4-2019

# Machine Learning for Real-Time Data-Driven Security Practices

Shane Coleman
*Department of Computing, Institute of Technology, Tralee, Kerry, Ireland*

# Machine Learning for Real-Time Data-Driven Security Practices

by

Shane Coleman

**Supervisors**

Mr. Andrew Shields

Dr. Pat Doody

A Thesis Submitted to Quality and Qualifications Ireland in Fulfilment of the Requirements for the Master of Science Degree

April 2019

# Abstract

The risk of cyber-attacks exploiting vulnerable organisations has increased significantly over the past several years. Cyber-attacks can be described as any type of aggressive strategy which targets computer information systems, computer networks, personal computer systems or other organisational infrastructures which may originate internally or externally. These attacks may combine to exploit a vulnerability breach within a system's protection strategy which has the potential for loss, damage or destruction of assets. Consequently, every vulnerability has an accompanying risk which is defined as the *"*intersection of assets, threats, and vulnerabilities*"*.

This research project uses various types of recommender system techniques, employed for the identification and similarity-based ranking of cyber security information, relating to software and hardware vulnerabilities. Here the hypothesis is that the similarity-based ranking of this cyber security information can increase the user satisfaction of security personnel through a ranked list of recommended security information. For this research project, Top-N collaborative filtering recommender system techniques were used, specifically the User-Based and Item-Based Memory-Based methods and state-of-the-art approaches of SLIM and FISM. Three User-Item-Rating datasets were constructed through the National Vulnerability Database (NVD) which were employed by the recommendation techniques. In addition, Top-N evaluation was performed through the AUC, NDCG and MAP metrics.

Results show that the FISM Top-N techniques out-perform both Memory-Based methods and SLIM approach for all the three software and hardware user-item-rating datasets. This Top-N Collaborative Filtering technique obtained the highest Top-N evaluation accuracy which offers security personal a top 10 recommendation list of software and hardware vulnerabilities based on the similarity of vulnerable assets and a vulnerability severity score. Furthermore, the FISM technique shows a significant improvement over Memory-Based and other state-of-the-art Collaborative Filtering techniques, through the Top-N evaluation of alternate real recommender system datasets.

# Acknowledgements

I would like to express my gratitude to both my supervisors Mr. Andrew Shields and Dr. Pat Doody for their continuous guidance and support throughout my thesis. I am indebted to my supervisors who always made time for me throughout my masters. I wish to extend my thanks to all my colleagues in IMaR and the Computing Department of the Institute of Technology Tralee.

Lastly, I would like to give a special thank you to Róisín Harty, her family and my family for their continual love and support, without which I would not be here.

# Table of Contents

**List of Figures** ..... vii
**List of Tables** ..... viii
**Preface** ..... x
**Chapter 1 Introduction** ..... 1
1.1 Project Motivation ..... 1
1.2 Aims and Objectives of Project ..... 3
1.3 Thesis Structure ..... 4
**Chapter 2 Security** ..... 6
2.1 Introduction ..... 6
2.2 Security within the Cyber Domain ..... 6
2.2.1 Cyber Security ..... 6
2.2.2 Information Security ..... 7
2.2.3 Information and Communication Technology (ICT) Security ..... 8
2.2.4 Information and ICT Security to Cyber Security ..... 9
2.3 Vulnerabilities ..... 10
2.4 Threat ..... 10
2.4.1 Threat Event ..... 11
2.4.2 Threat Source ..... 11
2.4.3 Threat Actor ..... 12
2.5 Attack ..... 12
2.5.1 Passive and Active Attacks ..... 13
2.5.2 Inside and Outside Attacks ..... 13
2.6 Risk ..... 14
2.6.1 Risk Impact ..... 15
2.6.2 Risk Likelihood ..... 15
2.7 Summary ..... 16
**Chapter 3 National Vulnerability Database** ..... 17
3.1 Introduction ..... 17
3.2 National Vulnerability Database Overview ..... 17
3.3 Components of the National Vulnerability Database ..... 19
3.3.1 Common Vulnerability and Exposures (CVE) ..... 20
3.3.2 Common Vulnerability Scoring System (CVSS) ..... 21
3.3.3 Common Weakness Enumeration (CWE) ..... 27

3.3.4 Common Platform Enumeration (CPE) ..... 27
3.4 The Use of the National Vulnerability Database within Research ..... 32
3.5 Summary ..... 37
**Chapter 4 Recommender Systems** ..... 38
4.1 Introduction ..... 38
4.2 Recommender Systems ..... 38
4.3 Data ..... 39
4.3.1 Items ..... 40
4.3.2 Users ..... 40
4.3.3 Transactions ..... 40
4.3.4 Ratings ..... 41
4.4 Categories of Recommender Systems ..... 41
4.4.1 Content-Based Filtering in contrast with Collaborative Filtering ..... 42
4.5 Collaborative Filtering ..... 44
4.5.1 Collaborative Filtering Process Overview ..... 44
4.6 Memory-Based Collaborative Filtering ..... 46
4.7 User-Based Collaborative Filtering ..... 47
4.7.1 Prediction ..... 47
4.7.2 Recommendation ..... 47
4.7.3 Similarity Computation Between Users for Prediction or Recommendation ..... 48
4.8 Item-Based Collaborative Filtering ..... 51
4.8.1 Prediction ..... 51
4.8.2 Recommendation ..... 51
4.8.3 Similarity Computation Between Items for Prediction or Recommendation ..... 52
4.8.4 Advantages and Disadvantages of Memory-Based Collaborative Filtering Technique ..... 54
4.9 Model-Based Collaborative Filtering ..... 57
4.9.1 Latent Factor Models ..... 58
4.9.2 Matrix Factorisation ..... 58
4.9.3 Relevant Research of Matrix Factorisation Based Models ..... 60
4.9.4 Related Work of Matrix Factorisation within a Top-N Recommendation Context ..... 61
4.10 Recommender System Evaluation Metrics ..... 64
4.10.1 Prediction Accuracy Metrics ..... 64

4.10.2 Classification and Ranking Accuracy Metrics ..... 66
4.11 Summary ..... 71
**Chapter 5 Methodology** ..... 72
5.1 Introduction ..... 72
5.2 Data Collection / Information Retrieval ..... 72
5.2.1 Open Vulnerability Data: National Vulnerability Database and CVE Details 73
5.3 Data Pre-Processing and Data Analysis ..... 76
5.4 Recommender System Tools and Techniques ..... 83
5.5 Experimental Setup / Design ..... 84
5.6 Experimental Evaluation Measures ..... 85
5.7 Summary ..... 85
**Chapter 6 Experiments and Results** ..... 87
6.1 Introduction ..... 87
6.2 Memory-Based Top-N Collaborative Filtering Neighbourhood Size Sensitivity via Similarity Measures ..... 88
6.2.1 Experiment A: Item-Based Neighbourhood Size Sensitivity via Pearson's Correlation Coefficient Similarity Measure ..... 88
6.2.2 Experiment B: Item-Based Neighbourhood Size Sensitivity via Cosine Similarity Measure ..... 90
6.2.3 Experiment C: User-Based Neighbourhood Size Sensitivity via Pearson's Correlation Coefficient Similarity Measure ..... 92
6.2.4 Experiment D: User-Based Neighbourhood Size Sensitivity via Cosine Similarity Measure ..... 94
6.3 Sparse Linear Methods (SLIM) Top-N Collaborative Filtering via regularisation Parameter Alteration ..... 96
6.4 Factored Item Similarity Models (FISM) Top-N Collaborative Filtering via Parameter Alteration ..... 98
6.5 Summary ..... 102
**Chapter 7 Discussion** ..... 103
7.1 Introduction ..... 103
7.2 Key Findings ..... 103
7.2.1 Research Question 1 ..... 104
7.2.2 Research Question 2 ..... 105
7.2.3 Research Question 3 ..... 106
7.2.4 Research Question 4 ..... 107
7.3 Summary ..... 110

