Schneier, Bruce. n.d. "Full Disclosure and the Window of Exposure." Accessed December 11, 2020. <https://www.schneier.com/cryptogram/archives/2000/0915.html>.
- [5] NVD. n.d. "Vulnerability Discovery Statistics." Accessed December 11, 2020. https://nvd.nist.gov/vuln/search/statistics?form_type=Basic&results_type=statistics&search_type=all.
- [6] Goel, Jai Narayan, Mohsen Hallaj Asghar, Vivek Kumar, and Sudhir Kumar Pandey. 2016. "Ensemble Based Approach to Increase Vulnerability Assessment and Penetration Testing Accuracy." 2016 1st International Conference on Innovation and Challenges in Cyber Security, ICICCS 2016, no. Iccics: 330–35. <https://doi.org/10.1109/ICICCS.2016.7542303>
- [7] Shinde, Prashant S., and Shrikant B. Ardhapurkar. 2016. "Cyber Security Analysis Using Vulnerability Assessment and Penetration Testing." *IEEE WCTFTR 2016 - Proceedings of 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare*. <https://doi.org/10.1109/STARTUP.2016.7583912>
- [8] Aksu, M. Ugur, M. Hadi Dilek, E. Islam Tatli, Kemal Bicakci, H. Ibrahim Dirik, M. Umut Demirezen, and Tayfun Aykir. 2017. "A Quantitative CVSS-Based Cyber Security Risk Assessment Methodology for IT Systems." *Proceedings - International Carnahan Conference on Security Technology* 2017-Octob: 1–8. <https://doi.org/10.1109/CCST.2017.8167819>.
- [9] Zhang, Heng Wei, and Jian Ming Huang. 2017. "System Vulnerability Risk Evaluation Using Connectivity Operator." 2016 2nd IEEE International Conference on Computer and Communications, ICC 2016 - Proceedings 395 (A V): 979–84. <https://doi.org/10.1109/CompComm.2016.7924851>.
- [10] Bogale, Milkyas. 2018. "PROPOSING INFORMATION SECURITY AWARENESS PROGRAM FOR ENAT BANK IN ETHIOPIA." Thesis, Addis Ababa University.
- [11] Ransbotham, Sam. 2010. "An Empirical Analysis of Exploitation Attempts Based on Vulnerabilities in Open Source Software." *Weis*, no. June: 1–25. Accessed December 21, 2020, http://weis2010.econinfosec.org/papers/session6/weis2010_ransbotham.pdf.
- [12] Karen Scarfone, Murugiah Souppaya, Amanda Cody, and Angela Orebaugh. 2008. "Technical Guide to Information Security Testing and Assessment Special

- Publication 800-115.” Nist Special Publication 800: 1–80. Accessed December 11, 2020 <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-30r1.pdf>
- [13] Team, The Ptes. 2021. “The Penetration Testing Execution Standard Documentation,” Accessed January 18, 2021 http://www.pentest-standard.org/index.php/Main_Page.
- [14] Murthy, P. V.R., and R. G. Shilpa. 2018. “Vulnerability Coverage Criteria for Security Testing of Web Applications *.” 2018 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2018, 489–94. <https://doi.org/10.1109/ICACCI.2018.8554656>.
- [15] Vats, Prashant, Manju Mandot, and Anjana Gosain. 2020. “A Comprehensive Literature Review of Penetration Testing Its Applications.” ICRITO 2020 - IEEE 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions), 674–80. <https://doi.org/10.1109/ICRITO48877.2020.9197961>.
- [16] Goel, Jai Narayan, and B. M. Mehtre. 2015. “Vulnerability Assessment & Penetration Testing as a Cyber Defence Technology.” Procedia Computer Science 57: 710–15. <https://doi.org/10.1016/j.procs.2015.07.458>.
- [17] NIST. 2018. “Risk Management Framework for Information Systems and Organizations.” NIST Special Publication - 800 Series, 183. Accessed January 28, 2021 ,<https://csrc.nist.gov/publications/detail/sp/800-37/rev-2/final%0Ahttps://doi.org/10.6028/NIST.SP.800-37r2%0Ahttps://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r2.pdf>.
- [18] P. Mell, K. Scarfone and S. Romanosky, "Common Vulnerability Scoring System," in *IEEE Security & Privacy*, vol. 4, no. 6, pp. 85-89, Nov.-Dec. 2006, doi: 10.1109/MSP.2006.145.
- [19] Base, The, and The Base. 2019. “CVSS v3.1 User Guide,” 1–22. Accessed November 18, 2020 <https://www.first.org/cvss/>.
- [20] FIRST. 2019. “Common Vulnerability Scoring System Version 3.1 Specification Accessed November 18, 2020 Document Revision 1,” 1–24. <https://www.first.org/cvss/>.
- [21] ITU-T. 2015. “Series X: Data Networks, Open System Communications and Security. Common Weakness Scoring System.” Accessed December 02, 2020, https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.1525-201504-I!!PDF-E&type=items
- [22] Christey, Steve(MITRE). n.d. “Common Weakness Scoring System (CWSS).” Accessed June 14, 2021. https://cwe.mitre.org/cwss/cwss_v1.0.1.html.
- [23] MITRE, Corporation. 2018. “CWE Version 3.1.” Cwe, 1822, Accessed February 18, 2021, <https://cwe.mitre.org/>
- [24] G. Chu and A. Lisitsa, "Poster: Agent-based (BDI) modeling for automation of penetration testing," *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, 2018, pp. 1-2, doi: 10.1109/PST.2018.8514211.
- [25] Metcalf, Leigh, and William Casey. 2016. Cybersecurity and Applied Mathematics. Cybersecurity and Applied Mathematics. Accessed June 14, 2021,