**Chapter 8 Conclusion and Suggestions for Future Work** ..... 112
**References** ..... 114

# List of Figures

**Figure 3.1** XML representation of NVD / CVE entry "CVE-2014-4090".....18
**Figure 3.2** Graphical representation of NVD / CVE entry "CVE-2014-4090" through CVE Details .....19
**Figure 3.3** WFN for Microsoft Internet Explorer 8.....28
**Figure 3.4** WFN Bound to a Uniform Resource Identifier (URI).....29
**Figure 3.5** WFN Bound to a Format String.....29
**Figure 3.6** CPE 2.3 Entry for Microsoft Internet Explorer 8 (CPE Dictionary) .....31
**Figure 4.1** Recommender System Techniques.....42
**Figure 4.2** The Collaborative Filtering Process .....45
**Figure 4.3** The Separation of the Co-Rated Items and Similarity Computation.....53
**Figure 5.1** Trend of Vulnerability Numbers.....77
**Figure 5.2** Top 20 Vendors (2005 to 2014).....77
**Figure 5.3** Frequency of Specified Vendors (2005 to 2014).....78
**Figure 5.4** Top 20 Affected Software / Hardware (Specified Vendors) from 2005 to 2014.....79

# List of Tables

**Table 3.1** Possible Values for the Impact Sub Metric ..... 23
**Table 3.2** Possible Values for the Exploitability Sub Metric ..... 23
**Table 4.1** Key Advantages of Memory-Based Collaborative Filtering ..... 54
**Table 4.2** Precision Recall ..... 68
**Table 5.1** National Vulnerability Database Attributes ..... 73
**Table 5.2** CVE Details Attributes ..... 74
**Table 5.3** Table of the Frequency of Specified Vendors from the years 2005 to 2014 ..... 78
**Table 5.4** Top 20 Affected Software / Hardware – From Highest (1) to Lowest (20) ..... 80
**Table 5.5** User-Item-Rating Datasets and Attributes ..... 81
**Table 6.1** Item-Based (Pearson) Top-N evaluation results for the first software / hardware vulnerability dataset ..... 89
**Table 6.2** Item-Based (Pearson) Top-N evaluation results for the second software / hardware vulnerability dataset ..... 89
**Table 6.3** Item-Based (Pearson) Top-N evaluation results for the third software / hardware vulnerability dataset ..... 89
**Table 6.4** Item-Based (Cosine) Top-N evaluation results for the first software / hardware vulnerability dataset ..... 91
**Table 6.5** Item-Based (Cosine) Top-N evaluation results for the second software / hardware vulnerability dataset ..... 91
**Table 6.6** Item-Based (Cosine) Top-N evaluation results for the third software / hardware vulnerability dataset ..... 91
**Table 6.7** User-Based (Pearson) Top-N evaluation results for the first software / hardware vulnerability dataset ..... 93
**Table 6.8** User-Based (Pearson) Top-N evaluation results for the second software / hardware vulnerability dataset ..... 93
**Table 6.9** User-Based (Pearson) Top-N evaluation results for the third software / hardware vulnerability dataset ..... 93
**Table 6.10** User-Based (Cosine) Top-N evaluation results for the first software / hardware vulnerability dataset ..... 94

**Table 6.11** User-Based (Cosine) Top-N evaluation results for the second software / hardware vulnerability dataset .....95
**Table 6.12** User-Based (Cosine) Top-N evaluation results for the first software / hardware vulnerability dataset .....95
**Table 6.13** Sparse Linear Methods (SLIM) Top-N evaluation results for the first software / hardware vulnerability dataset .....96
**Table 6.14** Sparse Linear Methods (SLIM) Top-N evaluation results for the second software / hardware vulnerability dataset .....97
**Table 6.15** Sparse Linear Methods (SLIM) Top-N evaluation results for the third software / hardware vulnerability dataset .....97
**Table 6.16** Factored Item Similarity Models (FISM), FISMrmse Top-N evaluation results for the first software / hardware vulnerability dataset .....99
**Table 6.17** Factored Item Similarity Models (FISM), FISMrmse Top-N evaluation results for the second software / hardware vulnerability dataset .....99
**Table 6.18** Factored Item Similarity Models (FISM), FISMrmse Top-N evaluation results for the third software / hardware vulnerability dataset .....100
**Table 6.19** Factored Item Similarity Models (FISM), FISMauc Top-N evaluation results for the first software / hardware vulnerability dataset .....101
**Table 6.20** Factored Item Similarity Models (FISM), FISMauc Top-N evaluation results for the second software / hardware vulnerability dataset .....101
**Table 6.21** Factored Item Similarity Models (FISM), FISMauc Top-N evaluation results for the third software / hardware vulnerability dataset .....102
**Table 7.1** User-Item-Rating Datasets and Attributes .....103

# Preface

This thesis is presented to the Institute of Technology, Tralee for consideration of a Master of Science award. Extracts of this work have been presented at the conference detailed underneath.

**Conferences:**

- Coleman, S., Doody, P. and Shields, A. 2018. Machine Learning for Real-Time Data-Driven Security Practices. The 29th Irish Signals and Systems Conference (ISSC 2018), Queen's University Belfast, Belfast, Northern Ireland. (http://programme.exordo.com/issc2018/delegates/presentation/29/)

# Chapter 1 Introduction

## 1.1 Project Motivation

Within the last number of years the amount of vulnerabilities recognised on the Internet has increased significantly (Gallon, 2011). These vulnerabilities create a crucial issue for the world today which depends on information technology (Toloudis, Spanos and Angelis, 2016). Additionally, these vulnerabilities signify a leading cause of cyber security issues (Zhang, Caragea and Ou, 2011). According to E.N.I.S.A. the top three cyber threats of 2017 were identified as Malware, Web-Based Attacks and Web-Application Attacks; which are equivalent in number to the years of 2015 and 2016 combined. (European Union Agency for Network and Information Security (ENISA), 2018). These types of threats and attacks are made possible through vulnerabilities located within an organisation's networked applications, services, hardware and software. As stated by the Open Web Application Security Project (O.W.A.S.P.), the top three out of the ten most critical web application security risks for software and hardware vulnerabilities for the year 2017 were (1) Injection, (2) Broken Authentication and (3) Sensitive Data Exposure. In contrast, for the year 2013, the OWASP top three comprised of (1) Injection, (2) Broken Authentication and Session Management, and (3) Cross-Site Scripting (XSS) (Open Web Application Security Project (OWASP), 2018). Without adequate network, information and cyber security, these vulnerabilities can be easily exploited by malicious individuals which in turn can have a devastating impact on an organisation's reputation, image and possible loss of revenue.

Security has one fundamental goal and that is to protect assets. Within an organisation the protection of information is crucial. Security helps preserve and maintain the confidentiality, integrity and availability of information, especially in terms of personal customer information and financial records (von Solms and van Niekerk, 2013). The intersection of these three characteristics is known as the CIA triangle. It is used as the traditional standard within industry. Furthermore, security *"protects against malicious attacks by outsiders and by insiders"* (Rufi, 2006).

To protect sensitive information in an organisation's networked infrastructure, devices such as Firewalls were introduced. With the continual expansion of the Internet and the increasing number of devices connected through various network communication platforms, the potential risk of cyber-threats and breaches in cyber security has increased significantly. Leading on from this expansion in network communication organisations needed to expand from the safety of their closed private networks into more open public networks through the connection of additional networks and devices. This transition not only opened new business opportunities for organisations and increased the quantity of data being generated; it also amplified the variety and quantity of risks, threats and attacks which have the potential to impose damage.

Risks, vulnerabilities, threats and attacks may occur within an organisation's network infrastructure. They can compromise hardware, software and vital sensitive information, even when encryption mechanisms are applied. To alleviate these risks and protect an organisation's infrastructure strict cyber security procedures are implemented to uphold the security of an organisation.

In parallel with these methodologies, organisations use scoring metrics, measures and common language of security terminologies (National Institute of Standards and Technology (NIST), 2016). They are used to present organisations with a standardised structured naming scheme and act as a dictionary of common identifiers for existing security vulnerabilities, attacks and weaknesses located in software and hardware. This cyber security information is shared and made available within the wider security community. This is used to generate baselines for the security efforts using a numeric score or severity rating. Furthermore, organisations can implement numerous professional standards, protocols, policies and frameworks which evaluate and enhance current cyber security measures (National Institute of Standards and Technology, 2014). However, through these efforts, an enormous amount of data may be produced which can easily overwhelm security personnel in taking the correct course of action.

Security personnel need guidance when dealing with the large number of security issues they are presented with. They may have differing opinions on the importance and priority which security issues need to be investigated. Also, vulnerabilities, attacks or threats may fall under the same security category requiring a combined action. However, they may be perceived as individual issues by security personnel.

Vast amounts of diverse information are being generated through these methodologies and scoring metrics. This research project proposes that a machine learning technique known as a recommender system be employed for the identification and prioritisation of cyber security information relating to vulnerabilities located within software and hardware.

## 1.2 Aims and Objectives of Project

The major aim of this research project is to use various categories of recommender system techniques for the identification and similarity-based ranking and rank aggregation of cyber security information. This cyber security information relates to software and hardware vulnerabilities. The hypothesis is that the similarity-based ranking and rank aggregation of this cyber security information can assist security personnel through a ranked list of recommended security information.

The key objectives of this research project can be outlined as follows:

- Development and exploration of recommender system techniques which undertakes the duties of identifying and ranking relevant cyber security information, relating to software and hardware vulnerabilities.
- Ability to make security personnel aware of potential software and hardware vulnerabilities, by means of a recommendation list.
- Ability to acquire information grounded on the recommender system results and user preferences.
- Evaluation of recommender system techniques.
- Analysis of existing cyber security software and hardware data

- Investigation of many diverse machine learning recommender system techniques, for the intention of distributing cyber security information to users.

## 1.3 Thesis Structure

The main goal of this thesis is to investigate the use of diverse recommender system techniques. Focus is on the recommendation (similarity-based ranking) of cyber security vulnerabilities relating to software and hardware assets. This is achieved through a category of recommender system technique known as Top-N Collaborative Filtering, where users are presented with a ranked recommendation list of items for a user. Both traditional and state-of-the-art recommender system technique will be explored, through the employment of software and hardware vulnerability cyber security data and evaluated using numerous advanced Top-N evaluation metrics.

Chapter 2 outlines security within the cyber domain, while presenting an overview of common cyber security terminologies, such as a vulnerability, threat, attack and risk.

Chapter 3 describes the cyber security vulnerability database, known as the National Vulnerability Database (NVD). In addition, outlined within this chapter are the numerous components which is used for the construction of the National Vulnerability Database.

Chapter 4 presents an overview of recommender systems, the various categories of existing recommender system techniques and the metrics used to evaluate diverse recommender system approaches.

Chapter 5 outlines the methods used in this research, relating to the collection and pre-processing of data; the analysis of data; the recommender system tools and techniques used; the design and setup of experiments and the evaluation measures and metrics used within the experiments.

Chapter 6 presents the experiments undertaken and its subsequent results, where cyber security vulnerability data by means of the National Vulnerability Database was used through various recommender system techniques.

Chapter 7 discusses the results of experiments, through the investigation of key findings observed by means of several proposed research questions.

Chapter 8 will summarise the conclusions of this research, along with the suggestion of future work to be undertaken.

# Chapter 2 Security

## 2.1 Introduction

Through the establishment of Information and Communication Technologies in addition to the expanded availability of the Internet, organisations may become vulnerable to numerous categories of threats. In effect, an organisation's information may be revealed by means of cyber-attacks and its subsequent harm towards an organisation. These threats can derive from various origins, for example, the activities of employees or cyber-attacks through a hacker (Jouini, Rabai and Aissa, 2014). A hacker can be described as an *"unauthorised user who attempts to or gain access to an information system"* (National Institute of Standards and Technology (NIST) and Kissel, 2013). The financial damages produced through security breaches generally cannot exactly distinguished since a substantial quantity of financial damages originates through minor security events which produced an low judgement relating to the security risk of information systems (Jouini, Rabai and Aissa, 2014).

Section 2.2 of this chapter will describe security within the cyber domain. Sections 2.3 to 2.6 presents an overview of cyber security terminologies: vulnerabilities, threats, attacks and risk. Section 2.7 concludes this chapter with a summary.

## 2.2 Security within the Cyber Domain

Security relates to the safeguarding of assets from the numerous threats produced through specific intrinsic vulnerabilities. Security procedures are typically concerned with the choosing and employment of security controls (countermeasures) which aid to lessen the risk presented through these vulnerabilities (International Organisaton for Standardisation (ISO) and International Electrotechnical Commission (IEC), 2013; von Solms and van Niekerk, 2013).

### 2.2.1 Cyber Security

As identified through (von Solms and van Niekerk, 2013) cyber security is used as a comprehensive term which relates to procedures undertaken to protect a computer or computer systems connected to the Internet from unapproved admission or attack.