<https://doi.org/10.1016/C2015-0-01807-X>

- [26] Lallie, Harjinder Singh, Kurt DeBattista, and Jay Bal. 2020. "A Review of Attack Graph and Attack Tree Visual Syntax in Cyber Security." *Computer Science Review* 35: 100219. <https://doi.org/10.1016/j.cosrev.2019.100219>.
- [27] Alhomidi, Mohammed, and Martin Reed. 2013. "Risk Assessment and Analysis through Population-Based Attack Graph Modelling." 2013 World Congress on Internet Security, WorldCIS 2013, 19–24. <https://doi.org/10.1109/WorldCIS.2013.6751011>.
- [28] Rebecca M. Blank. Patrick D. Gallagher. 2012. "NIST Special Publication 800-30 Revision 1 - Guide for Conducting Risk Assessments." NIST Special Publication, no. September: 95.
- [29] Alan Hevner, Salvatore March, Jinsoo Park, Sudha Ram. 2004. "Design Science Research in Information Systems." *Proceedings of the International Conference on Trends in Electronics and Informatics, ICOEI 2004*.
- [30] Gengler, Charles E, Matti Rossi, Wendy Hui, and Johanna Bragge. 2006. "The Design Science Research Process : A Model for Producing and Presenting Information System Research." In *Proceedings of the First International Conference on Design Science Research in Information Systems and Technology*, 83–106.
- [31] Jackson, Emerson Abraham. 2017. "Media Review: Practical Research Methods: A User-Friendly Guide to Mastering Research Techniques and Projects." *Journal of Mixed Methods Research* 11 (3): 417–18. <https://doi.org/10.1177/1558689816668103>.
- [32] Team, PTES. 2020. "PTES Technical Guidelines," 1–176. Accessed June 20, 2021, <https://buildmedia.readthedocs.org/media/pdf/pentest-standard/latest/pentest-standard.p>
- [33] Base, The, and The Base. 2019. "CVSS v3.1 User Guide," 1–22. Accessed November 18, 2020 <https://www.first.org/cvss/>.

APPENDIX A

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from cvss import CVSS3
from itertools import product
import math
from functools import reduce
import operator
import re

# # creating NVD Database data, scanning data dataframe

# In[2]:

nvddata = pd.read_csv('/home/Bushu/Documents/Enviroment/allen/data/nvdwithmissng
.csv', header="infer")
#nvddata[nvddata.cvss_scorev3 != 'nov3']
nvddata = nvddata.iloc[:, [0, 1, 3, 6, 7,]]
targetdata = pd.read_csv('/home/Bushu/Documents/Enviroment/allen/data/mainscan/t
argetresult.csv')
threatdata = pd.read_csv('/home/Bushu/Documents/Enviroment/allen/data/mainscan/t
hreatresult.csv')
cwe_patterns = pd.read_csv('/home/Bushu/Documents/Enviroment/allen/data/cwelist.
csv')
#                                     data = pd.read_csv('/home/Bushu/Documents/Final
Paper/data/nvd2021e.csv', header="infer")
#https://www.cvedetails.com/vulnerability-list/year-2007/month-1/January.html

# In[3]:

len(nvddata[nvddata.cvss_scorev3 != "nov3"])

# ## Data cleaning lines

# In[4]:

targetdata.fillna(0, inplace=True)
```

```

#
nvddata.cvss_scorev3.replace(to_replace=["CVSS:3.1","CVSS:3.0"],value="nov3",i
nplace=True)
cwe_patterns=cwe_patterns.iloc[:,[0,6,21]]

# In[5]:

#open sour e vulnerablity database data
nvddata.head(10)

# In[6]:

# CWE CAPECA data
cwe_patterns.head(10)

# In[7]:

# reconaissance synthetic data
threatdata.head(10)

# In[8]:

targetdata.head(10)

#

# # Probablity calculation formula from ("Vulnerblity Connectivity") by Heng
Wei Zhang

# In[9]:

defVulnerablity_connector(vec):
    ifvec != "nov3":
        v3=vec.split('/')
        v3_dic={ x.split(':')[0]:x.split(':')[1] for x in v3 }

        # this values are taken form cvss 3.1 metric score table
        # Attack Vector = N=0.85 A=0.62 L=0.55 P=0.22
        # Attack Complexity = H=0.77 L=0.44

```

```

        # Priviledges Required N=0.85 scope changed(L=0.68,H=0.5) or scope not
changed(L=0.62,H=0.27)
        # scope C is changed and U is not changed

        # math. sqrt(x)
        av_value={"N":0.85, "A":0.62 , "L":0.55 , "P":0.22}
        ac_value={"H":0.77, "L":0.44 }
        pr_value={ "C":{"L":0.68,"H":0.5, "N":0.85,}, "U":{"L":0.62,"H":0.27,
"N":0.85,}}

        probablity=(pr_value[v3_dic["S"]][v3_dic["PR"]]/(pr_value[v3_dic["S"]][
v3_dic["PR"]]+math.sqrt(av_value[v3_dic["AV"]]*ac_value[v3_dic["AC"]])))

        return probablity
        return "unknown"
nvddata["Vulnerability_connector"]=nvddata.cvss_scorev3.map(Vulnerability_conne
ctor)

# # Scanning result filter and miscellaneous functions

# In[10]:

# this is target version,os and device list
#////////////////////////////////////
ver=targetdata.version.unique()
os=targetdata.OSInfo.unique()
os=[x for x in os if x != 0]
dev=targetdata.Device.unique()
dev=[x for x in dev if x != 0]
#////////////////////////////////////
# this is threat version,os and device list
# //////////////////////////////////
th_ver=threatdata.version.unique()
th_os=threatdata.OSInfo.unique()
th_os=[x for x in th_os if x != 0]
th_dev=threatdata.Device.unique()
th_dev=[x for x in th_dev if x != 0]
# //////////////////////////////////
# This is target session filter
# //////////////////////////////////
def test_target_service(description):
    if re.search(f'({"|".join(os)})',description):
        return True
    return False

def test_target_device(description):

```

```

    ifre.search(f'({"|".join(dev)})',description):
        returnTrue
    returnFalse

def test_target_version(description):
    ifre.search(f'({"|".join(ver)})',description):
        returnTrue
    returnFalse
# //////////////////////////////////////
# This is threat source version filter
# //////////////////////////////////////
def test_threat_service(description):
    ifre.search(f'({"|".join(th_os)})',description):
        returnTrue
    returnFalse

def test_threat_device(description):

    ifre.search(f'({"|".join(th_dev)})',description):
        returnTrue
    returnFalse

def test_threat_version(description):
    ifre.search(f'({"|".join(th_ver)})',description):
        returnTrue
    returnFalse
# //////////////////////////////////////
#scoringfunc### Availablity Severity distributionon
# //////////////////////////////////////
def cvs3_sc(vector):
    score=CVSS3(vector)
    return score.scores()[0]
#distribution function
#thiscatagoraizaitn is based on CVSS3 specification table
# //////////////////////////////////////
def compare(score):
    if score == 0:
        return "None"
    elif 0.1<= score <= 3.9:
        return "Low"
    elif 4<= score <=6.9:
        return "Medium"
    elif 7<=score<=8.9:
        return "High"
    elif 9<=score<=10:
        return "Critical"

```

```

# ///////////////////////////////////////////////////

# # Filtering Possible CVEs in target and threat source

# In[11]:

# Running Filter Map functions for target
# ///////////////////////////////////////////////////
# please note that order matters here when executing
nvddata['target_possible_v']=nvddata.description.map(test_target_version)
nvddata['target_possible_s']=nvddata.description.map(test_target_service)
nvddata['target_possible_dev']=nvddata.description.map(test_target_device)

# Running Filter Map functions for threat source
# ///////////////////////////////////////////////////
# please note that order matters here when executing
nvddata['threat_possible_v']=nvddata.description.map(test_threat_version)
nvddata['threat_possible_s']=nvddata.description.map(test_threat_service)
#nvddata['threat_possible_dev']=nvddata.description.map(test_threat_device)

target_cves=nvddata[(nvddata.target_possible_v == True) |
(nvddata.target_possible_s == True) | (nvddata.target_possible_dev == True) ]
threat_cves=nvddata[(nvddata.threat_possible_v == True) |
(nvddata.threat_possible_s == True)]

# # Generating The List

# In[12]:

target_list_with_nov3=target_cves[target_cves.cvss_scorev3 == "nov3"]
target_list_with_score=target_cves[target_cves.cvss_scorev3 != "nov3"]
threat_list_with_nov3=threat_cves[threat_cves.cvss_scorev3 == "nov3"]
threat_list_with_score=threat_cves[threat_cves.cvss_scorev3 != "nov3"]

# In[13]:

len(target_cves)

# In[14]:

```

```

len(threat_cves)

# ### Combination of the lists

# In[15]:

#generated_combination
generated_combination=list(product(threat_list_with_score.cve_number,target_list_with_score.cve_number))
chained_list=pd.DataFrame({'threat_target':generated_combination},columns=['threat_target'])

# In[16]:

print(f"length of target cve is : {len(target_list_with_score.cve_number)}")
print(f"length of threat cve is : {len(threat_list_with_score.cve_number)}")

# In[17]:

# dictionary of connector,score and cwe id
nvd_score_dic=dict(zip(nvddata.cve_number,nvddata.cvss_scorev3))
connector_dic=dict(zip(nvddata['cve_number'],nvddata['Vulnerability_connector']))
cweid_dic=dict(zip(nvddata['cve_number'],nvddata['cwe_number']))

# In[18]:

#calculating combined vulnerability connectivity probability
defcombined_connector(v3):
    returnconnector_dic[v3[0]]*connector_dic[v3[1]]

#connectivity operator as per zhang
defzhang_connectivity_op(th_targ):
    v3t=nvd_score_dic.get(th_targ[0])
    v3targ=nvd_score_dic.get(th_targ[1])
    v3t_dic={ x.split(':')[0]:x.split(':')[1] for x in v3t.split('/') }
    v3targ_dic={ x.split(':')[0]:x.split(':')[1] for x in v3targ.split('/') }
}

av_value={"N":0.85, "A":0.62 , "L":0.55 , "P":0.22}
ac_value={"H":0.77, "L":0.44 }

```

```

pr_value={ "C":{"L":0.68,"H":0.5, "N":0.85,}, "U":{"L":0.62,"H":0.27,
"N":0.85,}}

probability=pr_value[v3t_dic["S"]][v3t_dic["PR"]]/(pr_value[v3t_dic["S"]
][v3t_dic["PR"]]+math.sqrt(ac_value[v3t_dic["AC"]]*av_value[v3targ_dic["AV"]
]))

returnprobability
#probability=(pr_value[v3_dic["S"]][v3_dic["PR"]]/(pr_value[v3_dic["S"]
][v3_dic["PR"]]+math.sqrt(av_value[v3_dic["AV"]]*ac_value[v3_dic["AC"]]))))

#calculatingchained score
defcombined_chained_score(v3):

    ifnvd_score_dic.get(v3[0]):
        threat_lis=nvd_score_dic.get(v3[0]).split('/')
    ifnvd_score_dic.get(v3[1]):
        target_lis=nvd_score_dic.get(v3[1]).split('/')
    ifnvd_score_dic.get(v3[1]) andnvd_score_dic.get(v3[0]):
        threat={x.split(':')[0]:x.split(':')[1] for x inthreat_lis}
        target={x.split(':')[0]:x.split(':')[1] for x intarget_lis}

        # this values are taken form cvss 3.1 metric score table
        # Attack Vector = N=0.85 A=0.62 L=0.55 P=0.22
        # confidentiality/Integerity/Availablity H=0.56,L=0.22,N=0
        # User Interaction N=0.85 R=0.62
        # Attack Complexity = H=0.77 L=0.44
        # Priviledges Required N=0.85 scope changed(L=0.68,H=0.5) or scope
not changed(L=0.62,H=0.27)
        #print(threat)

        scope=threat['S']
        if threat['S'] != target['S']:
            scope="C"

        ui=threat['UI']
        if threat['UI'] != target['UI']:
            ui="N"

        ac=threat['AC']
        if threat['AC'] != target['AC']:
            ac="H"

        if threat['C'] == "H"or target['C'] == "H":
            conf="H"
            elif threat['C'] == "L"or target['C'] == "L"and (threat['C'] !=
"H"or target['C'] != "H"):

```