However, as acknowledged by the International Telecommunication Union (ITU) cyber security can be defined as follows:

*"Cyber security is the collection of tools, policies, security concepts, security safeguards, guidelines, risk management approaches, actions, training, best practices, assurance and technologies that can be used to protect, the cyber environment and organisation and user's assets. Organisation and user's assets include connected computing devices, personnel, infrastructure, applications, services, telecommunications systems and the totality of transmitted and/or stored information in the cyber environment. Cybersecurity strives to ensure the attainment and maintenance of the security properties of the organisation and user's assets against relevant security risks in the cyber environment. The general security objectives comprise of the following: availability; integrity which may include authenticity and non-repudiation, and confidentiality"* (International Telecommunication Union (ITU), 2008).

### 2.2.2 Information Security

The term cyber security can be comparable to that of information security. The goal of information security is to guarantee business progression and reduce harm of a business by restricting the impact of security occurrences (von Solms and van Niekerk, 2013). Information security according to the international standard ISO/IEC 27002 can be defined as the *"protection of information from a wide range of threats in order to ensure business continuity, minimise business risk and maximise return on investments and business opportunities"* (International Organisaton for Standardisation (ISO) and International Electrotechnical Commission (IEC), 2005). Information security acts on the protection of the confidentiality, integrity and availability of information. This information can take numerous forms: written or printed on paper, stored on films, transmitted or stored by post or electronically, communicated in conversation et cetera (von Solms and van Niekerk, 2013).

According to (Whitman and Mattord, 2011) safeguarding the confidentiality, integrity and availability of information within the area of information security is identified as the CIA triangle and has been used as the traditional standard within industry. The authors identify that the *"security of these three characteristics of information is as important today as it has always been, but the CIA triangle model*

*no longer adequately addresses the constantly changing environment of the computer industry"* (Whitman and Mattord, 2011).

Two factors were identified through (von Solms and van Niekerk, 2013). Firstly, information security is not a product or technology but a process. Secondly, information security is generally described in relation to the characteristics which protect information. These typically contain the characteristics: confidentiality, integrity and availability but can comprise of additional characteristics, for instance: non-repudiation, accountability, authentication, and reliability of information measures. Subsequently, information security embraces the safeguarding of the fundamental information measures. It can be proclaimed that Information and Communication Technology (ICT) security is a sub section of information security (von Solms and van Niekerk, 2013).

### 2.2.3 Information and Communication Technology (ICT) Security

Information and Communication Technology (ICT) security manages the protection of real technology-based systems where information is normally kept and/or communicated. According to the ISO/IEC 13335-1 international standard ICT security can be described as *"all characteristics associated to defining, accomplishing and preserving the confidentiality, integrity, availability, non-repudiation, accountability, authentication, and reliability of information measures"* (International Organisaton for Standardisation (ISO) and International Electrotechnical Commission (IEC), 2004).

This ICT security definition shares similarities to that of information security. However, the additional characteristics of non-repudiation, accountability, authentication, and reliability can be best labelled as services which should be made available through protected information measures. Through the definitions of information security and ICT security it is clear that there is a difference amongst protecting information resources and protecting ICT resources (von Solms and van Niekerk, 2013).

As stated by (Whitman and Mattord, 2011) the first three characteristics of confidentiality, integrity and availability produce what is known as the CIA triangle. This model has been regarded as the industry standard for computer security. In relation to the additional characteristics, these were added to undertake the

supplementary security needs of an organisation within today's inter-networked business environment. In relation to both information security and ICT security, it is important for a clear understanding of the previously identified characteristics and/or services. Without the seven acknowledged characteristics relating to information measures, information cannot be considered protected. Furthermore, these seven characteristics as well as the information properties of accuracy, authenticity, utility and possession play an important part within information security and ought to considered correspondingly significant (Whitman and Mattord, 2011).

Through the analysing of ICT security numerous threats are targeting related vulnerabilities that can have a negative effect on ICT infrastructure. Here, the technological infrastructure is the asset which requires safeguarding, whereas in ICT security, ICT is described as the asset which is protected. In information security, ICT is recognised as the infrastructure which stores, processes and transmits the information. In this circumstance, information is regarded as the asset which requires safeguarding. In this context, ICT can be classified as a vulnerability which is targeted through numerous threats for the intention of compromising the asset (information). Consequently, in the domain of information security it is the information is recognised as the asset to be protected (von Solms and van Niekerk, 2013).

### 2.2.4 Information and ICT Security to Cyber Security

In cyber security the assets which are to be safeguarded can include anyone or anything which can be reached through cyber space. Here cyber security is connected but not comparable to information security where the information and ICT are identified as the principal reason of a vulnerability. The most important feature of cyber security is that all assets which should be safeguarded are protected from vulnerabilities which exist because of ICT which constructs a foundation of cyberspace. As information security extends ICT security, cyber security is best to be viewed as an extension of information security. The goal and definition of cyber security can be described as the safeguarding of cyber space itself, electronical information, ICTs which assist cyber space, users of cyber space within their personal, national and social position (comprising either tangible or intangible interests) that are vulnerable to attacks which originate within cyberspace. Furthermore, cyber security is regarded as being more comprehensive than information and/or ICT security of which it incorporates. (von Solms and van Niekerk, 2013).

## 2.3 Vulnerabilities

Within cyber security a vulnerability can be defined as a *"weakness in the design, implementation, operation or internal control of a process that could expose the system to adverse threats from threat events"* (Information Systems Audit and Control Association (ISACA), 2016). Similar to the definition acknowledged above the International Telecommunication Union describes a vulnerability as the objective of a threat actor and/or threat source which frequently emerges into an attack mainly due to exploit weakness present within security controls (International Telecommunication Union (ITU) and Wamala (CISSP), 2012).

## 2.4 Threat

A threat can be defined as *"any circumstance or event with the potential to adversely impact organisational operations (including mission, functions, image, or reputation), organisational assets, individuals, other organisations, or the Nation through an information system via unauthorised access, destruction, disclosure, modification of information, and/or denial of service"* (National Institute of Standards and Technology (NIST) and Kissel, 2013; National Institute of Standards and Technology (NIST), 2012; Johnson et al., 2016). Additionally, a threat can also cover the *"potential for a threat source to successfully exploit a particular system vulnerability"* (National Institute of Standards and Technology (NIST) and Kissel, 2013).

According to the ITU cyber threats can be described as the possible violation of security assets. In addition, these threats can be distinguished as Accidental or Intentional and Active or Passive. Accidental threats arise without any premediated intention. An example of this type of threat comprises of software or system failures. However, intentional threats are a result of intentional actions against an asset's protection. This sort of threat can range from the inspection of computer networks using monitoring tools, to complex attacks by using superior knowledge of a computer system. In addition, intentional threats transpire to develop into attacks. In relation to the next category of threats, active threats are the result of an alteration to either the condition or operation of a system, for example the modification of information and

the damage of tangible assets. On the other hand, passive threats do not include a modification in the condition of an asset. The goal of these type of threats is to collect data from a system without disturbing the system's services. Examples of conventional passive threats comprise of: wiretapping, deep packet analysis, inspections or eavesdropping. Furthermore, passive threats develop into passive attacks (International Telecommunication Union (ITU) and Wamala (CISSP), 2012).

### 2.4.1 Threat Event

A threat event can be described as any form of event where a threat actor undertakes some form of destructive action which has the possibility to cause unwanted damage to an asset (Information Systems Audit and Control Association (ISACA), 2016). Threat events are initiated through a threat source. A threat event can be distinguished as both physical and cyber-attacks through the use of tactics, techniques and procedures (TTPs) which are used by a threat actor or threat actors (National Institute of Standards and Technology (NIST), 2012).

### 2.4.2 Threat Source

A threat source interchangeable with the term threat actor is both the purpose and technique used for the deliberate exploit of a vulnerability. It can also refer to a circumstance and purpose which could unintentionally generate a vulnerability (National Institute of Standards and Technology (NIST), 2012).

According to the NIST Special Publication 800-30 threat sources can come in a number of different categories which include: tangible and malicious cyber-attacks, human error of both omission (where a person or object has been omitted) or commission (an instruction, command or role which has been given to either an individual or a group), operational failures of vital organisational regulated assets (which can include software, hardware and environmental controls) and hazards, accidents, natural or artificial disasters and other misfortunes that are outside an organisation's jurisdiction (National Institute of Standards and Technology (NIST), 2012).

Additionally, a threat source is an any object where its purpose is to gain unauthorised access to sensitive information or the security controls of a physical asset or assets. Its definitive goal is to profit from the breach itself. There are a number of various sources from where threats can originate, these may include: hacktivists,

organised crime groups, foreign intelligence services, disaffected employees, investigative journalists and extremist organisations (International Telecommunication Union (ITU) and Wamala (CISSP), 2012) and cyber-attacks through large scale attacks, terrorists, foreign nations, organised crime or political activism against information and communication technology (ICT) systems (Klimburg, 2012).

### 2.4.3 Threat Actor

A threat actor also referred to as a threat agent is an object which executes an attack or to exploit a vulnerability incidence. Threat actors can range from several diverse categories. According to NIST's Special Publication 800-150, these can be *"individuals, autonomous attackers to well-resourced groups operating in a coordinal manner as part of a criminal enterprise or on behalf of a nation-state"* (Johnson et al., 2016). The nature and motive of a threat actor can be one of purpose, drive and intent using numerous TTPs. Through the use of these different TTPs and the urgency of the threat actor their motive is to compromise computer systems, interrupt key services (for example, through a Denial of Service (DoS) attacks) and the disclosure or theft of sensitive data (Johnson et al., 2016).

## 2.5 Attack

An attack can be described as an *"attempt to gain unauthorised access to system services, resources, or information, or an attempt to compromise system integrity"* (National Institute of Standards and Technology (NIST) and Kissel, 2013). Furthermore, an attack can also be specified as either an intrusion or an exploit. In this circumstance, an attack can be represented as an aggressive violation to a system in which originates from the deliberate breach of a vulnerability (Bertino, Martino, Paci and Squicciarini, 2010). However, when it comes to an attack within the cyber domain the characteristics of this general attack definition becomes more focused to what is known as a cyber-attack.

As specified in the NIST Special Publication 800-30 a cyber-attack is an attack that occurs through cyberspace which specifically targets an organisation's cyberspace usage. Here, cyber space can be described as a worldwide domain which is situated

within the information environment. This information environment consists of a symbiotic network of information systems infrastructures which comprise of the Internet, telecommunications networks, computer systems along with controllers and embedded processors. The goal for such an attack is to disorganise, restrict, damage or to maliciously govern a computing infrastructure along with the corruption to the integrity of information or the theft of such information (National Institute of Standards and Technology (NIST), 2012).

Cyber-attacks materialise when a threat violates security controls throughout a tangible or information asset which can be classified by origin and condition as passive and active attacks and inside and outside attacks (International Telecommunication Union (ITU) and Wamala (CISSP), 2012).

### 2.5.1 Passive and Active Attacks

A passive attack is an attack which does not alter systems or information. A more technical definition states that a passive attack is an attack that is against an authentication protocol where the individual committing the attack (attacker) captures information which is being traversed along a network, amid both the claimant and the verifier. However, the attacker does not modify the information (National Institute of Standards and Technology (NIST) and Kissel, 2013). The overall intention of a passive attack is to acquire information, but not to amend the original information. These kinds of attacks are difficult to detect as they do not modify the information (Ahmad, Vivekananda and Pradesh, 2011).

Conversely, an active attack is a type of an attack which can modify information or a system. In more technical terms an active attack is an attack on the authentication protocol in which the individual committing the attack (attacker) sends information to either the claimant, credential service provider, verifier or the relying party (National Institute of Standards and Technology (NIST) and Kissel, 2013). These types of attacks are extremely complicated and are really difficult to avert (Ahmad, Vivekananda and Pradesh, 2011).

### 2.5.2 Inside and Outside Attacks

Cyber-attacks within the security domain can also be categorised as inside and outside attacks which can be executed by either an ‘insider’ or an ‘outsider’ of an organisation. The Request for Comments (RFC): 4949 describes an inside attack as an

*"attack initiated by an entity inside the security perimeter (also known as an 'insider'), i.e., an entity that is authorised to access system resources but uses them in a way not approved by those who granted the authorisation"* (Shirey, 2007). These type of attacks are initiated by prohibited or unlawful individuals from within the security boundary (International Telecommunication Union (ITU) and Wamala (CISSP), 2012). Possible insiders who commits such attacks may consist of inadequately trained, resentful, malevolent, careless, untruthful or dismissed employees (International Organisaton for Standardisation (ISO) and International Electrotechnical Commission (IEC), 2011).

An outside attack is a type of attack which is *"initiated from outside the perimeter, by an unauthorised or illegitimate user of the systems (also known as an 'outsider')"* (Shirey, 2007). These kind of attacks are difficult to protect against since offenders exploit the access privileges acquired for the purpose of genuine organisational purposes (International Telecommunication Union (ITU) and Wamala (CISSP), 2012). Possible outsiders who commit such attacks may comprise of hackers, crackers, computer criminals, cyber terrorists along with intelligence organisations, foreign governments through cyber espionage (International Organisaton for Standardisation (ISO) and International Electrotechnical Commission (IEC), 2011).

## 2.6 Risk

According to the NIST Special Publication 800-30 for information security a risk can be described as a *"measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of: (a) the adverse impacts that would arise if the circumstance or event occurs; and (b) the likelihood of occurrence"* (National Institute of Standards and Technology (NIST), 2012).