```

        conf="L"
    else:
        conf="N"

    if threat['I'] == "H"or target['I'] == "H":
        integ="H"
    elif threat['I'] == "L"or target['I'] == "L"and (threat['I'] !=
"H"or target['I'] != "H"):
        integ="L"
    else:
        integ="N"

    if threat['A'] == "H"or target['A'] == "H":
        avail="H"
    elif threat['A'] == "L"or target['A'] == "L"and (threat['A'] !=
"H"or target['A'] != "H"):
        avail="L"
    else:
        avail="N"

    if threat['PR'] == "N"or target['PR'] == "N":
        pr="N"
    elif threat['PR'] == "L"or target['PR'] == "L"and (threat['PR'] !=
"N"or target['PR'] != "N"):
        pr="L"
    else:
        pr="H"

    #NALP
    if threat['AV'] == "N"or target['AV'] == "N":
        av="N"
    elif threat['AV'] == "A"or target['AV'] == "A"and (threat['AV'] !=
"N"or target['AV'] != "N"):
        av="A"
    elif threat['AV'] == "L"or target['AV'] == "L"and (threat['AV'] !=
"A"or target['AV'] != "A"or threat['AV'] != "N"or target['AV'] != "N"):
        av="L"
    else:
        av="P"

    chained_vector=
f"CVSS:3.1/AV:{av}/AC:{ac}/PR:{pr}/UI:{ui}/S:{scope}/C:{conf}/I:{integ}/A:{ava
il}"

    #
    return CVSS3(chained_vector).scores()[0]
    #returnchained_vector
else:

```

```

        return "unkown"

#checking related weakness from the CWE related pattern list
cwe_patterns.Related_Attack_Patterns.fillna("nothing",inplace=True)
def cwe_related_patterns(h):

    if h != "nothing":
        return h.strip().split("::")
cwe_patterns['related_list']=cwe_patterns.Related_Attack_Patterns.map(cwe_related_patterns)
pattern_litmus_dic=dict(zip(cwe_patterns.CWE_ID,cwe_patterns.related_list))
def checking_related_pattern(vec):

    if cweid_dic.get(vec[0]) or cweid_dic.get(vec[1]):
        if cweid_dic.get(vec[0])
incweid_dic.get(vec[1]) or cweid_dic.get(vec[1]) incweid_dic.get(vec[0]):
        return "probable relation"
        return "no information"

    return "no information"

# In[19]:

chained_list['vulnerability_connector']=chained_list.threat_target.map(combined_connector)
chained_list['cvssv3_chained_score']=chained_list.threat_target.map(combined_chained_score)
chained_list['relation_for_chaining']=chained_list.threat_target.map(checking_related_pattern)
#
chained_list["zhang_connector"]=chained_list.threat_target.map(combined_connector)

# In[20]:

#
chained_list["zhang_connector"]=chained_list.threat_target.map(zhang_connectivity_op)

# # Finally Generated csv file List

# In[21]:

```

```

def target_source_av(v3):
    if nvd_score_dic.get(v3[0]):
        threat_lis = nvd_score_dic.get(v3[0]).split('/')
    if nvd_score_dic.get(v3[1]):
        target_lis = nvd_score_dic.get(v3[1]).split('/')
    if nvd_score_dic.get(v3[1]) and nvd_score_dic.get(v3[0]):
        threat = {x.split(':')[0]:x.split(':')[1] for x in threat_lis}
        target = {x.split(':')[0]:x.split(':')[1] for x in target_lis}
    return f"{threat['AV']}-{target['AV']}"

def filter_source_target_logic(x):
    if x.split('-')[0] == "P" or x.split('-')[1] == "P" or x == "A-A":
        return False
    return True

chained_list['av_source_target'] = chained_list.threat_target.map(target_source_av)
chained_list['chain_logic'] = chained_list.av_source_target.map(filter_source_target_logic)

# In[22]:

# filtering with chainlogic values
chained_list = chained_list[chained_list.chain_logic == True]
chained_list.drop('chain_logic', 1)
print('nothing')

# In[23]:

# simple splitting functions
def source_split(x):
    return x[0]
def target_split(x):
    return x[1]
def compare(score):
    if score == 0:
        return "None"
    elif 0.1 <= score <= 3.9:
        return "Low"
    elif 4 <= score <= 6.9:
        return "Medium"
    elif 7 <= score <= 8.9:
        return "High"

```

```

elif 9 <= score <= 10:
    return "Critical"

chained_list['severity'] = chained_list.cvssv3_chained_score.map(compare)
chained_list['source_cve'] = chained_list.threat_target.map(source_split)
chained_list['target_cve'] = chained_list.threat_target.map(target_split)

# # Sections following are sample ways of using generated list

# In[24]:

chained_list.vulnerability_connector.describe()

# In[25]:

chained_list.cvssv3_chained_score.describe()

# In[26]:

chained_list.head(5)

# In[27]:

target_weakness = chained_list.groupby(['target_cve'])['severity'].value_counts(
).reset_index(name='severity_counts')
target_weakness.head(10)

# In[28]:

#
target_weakness_normalized = chained_list.groupby(['target_cve'])['severity'].value_counts(normalize=True).reset_index(name='severity_weights')
# target_weakness_normalized.head(10)
tw = chained_list.groupby(['target_cve'])['severity'].value_counts(normalize=True)

# In[29]:

```

```

twm

# In[30]:

cwm=dict(zip(twm.index, twm.values))

# In[31]:

#
target_weakness_connector=chained_list.groupby(['target_cve', 'severity'])['vulnerability_connector'].mean().sort_values(ascending=False)
twc=chained_list.groupby(['target_cve', 'severity'])['vulnerability_connector'].mean()

# In[32]:

vcd=dict(zip(twc.index, twc.values))

# In[33]:

tv3s=chained_list.groupby(['target_cve', 'severity'])['cvssv3_chained_score'].mean()
# tv3s=chained_list.groupby(['target_cve'])['cvssv3_chained_score'].mean()
tv3s