However, relating to the security of information and information systems a risk has a somewhat different meaning. In this context, a risk may occur by means of the *"loss of confidentiality, integrity, or availability of information or information systems considering impacts to organisational operations and assets, individuals, other organisations, and the Nation"* (National Institute of Standards and Technology (NIST), 2012). In addition, the NIST Special Publication 800-60 outlines a risk as the degree of impact it can have towards the operations and assets of an organisation and

alternative organisations, entities such as people, or the Nation. This is the outcome of the process of an information system given both the possible impact a threat can have and also the likelihood of a threat taking place (Stine et al., 2008).

For a risk to exist and cause a threat to an organisation's assets, a vulnerability must first be present in which can be exploitable (International Organisaton for Standardisation (ISO) and International Electrotechnical Commission (IEC), 2004). It is identified throughout the various definitions that for a risk to arise two key characteristics need to exist. These are identified as the impact level if a certain incident should arise along with likelihood of an incident taking place.

### 2.6.1 Risk Impact

The impact associated with a risk relates to the scale of damage which are the subsequent ramifications of the exposure of sensitive information that is not sanctioned, the unapproved alteration of sensitive information, the unapproved eradication of sensitive information, the misplacement of sensitive information or the accessibility of information systems (Stine et al., 2008). According to the NIST Special Publication 800-30 (National Institute of Standards and Technology (NIST), 2012) and the NIST Special Publication 800-34 (Swanson et al., 2010) the level and value of an impact can be categorised as low, moderate and high. For the possible risk towards information systems these categories represent the severity level of a possible impact towards an information system which may be compromised. However, for the possible risk towards information these categories represent the measured possible impact which is the resulting compromise of confidentiality, integrity and availability of any category of information (National Institute of Standards and Technology (NIST) and Kissel, 2013).

### 2.6.2 Risk Likelihood

The likelihood associated with a risk as described in ISO / IEC 27005: 2011 signifies the possibility of an incident or circumstance taking place. When it comes to the management of risks which may affect an organisation the term likelihood represents *"the chance of somethings happening, whether it be defined, measured or determined objectively or subjectively, qualitatively or quantitatively, and described using general terms or mathematically (either through a probability or frequency*

*specified over a period of time)"* (International Organisaton for Standardisation (ISO) and International Electrotechnical Commission (IEC), 2011).

Within the likelihood of a risk it identifies a term known as the 'likelihood of occurrence'. The likelihood of occurrence is a commonly used terminology for the management and analysis of information security risks. It is a weighted risk factor which is grounded on an examination that the prospect of a certain threat has the ability of exploiting a vulnerability (or collection of vulnerabilities). This probability risk factor is the projected combination of both the likelihood that the treat will emerge, along with an approximation of the impact's probability. For example, the probability that the threat will be the result of undesirable impacts. The likelihood of impact concentrates on the or possibility that a threat event will result in an destructive impact, irrespective of the degree of harm which may be projected (National Institute of Standards and Technology (NIST), 2012).

## 2.7 Summary

The purpose of this chapter was to provide an overview of security within the cyber domain. The primary focus was on the various fields of security in relation to information and assets within an organisation. Domains such as Information Security and Information and Communication Technology were identified and how these cyber domains can be incorporated to form a field known as Cyber Security. An overview of Cyber Security was presented in this chapter. Various terminologies within security were recognised with focus on the cyber domain. Events such as a vulnerability, threat, attack and risk were described outlining the characteristics required to achieve a type of cyber event. These events can be interconnected for causing serious damage to the confidentiality, integrity and availability of information within an organisation.

In the next chapter, an overview of the National Vulnerability Dataset is presented.

# Chapter 3 National Vulnerability Database

## 3.1 Introduction

The National Vulnerability Database (NVD) is a product belonging to the National Institute of Standards and Technology (NIST) Computer Security Division and is also supported through the Department of Homeland Security's (DHS) National Cyber Security Division (National Institute of Standards and Technology (NIST), 2016).

Section 3.2 of this chapter will present an overview of the National Vulnerability Database (NVD). Section 3.3 will outline the components which construct the NVD. Section 3.4 will discuss the use of the National Vulnerability Database within research.

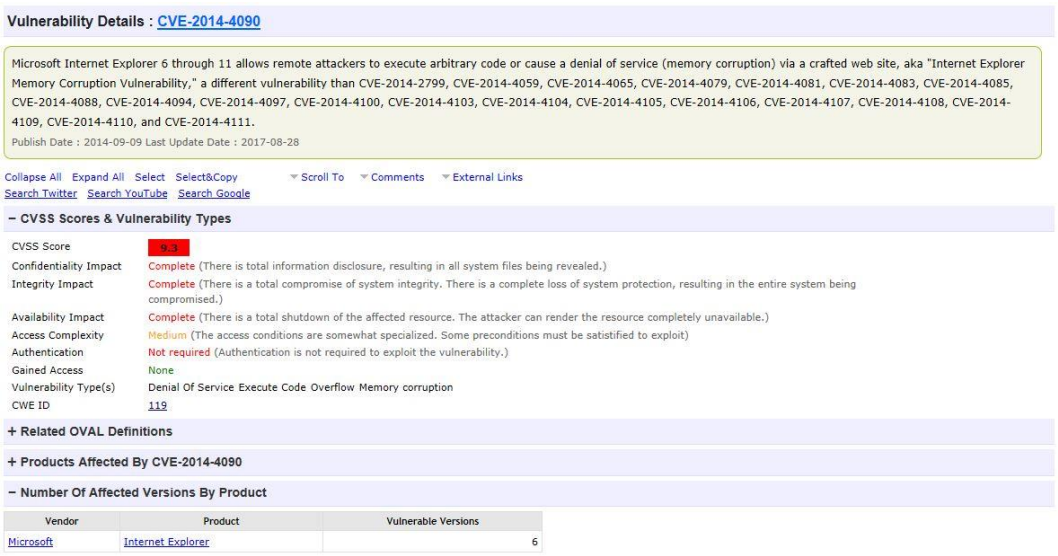
## 3.2 National Vulnerability Database Overview

The National Vulnerability Database (NVD) is known as an all-inclusive Cyber Security Vulnerability Database which incorporates openly accessible U.S. government vulnerability resources, along with providing references to business resources. In relation to this information, the National Vulnerability Database offers access to this information through numerous forms which include: a highly-detailed web search capability and data feeds such as XML, RSS and web service (National Institute of Standards and Technology (NIST), 2016).

Shown below are two examples of an entry which is located within the National Vulnerability Database (NVD). The first example, Figure 3.1 represents the NVD and CVE entry *"CVE-2014-4090"* in Extensible Markup Language (XML) language format which has not been modified for this example.


**Figure 3.1** XML representation of NVD / CVE entry *"CVE-2014-4090"*

Displayed below in Figure 3.2, is a graphical representation of the NVD and CVE entry “CVE-2014-4090”. Unlike the raw XML information of Figure 3.1, the data presented below was obtained through the source entitled ‘CVE Details’. This source can be described as a permitted CVE security vulnerability database or information source, providing a web interface to CVE vulnerability information or data (CVE Details, 2018).