# In[34]:

# chained_list.groupby

# In[35]:

smd=dict(zip(tv3s.index, tv3s.values))

# In[36]:

ptc=chained_list.groupby('target_cve')['relation_for_chaining'].value_counts(
)

```

```

# In[37]:

ptcd=dict(zip(ptc.index,ptc.values))

# In[38]:

final_product=pd.DataFrame({'target_cve':chained_list.target_cve.unique()})

defweighted_average_connectivity(x):
    s=["High","Medium","Low","Critical"]
    return
    ((twc.get((x,s[0]),0)*twn.get((x,s[0]),0))+(twc.get((x,s[1]),0)*twn.get((x,s[1]
    ]),0))+(twc.get((x,s[2]),0)*twn.get((x,s[2]),0))+(twc.get((x,s[3]),0)*twn.get(
    (x,s[3]),0)))/(twn.get((x,s[0]),0)+twn.get((x,s[1]),0)+twn.get((x,s[2]),0)+twn
    .get((x,s[3]),0))

defaverage_v3_score(x):
    s=["High","Medium","Low","Critical"]
    return
    ((smd.get((x,s[0]),0)*twn.get((x,s[0]),0))+(smd.get((x,s[1]),0)*twn.get((x,s[1]
    ]),0))+(smd.get((x,s[2]),0)*twn.get((x,s[2]),0))+(smd.get((x,s[3]),0)*twn.get(
    (x,s[3]),0)))/(twn.get((x,s[0]),0)+twn.get((x,s[1]),0)+twn.get((x,s[2]),0)+twn
    .get((x,s[3]),0))
defpattern_count(x):
    returnptcd.get((x,"probable relation"),0)

final_product['average_connectivity']=final_product.target_cve.map(weighted_av
erage_connectivity)
final_product['average_cvss3_score']=final_product.target_cve.map(average_v3_s
core)
final_product['relation_count']=final_product.target_cve.map(pattern_count)

# In[39]:

plt.title("cwerealtion count distribution in target")
sns.violinplot(final_product.relation_count)
plt.savefig('relationcount.png')

# In[40]:

plt.title("vulnerabilitiy connectivity distribution in target")

```

```

sns.violinplot(final_product.average_connectivity)
plt.savefig('acon_frequencey.png')

# In[41]:

plt.title("CVSS v3 chained score distribution in target")
sns.violinplot(final_product.average_cvss3_score)
plt.savefig('av3socre_frequencey.png')

# In[42]:

#maximumvulnerablity connectivity can assume
defmax_connectivity(x):
    return x/(x+math.sqrt(0.2*0.44))
gt=np.arange(0.27, 0.85, 0.01)
plt.plot(gt, max_connectivity(gt),'r--')

# In[43]:

#minimumvulnerablityconnectititvity can assume
defmin_connectivity(x):
    return x/(x+math.sqrt(0.77*0.85))
gt=np.arange(0.27, 0.85, 0.01)
plt.plot(gt, min_connectivity(gt),'r--')

# In[44]:

# Max connector value
print(0.85/(0.85+math.sqrt(0.2*0.44))) # max connectivity value

# In[45]:

#minconnector value
print(0.27/(0.27+math.sqrt(0.7*0.85))) # min connectivity value

# # Below is exporting chained list with cvss scores and
# vulnerability_connectivity scores

# In[46]:

```

```

# data with scores and that do not have "Physical" Attack vector value in the
target node
final_product.head(10)

# In[47]:

# data that have "Physical" Attack vector value in the target node

def filter_physical(x):
    sam=x.split("/")
    test_dic={x.split(':')[0]:x.split(':')[1] for x in sam}
    if test_dic.get('AV',None)=="P":
        return True
    return False
targ_pd=pd.DataFrame({'cve_number':target_list_with_score.cve_number,'cvss_vector':target_list_with_score.cvss_scorev3},columns=['cve_number','cvss_vector'])

targ_pd["physical"]=targ_pd.cvss_vector.map(filter_physical)
# targ_pd['base_score']=targ_pd.cvss_vector.map(cvs3_sc)
targ_pd[targ_pd.physical != False]
# print(CVSS3('CVSS:3.1/AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H').scores()[0])

# In[48]:

# weakness in target node with no CVSS scores
target_list_with_nov3.iloc[:,[0,3]].head(10)

# # Some Numeric Evaluation of Results

# In[49]:

print(f"length of target cve is : {len(target_cves.cve_number)}")
print(f"length of threat cve is : {len(threat_cves.cve_number)}")
print(f"length of target cve with cvss scores is : {len(target_list_with_score.cve_number)}")
print(f"length of threat cve with cvss scores is : {len(threat_list_with_score.cve_number)}")

# In[50]:

```

```

len(generated_combination)
#
generated_combination=list(product(threat_list_with_score.cve_number,target_list_with_score.cve_number))

#
three_stage_threat_path=(list(product(threat_list_with_score.cve_number,threat_list_with_score.cve_number,target_list_with_score.cve_number)))
three_stage_threat_path=(list(product(threat_list_with_score.cve_number,threat_list_with_score.cve_number,target_list_with_score.cve_number)))
print(f"    The    single    stage    paths    that    can    be    formed are {len(generated_combination)}")
print(f"    The    three    stage    paths    that    can    be    formed are {len(three_stage_threat_path)}") # if multistage path generation for risk assessment was employed

# In[51]:

len(final_product[final_product.average_cvss3_score >4])