**Vulnerability Details : CVE-2014-4090**

Microsoft Internet Explorer 6 through 11 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2014-2799, CVE-2014-4059, CVE-2014-4065, CVE-2014-4079, CVE-2014-4081, CVE-2014-4083, CVE-2014-4085, CVE-2014-4088, CVE-2014-4094, CVE-2014-4097, CVE-2014-4100, CVE-2014-4103, CVE-2014-4104, CVE-2014-4105, CVE-2014-4106, CVE-2014-4107, CVE-2014-4108, CVE-2014-4109, CVE-2014-4110, and CVE-2014-4111.

Publish Date : 2014-09-09 Last Update Date : 2017-08-28

Collapse All Expand All Select Select&Copy Scroll To Comments External Links
Search Twitter Search YouTube Search Google

**– CVSS Scores & Vulnerability Types**

| | |
|---|---|
| CVSS Score | 9.3 |
| Confidentiality Impact | Complete (There is total information disclosure, resulting in all system files being revealed.) |
| Integrity Impact | Complete (There is a total compromise of system integrity. There is a complete loss of system protection, resulting in the entire system being compromised.) |
| Availability Impact | Complete (There is a total shutdown of the affected resource. The attacker can render the resource completely unavailable.) |
| Access Complexity | Medium (The access conditions are somewhat specialized. Some preconditions must be satisfied to exploit) |
| Authentication | Not required (Authentication is not required to exploit the vulnerability.) |
| Gained Access | None |
| Vulnerability Type(s) | Denial Of Service Execute Code Overflow Memory corruption |
| CWE ID | 119 |

**+ Related OVAL Definitions**

**+ Products Affected By CVE-2014-4090**

**– Number Of Affected Versions By Product**

| Vendor | Product | Vulnerable Versions |
|---|---|---|
| Microsoft | Internet Explorer | 6 |

**Figure 3.2** Graphical representation of NVD / CVE entry *"CVE-2014-4090"* through CVE Details

*Source: Adapted from https://www.cvedetails.com/cve/CVE-2014-4090/?q=cve-2014-4090*

## 3.3 Components of the National Vulnerability Database

The National Vulnerability Database (NVD) is built on the Common Vulnerabilities and Exposures (CVE), a dictionary for vulnerability standards. The NVD also makes available for use the Common Vulnerabilities Scoring Systems (CVSS), relating to all Common Vulnerabilities and Exposures (CVE) vulnerabilities. The National Vulnerability Database (NVD) also emphasises the vulnerability's relevance by means of the Common Platform Enumeration (CPE) (National Institute of Standards and Technology (NIST), 2016). Additionally, the National Vulnerability Database (NVD) also makes available for use the Common Weakness Enumeration (CWE) which represents that a vulnerability has an associated category (Murtaza, Khreich, Hamou-Lhadj and Bener, 2016).

### 3.3.1 Common Vulnerability and Exposures (CVE)

Individual vulnerabilities within the National Vulnerability Database (NVD) is allocated an exclusive identifier which is denoted as a Common Vulnerability and Exposures (CVE) identifier (CVE ID) (Murtaza et al., 2016). The Common Vulnerabilities and Exposures (CVE) is a catalogue or dictionary of communal terminologies for publicly recognised cybersecurity vulnerabilities (CVE Community, 2017a). These CVE identifiers are also specified to as 'CVEs', 'CVE IDs', 'CVE Names', 'CVE Numbers' and 'CVE Entries' (CVE Community, 2017b).

Vulnerabilities are recognised through their unique identifier (CVE ID), in this instance, each individual CVE identifier comprises of the following: (i) a CVE Identifier number with four or more digits within the series numeric segment of the ID itself (for example, 'CVE-2017-0001', 'CVE-2017-12345', 'CVE-2017-1234567'); (ii) a short description of the security vulnerability or exposure; and (iii) any relevant sources (that is, vulnerability reports and announcements) (CVE Community, 2017b). The aim of Common Vulnerabilities and Exposures (CVE) is to alleviate the distribution of information across independent vulnerability capabilities (repositories, services and tools) through this communal inventory (CVE Community, 2017c; Bhuddtham and Watanapongse, 2016).

It is important to note, unaided, the Common Vulnerabilities and Exposures (CVE) does not comprise of or contain a resolution, a vendor's technical specifications or level of impact since this knowledge can be located at several vulnerability security sources, for instance the National Vulnerability Database (NVD). This repository aids security teams by offering resolutions and additional recommendations intended for identifiers located on the list of Common Vulnerabilities and Exposures (CVEs) (Bhuddtham and Watanapongse, 2016).

Information contained within every individual Common Vulnerability and Exposures (CVE) identifier (CVE ID), incorporates but not restricted to influenced or affected software application, product and subordinate applications, the many diverse varieties of the application, the impact of the exploit of a vulnerability, the description of the vulnerability and a vulnerability's score which is determined through using a consistent and uniform vulnerability scoring technique known as the Common Vulnerability Scoring System or CVSS (Murtaza et al., 2016).

### 3.3.2 Common Vulnerability Scoring System (CVSS)

The Common Vulnerability Scoring Systems (CVSS) is observed as a requirement for the recording of key components of vulnerabilities, in addition to calculating the possible effect of the exploitation of a vulnerability (Scarfone and Mell, 2009; Mell, Scarfone and Romanosky, 2007). The incentive for establishing the Common Vulnerability Scoring System (CVSS) was to make available uniform knowledge intended for organisations to use designed for prioritising or ranking vulnerability mitigation (Scarfone and Mell, 2009).

The Common Vulnerability Scoring System (CVSS) supports security personnel and / or security teams by prioritising vulnerabilities by means of supplying a metric for the severity of a vulnerability (Frühwirth and Männistö, 2009; Mell, Scarfone and Romanosky, 2007). Here, the Common Vulnerability Scoring System (CVSS) allocates every respective vulnerability with a numeric score which ranges between 0 being the lowest to 10 being the highest, where the greater the score indicates a higher severity (Frühwirth and Männistö, 2009).

As previously identified, the Common Vulnerability Scoring Systems (CVSS) is a series of metrics used to quantitatively equate and outline diverse vulnerabilities, in relation to several different characteristics (Holm, Ekstedt and Andersson, 2012). It originated as an open framework which comprises of three main metric groups: Base, Temporal and Environmental (Frühwirth and Männistö, 2009).

#### 3.3.2.1 CVSS Metric Score Breakdown

The Base metric stands as the principal Common Vulnerability Scoring System (CVSS) metric or score. Its purpose is to portray a vulnerability's significance and how strenuous it is to apply or exploit (Holm, Ekstedt and Andersson, 2012). In other words, the Base metric signifies both the important and essential vulnerability features which are persistent over a period of time, along with the user's settings. Its intention is to both describe and transmit the important and essential vulnerability components. By taking this objective approach to the describing of vulnerabilities offers users with a transparent and inherent description of a vulnerability (Mell, Scarfone and Romanosky, 2007).