# In[52]:

# interms of performance of the approach
# realexploitation should be conducted and compare results

# # Finally Exporting to CSV

# In[53]:

# exporting to cvss
# final_product.to_csv("finalproduct.csv",index=False,encoding="utf-8")

```

APPENDIX B

```
#####
command (this is taken as target)
nmap -sV -p- --osscan-guess -T3 -A -f 10.90.10.76
#####
Nmap scan report for 10.90.10.76
Host is up (0.00045s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE VERSION
22/tcpopen  ssh      OpenSSH 6.7p1 Debian 5+deb8u3 (protocol 2.0)
| ssh-hostkey:
|   1024 b0:ac:cd:cb:72:1a:51:f4:a7:1f:2c:31:02:a4:8a:47 (DSA)
|   2048 e7:eb:30:03:38:1e:13:bd:83:aa:22:e3:aa:7c:f3:32 (RSA)
|   256 49:08:93:cc:6b:7a:ed:c9:d4:d7:8e:19:b0:40:3c:21 (ECDSA)
|_  256 75:ef:56:ce:8f:3f:eb:72:b5:51:67:3d:ea:f3:d8:43 (ED25519)
80/tcpopen  http      Apache httpd 2.4.10 ((Debian))
| http-robots.txt: 6 disallowed entries
| /calendar/action~posterboard/
| /calendar/action~agenda/ /calendar/action~oneday/ /calendar/action~month/
|_ /calendar/action~week/ /calendar/action~stream/
|_ http-server-header: Apache/2.4.10 (Debian)

Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 3 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 1720/tcp)
HOP RTT      ADDRESS
1   10.36 ms  _gateway (10.9.30.1)
2   36.22 ms  10.1.0.89
3   0.23 ms   10.90.10.76

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.60 seconds
#####
command
sudonmap -sV -F --osscan-guess -T2 -A -f
10.9.30.1/10.1.0.89/10.1.0.22/10.1.41.14
#####
Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-14 11:28 EAT
Nmap scan report for 10.1.0.89
Host is up (0.0040s latency).
All 100 scanned ports on 10.1.0.89 are filtered
```

```

Warning: OSScan results may be unreliable because we could not find at least 1
open and 1 closed port
Device type: switch
Running: Cisco IOS 12.X
OS CPE: cpe:/o:cisco:ios:12.1
OS details: Cisco 2900-series or 3700-series switch (IOS 12.1)
Network Distance: 2 hops

TRACEROUTE (using proto 1/icmp)
HOP RTT      ADDRESS
1   4.13 ms  _gateway (10.9.30.1)
2   3.29 ms  10.1.0.89

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 89.02 seconds
#####
#####
command
sudonmap -sV -F --osscan-guess -T3 -A -f 10.1.0.22
#####
#####
Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-14 11:30 EAT
Nmap scan report for 10.1.0.22
Host is up (0.0027s latency).
All 100 scanned ports on 10.1.0.22 are filtered
Warning: OSScan results may be unreliable because we could not find at least 1
open and 1 closed port
Device type: switch
Running: Cisco IOS 12.X
OS CPE: cpe:/o:cisco:ios:12.1
OS details: Cisco 2900-series or 3700-series switch (IOS 12.1)
Network Distance: 3 hops

TRACEROUTE (using proto 1/icmp)
HOP RTT      ADDRESS
1   99.70 ms  _gateway (10.9.30.1)
2   22.73 ms  10.1.0.89
3    2.00 ms  10.1.0.22

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 4.93 seconds
#####
#####
command
sudonmap -sV -F --osscan-guess -T2 -A -f 10.6.1.11

```

```
#####
#####
Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-14 11:36 EAT
Nmap scan report for 10.6.1.11
Host is up (0.00039s latency).
Not shown: 95 filtered ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          (protocol 2.0)
| fingerprint-strings:
|   NULL:
|_  SSH-2.0-AeDepc
| ssh-hostkey:
|   2048 57:0d:a5:bd:e5:f6:3a:03:a2:89:49:6d:06:f0:de:0a (RSA)
|_  256 10:fa:a8:48:f6:d0:ae:32:74:b9:7d:75:3f:e7:b6:f4 (ED25519)
80/tcp    open  tcpwrapped
|_ http-server-header: xxxxxxxx-xxxxx
|_ http-title: Did not follow redirect to https://10.6.1.11:443/
113/tcp   closed ident
179/tcp   open  tcpwrapped
443/tcp   open  ssl/tcpwrapped
|_ http-favicon: Fortinet
|_ http-server-header: xxxxxxxx-xxxxx
|_ http-title: Site doesn't have a title (text/html; charset=utf-8).
|                                     ssl-cert: Subject:
commonName=FG1K5DT918802592/organizationName=Fortinet/stateOrProvinceName=California/countryName=US
| Not valid before: 2018-07-27T03:40:19
|_ Not valid after: 2038-01-19T03:14:07
|_ ssl-date: TLS randomness does not represent time
1 service unrecognized despite returning data. If you know the service/version,
please submit the following fingerprint at https://nmap.org/cgi-
bin/submit.cgi?new-service :
SF-Port22-TCP:V=7.80%I=7%D=9/14%Time=61405F87%P=x86_64-redhat-linux-gnu%(
SF:NULL,10,"SSH-2\0-AeDepc\r\n");
Device type: general purpose
Running (JUST GUESSING): OpenBSD 4.X (85%)
OS CPE: cpe:/o:openbsd:openbsd:4.0
Aggressive OS guesses: OpenBSD 4.0 (85%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 3 hops

TRACEROUTE (using port 113/tcp)
HOP RTT      ADDRESS
1   13.12 ms  _gateway (10.9.30.1)
2    6.69 ms  10.1.0.89
3    0.39 ms  10.6.1.11
```

```

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 136.57 seconds
#####
#####
command
sudonmap -sV -F --osscan-guess -T2 -A -f 10.133.193.243
#####
#####
Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-14 11:37 EAT
Nmap scan report for 10.133.193.243
Host is up (0.0010s latency).
Not shown: 94 filtered ports
PORT      STATE SERVICE      VERSION
113/tcp    closed ident
179/tcp    open  tcpwrapped
646/tcp    open  tcpwrapped
2000/tcp   open  cisco-sccp?
5060/tcp   open  sip?
8008/tcp   open  http
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.1 302 Found
|     Location: https://:8010/nice%20ports%2C/Tri%6Eity.txt%2ebak
|     Connection: close
|     X-Frame-Options: SAMEORIGIN
|     X-XSS-Protection: 1; mode=block
|     X-Content-Type-Options: nosniff
|     Content-Security-Policy: frame-ancestors
|   GenericLines, HTTPOptions, RTSPRequest, SIPOptions:
|     HTTP/1.1 302 Found
|     Location: https://:8010
|     Connection: close
|     X-Frame-Options: SAMEORIGIN
|     X-XSS-Protection: 1; mode=block
|     X-Content-Type-Options: nosniff
|     Content-Security-Policy: frame-ancestors
|   GetRequest:
|     HTTP/1.1 302 Found
|     Location: https://:8010/
|     Connection: close
|     X-Frame-Options: SAMEORIGIN
|     X-XSS-Protection: 1; mode=block
|     X-Content-Type-Options: nosniff
|     Content-Security-Policy: frame-ancestors
|_ http-title: Did not follow redirect to https://10.133.193.243:8010/
|_ https-redirect: ERROR: Script execution failed (use -d to debug)

```

```

1 service unrecognized despite returning data. If you know the service/version,
please submit the following fingerprint at https://nmap.org/cgi-
bin/submit.cgi?new-service :
SF-Port8008-TCP:V=7.80%I=7%D=9/14%Time=614060DA%P=x86_64-redhat-linux-gnu%
SF:r(GetRequest,CC,"HTTP/1\1\20302\20Found\r\nLocation:\20https://:801
SF:0\r\nConnection:\20close\r\nX-Frame-Options:\20SAMEORIGIN\r\nX-XSS-P
SF:rotection:\20;\20mode=block\r\nX-Content-Type-Options:\20nosniff\r\
SF:nContent-Security-Policy:\20frame-ancestors\r\n\r\n")%r(FourOhFourRequ
SF:est,EF,"HTTP/1\1\20302\20Found\r\nLocation:\20https://:8010/nice%20
SF:ports%2C/Tri%6Eity\1.txt%2ebak\r\nConnection:\20close\r\nX-Frame-Option
SF:s:\20SAMEORIGIN\r\nX-XSS-Protection:\20;\20mode=block\r\nX-Content-
SF:Type-Options:\20nosniff\r\nContent-Security-Policy:\20frame-ancestors
SF:\r\n\r\n")%r(GenericLines,CB,"HTTP/1\1\20302\20Found\r\nLocation:\2
SF:0https://:8010\r\nConnection:\20close\r\nX-Frame-Options:\20SAMEORIGI
SF:N\r\nX-XSS-Protection:\20;\20mode=block\r\nX-Content-Type-Options:\x
SF:20nosniff\r\nContent-Security-Policy:\20frame-ancestors\r\n\r\n")%r(HT
SF:TPOptions,CB,"HTTP/1\1\20302\20Found\r\nLocation:\20https://:8010\r
SF:\nConnection:\20close\r\nX-Frame-Options:\20SAMEORIGIN\r\nX-XSS-Prote
SF:ction:\20;\20mode=block\r\nX-Content-Type-Options:\20nosniff\r\nCon
SF:tent-Security-Policy:\20frame-ancestors\r\n\r\n")%r(RTSPRequest,CB,"HT
SF:TP/1\1\20302\20Found\r\nLocation:\20https://:8010\r\nConnection:\2
SF:0close\r\nX-Frame-Options:\20SAMEORIGIN\r\nX-XSS-Protection:\20;\20
SF:mode=block\r\nX-Content-Type-Options:\20nosniff\r\nContent-Security-Po
SF:licy:\20frame-ancestors\r\n\r\n")%r(SIPOptions,CB,"HTTP/1\1\20302\2
SF:0Found\r\nLocation:\20https://:8010\r\nConnection:\20close\r\nX-Frame
SF:-Options:\20SAMEORIGIN\r\nX-XSS-Protection:\20;\20mode=block\r\nX-C
SF:ontent-Type-Options:\20nosniff\r\nContent-Security-Policy:\20frame-an
SF:cestors\r\n\r\n");
OS fingerprint not ideal because: maxTimingRatio (4.000000e+00) is greater than
1.4
No OS matches for host
Network Distance: 3 hops

TRACEROUTE (using port 113/tcp)
HOP RTT ADDRESS
1 11.08 ms _gateway (10.9.30.1)
2 24.91 ms 10.1.0.89
3 0.25 ms 10.133.193.243

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 618.19 seconds
#####
#####
command
sudonmap -sV -F --osscan-guess -T2 -A -f 1.1.1.2
#####
#####

```

```

Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-14 12:39 EAT
Nmap scan report for 1.1.1.2
Host is up (0.0017s latency).
Not shown: 94 filtered ports
PORT      STATE SERVICE      VERSION
113/tcp    closed ident
179/tcp    open  tcpwrapped
646/tcp    open  tcpwrapped
2000/tcp   open  cisco-sccp?
5060/tcp   open  sip?
8008/tcp   open  http
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.1 302 Found
|     Location: https://:8010/nice%20ports%2C/Tri%6Eity.txt%2ebak
|     Connection: close
|     X-Frame-Options: SAMEORIGIN
|     X-XSS-Protection: 1; mode=block
|     X-Content-Type-Options: nosniff
|     Content-Security-Policy: frame-ancestors
|   GenericLines, HTTPOptions, RTSPRequest, SIPOptions:
|     HTTP/1.1 302 Found
|     Location: https://:8010
|     Connection: close
|     X-Frame-Options: SAMEORIGIN
|     X-XSS-Protection: 1; mode=block
|     X-Content-Type-Options: nosniff
|     Content-Security-Policy: frame-ancestors
|   GetRequest:
|     HTTP/1.1 302 Found
|     Location: https://:8010/
|     Connection: close
|     X-Frame-Options: SAMEORIGIN
|     X-XSS-Protection: 1; mode=block
|     X-Content-Type-Options: nosniff
|_   Content-Security-Policy: frame-ancestors
|_http-title: Did not follow redirect to https://1.1.1.2:8010/
|_https-redirect: ERROR: Script execution failed (use -d to debug)
1 service unrecognized despite returning data. If you know the service/version,
please submit the following fingerprint at https://nmap.org/cgi-
bin/submit.cgi?new-service :
SF-Port8008-TCP:V=7.80%I=7%D=9/14%Time=61406F58%P=x86_64-redhat-linux-gnu%
SF:r(GetRequest,CC,"HTTP/1\1\20302\20Found\r\nLocation:\20https://:801
SF:0/\r\nConnection:\20close\r\nX-Frame-Options:\20SAMEORIGIN\r\nX-XSS-P
SF:rotection:\201;\20mode=block\r\nX-Content-Type-Options:\20nosniff\r\
SF:nContent-Security-Policy:\20frame-ancestors\r\n\r\n")%r(FourOhFourRequ
SF:est,EF,"HTTP/1\1\20302\20Found\r\nLocation:\20https://:8010/nice%20
SF:ports%2C/Tri%6Eity\1.txt%2ebak\r\nConnection:\20close\r\nX-Frame-Option

```

```

SF:s:\x20SAMEORIGIN\r\nX-XSS-Protection:\x201;\x20mode=block\r\nX-Content-
SF:Type-Options:\x20nosniff\r\nContent-Security-Policy:\x20frame-ancestors
SF:\r\n\r\n")%r(GenericLines,CB,"HTTP/1\1\x20302\x20Found\r\nLocation:\x2
SF:0https://:8010\r\nConnection:\x20close\r\nX-Frame-Options:\x20SAMEORIGI
SF:N\r\nX-XSS-Protection:\x201;\x20mode=block\r\nX-Content-Type-Options:\x
SF:20nosniff\r\nContent-Security-Policy:\x20frame-ancestors\r\n\r\n")%r(HT
SF:TPOptions,CB,"HTTP/1\1\x20302\x20Found\r\nLocation:\x20https://:8010\r
SF:\nConnection:\x20close\r\nX-Frame-Options:\x20SAMEORIGIN\r\nX-XSS-Prote
SF:ction:\x201;\x20mode=block\r\nX-Content-Type-Options:\x20nosniff\r\nCon
SF:tent-Security-Policy:\x20frame-ancestors\r\n\r\n")%r(RTSPRequest,CB,"HT
SF:TP/1\1\x20302\x20Found\r\nLocation:\x20https://:8010\r\nConnection:\x2
SF:0close\r\nX-Frame-Options:\x20SAMEORIGIN\r\nX-XSS-Protection:\x201;\x20
SF:mode=block\r\nX-Content-Type-Options:\x20nosniff\r\nContent-Security-Po
SF:licy:\x20frame-ancestors\r\n\r\n")%r(SIPOptions,CB,"HTTP/1\1\x20302\x2
SF:0Found\r\nLocation:\x20https://:8010\r\nConnection:\x20close\r\nX-Frame
SF:-Options:\x20SAMEORIGIN\r\nX-XSS-Protection:\x201;\x20mode=block\r\nX-C
SF:ontent-Type-Options:\x20nosniff\r\nContent-Security-Policy:\x20frame-an
SF:cestors\r\n\r\n");
OS fingerprint not ideal because: maxTimingRatio (4.002000e+00) is greater than
1.4
No OS matches for host
Network Distance: 3 hops

TRACEROUTE (using port 113/tcp)
HOP RTT ADDRESS
1 10.68 ms _gateway (10.9.30.1)
2 9.68 ms 10.1.0.89
3 0.33 ms 1.1.1.2

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 626.50 seconds

#####
#####
command (this is the DNS server)
sudonmap -sV -F --osscan-guess -T2 -A -f 10.90.10.28/10.90.10.29
#####
#####
Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-14 13:22 EAT
Nmap scan report for 10.90.10.28
Host is up (0.00047s latency).
Not shown: 98 closed ports
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol
2.0)
| ssh-hostkey:
| 2048 1a:80:13:ba:12:1d:2e:6c:65:10:e1:1c:2d:35:17:1b (RSA)

```

```
| 256 86:43:c2:d5:23:bd:f4:c2:e0:f9:e4:4a:bd:fd:a3:b7 (ECDSA)
|_ 256 11:61:5f:3c:a8:d2:ea:42:1e:fe:91:fd:51:80:d4:45 (ED25519)
53/tcpopen domain ISC BIND 9.10.3-P4 (Ubuntu Linux)
| dns-nsid:
|_ bind.version: 9.10.3-P4-Ubuntu
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 3 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 554/tcp)
HOP RTT ADDRESS
1 9.55 ms _gateway (10.9.30.1)
2 5.66 ms 10.1.0.89
3 0.25 ms 10.90.10.28

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 72.85 seconds
```

APPENDIX C

Results that suberize formed edges

	target_cve	average_connectivity	average_cvss3_score	relation_count
0	CVE-2000-0076	0.344841	6.946544	29
1	CVE-2000-0366	0.344841	6.946544	29
2	CVE-2000-0367	0.339565	7.137912	29
3	CVE-2000-1135	0.344841	7.988940	29
4	CVE-2000-1136	0.344841	7.988940	29
5	CVE-2000-1169	0.316618	8.116129	29
6	CVE-2001-0069	0.344841	6.946544	29
7	CVE-2001-0195	0.344841	7.085253	29
8	CVE-2001-0361	0.278934	7.105069	5
9	CVE-2001-0456	0.316618	8.116129	29

Results that show CVEs with no Score

	cve_number	description
365	CVE-1999-0373	Buffer overflow in the "Super" utility in Debi...
366	CVE-1999-0374	Debian GNU/Linux cfengine package is susceptib...
380	CVE-1999-0389	Buffer overflow in the bootp server in the Deb...
634	CVE-1999-0661	A system is running a version of software that...
668	CVE-1999-0698	Denial of service in IP protocol logger (ippl)...
699	CVE-1999-0730	The zsoelim program in the Debian man-db packa...
700	CVE-1999-0732	The logging facility of the Debian smtp-refuse...
709	CVE-1999-0742	The Debian mailman package uses weak authentic...
881	CVE-1999-0914	Buffer overflow in the FTP client in the Debia...
906	CVE-1999-0939	Denial of service in Debian IRC Epic/epic4 cli...

Results that show CVEs not considered in the chaining process

	cve_number	cvss_vector	physical
91745	CVE-2016-4484	CVSS:3.0/AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	True