The Temporal metric considers assessments like the presence of a patch for a vulnerability, or for the existence of an exploit located in the wild. The Environmental metric observes additional assessments which are adapted or tailored around a distinct system implementation. However, of the three CVSS metrics only the Base score has been recognised through best practices and standards as the metric to depend on for management of vulnerabilities. Furthermore, the Base score is the only frequently reported metric located within public datasets and vulnerability bulletins (Allodi and Massacci, 2014).

The Base score is calculated as a result of multiplying two further sub metrics: Impact and Exploitability. Consequently, the CVSS Base metric ($CVSS_b$) takes the following formula, shown in Equation 3.1 below closely resembles the conventional or long-established definition of risk as *Impact* × *Likelihood.* The Impact sub metric can be described as an estimation of the impact the exploitation of a vulnerability has towards software and hardware. The Exploitability sub metric is determined through factors, for instance, the level of difficulty in relation to the exploitation and reachability (for example, from the network or through local admission only). Therefore, this metric is occasionally portrayed as an estimate towards the *"likelihood of an exploit"* (Allodi and Massacci, 2014).

$$CVSS_b = Impact \times Exploitability \qquad \textit{Equation 3.1}$$

*Source: Adapted from (Allodi and Massacci, 2014)*

As acknowledged by (Mell, Scarfone and Romanosky, 2007), the Base metric recognises 'fundamental' features of a vulnerability which are persistent over time and through user environments. Incorporated within this Base metric are six sub metrics which include: Access Vector, Access Complexity and Authentication; along with three impact metrics: Confidentiality, Integrity and Availability.

The first three metrics records how the vulnerability is retrieved, and if additional conditions are essential for its exploitation. The final three metrics determine if a vulnerability is exploited, the immediate influence it will have on an Information Technology (I.T.) asset. These impacts are individually characterised in relation to the extent of loss towards confidentiality, integrity and availability (Mell, Scarfone and Romanosky, 2007).

### 3.3.2.2 Impact and Exploitability Sub Metrics

The Impact and Exploitability sub metrics which make up the CVSS Base score are calculated on the foundation of the additional features described in Tables 3.1 and 3.2. As identified in the above paragraph, the Impact sub metric comprises of three distinct assessment: Confidentiality, Integrity and Availability. Each feature in the Impact sub metric can consist of three values: Complete, Partial and None. These values are described in greater detail within this section. The Exploitability sub metric also comprises of three separate assessments: Access Vector, Access Complexity and Authentication (Allodi and Massacci, 2014). Unlike the three values of the Impact sub metric, this sub metric comprises of distinct values depending on the assessment. These are identified in Tables 3.1 and 3.2 and are described in greater detail in the following sections.

**Table 3.1** Possible Values for the Impact Sub Metric

| ***Impact Sub Metric*** | | |
|---|---|---|
| **Confidentiality** | **Integrity** | **Availability** |
| None | None | None |
| Partial | Partial | Partial |
| Complete | Complete | Complete |

**Table 3.2** Possible Values for the Exploitability Sub Metric

| ***Exploitability Sub Metric*** | | |
|---|---|---|
| **Access Vector** | **Access Complexity** | **Authentication** |
| Local | High | Multiple |
| Adjacent Network | Medium | Single |
| Network | Low | None |

#### 3.3.2.3 Access Vector, Access Complexity and Authentication Exploitability Metrics

The Access Vector metric expresses how a vulnerability can be exploited. In relation to this metric, the more remote an attacker can be to attack a host, the higher the vulnerability score. This metric has three possible scoring evaluation outcomes: Local, Adjacent Network and Network (Mell, Scarfone and Romanosky, 2007; Toloudis, Spanos and Angelis, 2016; Scarfone and Mell, 2009). Local, signifies that a vulnerability exploitable through ***local access*** only, involves the assailant (attacker) to acquire physical entry into a vulnerable system or a local account. Adjacent Network, indicates that a vulnerability exploitable by ***adjacent network access***, needs the assailant (attacker) to have either access to the broadcast or collision domain of the software in which is vulnerable. Network means that a vulnerability exploitable by means of ***network access*** represents that the vulnerable software is destined to the network [stack] and the assailant (attacker) does not need either local network access or local access. This form of vulnerability is often referred to as being 'remotely exploitable', meaning a vulnerability can be exploited externally (outside) rather than internally (inside) (Mell, Scarfone and Romanosky, 2007).

The Access Complexity metric evaluates the attack complexity required to exploit a vulnerability once access to the target system has been achieved by the assailant (attacker). In relation to this metric, the lower the complexity, the more superior the vulnerability score is. This metric has three possible scoring evaluation outcomes which comprises of values: High, Medium and Low (Mell, Scarfone and Romanosky, 2007; Toloudis, Spanos and Angelis, 2016; Scarfone and Mell, 2009). ***High*** signifies that specific access conditions or settings are present. For instance: the vulnerable system configuration is rarely seen in practice. ***Medium*** indicates that the access conditions or settings are to some extent specific. For example: little information or data needs to be acquired before an effective attack can be initiate, and the assailants (attackers) are restricted to either a set of users or systems by a form of authorisation, perhaps untrusted authorisation. ***Low*** represents that the specific access conditions or settings or the mitigating conditions or settings are not present or may not exist. For instance: an attack can be manually achieved or executed and involves limited ability or the further accumulation of information or data (Mell, Scarfone and Romanosky, 2007).

The Authentication metric evaluates how frequently an assailant (attacker) needs to confirm or verify towards a target for the exploitation of a vulnerability. Notably, the Authentication metric does not measure either the strength or the intricacy of the authentication operation, but that an assailant (attacker) is compulsory in supplying authorisations or credentials before a breach arises. This metric has three possible scoring evaluation outcomes which comprises of values: None, Single and Multiple (Mell, Scarfone and Romanosky, 2007; Toloudis, Spanos and Angelis, 2016; Scarfone and Mell, 2009). ***None*** signifies that authentication or verification is not essential to both access and exploit a vulnerability. ***Single*** means that one occurrence of authentication is needed to both access and exploit a vulnerability. ***Multiple*** indicates that the exploitation of a vulnerability needs the assailant (attacker) to authenticate two time or more, although identical authorisations or credentials are used on every occasion (Mell, Scarfone and Romanosky, 2007).

#### 3.3.2.4 Confidentiality, Integrity and Availability Impact Metrics

The Confidentiality Impact metric evaluates the impact on confidentiality, in relation to a vulnerability which has been successfully exploited. In this circumstance, confidentiality involves both the restriction towards the access of information and acknowledgement to authorised personnel only. It also represents the prohibiting of access by or acknowledgement to unauthorised personnel. Important to note is that the increased impact on confidentiality boosts the vulnerability score. This metric has three possible scoring evaluation outcomes which comprises of the following values: None, Partial and Complete (Mell, Scarfone and Romanosky, 2007; Toloudis, Spanos and Angelis, 2016; Scarfone and Mell, 2009; Zhang, Ou and Caragea, 2015). ***None*** indicates that no impact has occurred towards the system's confidentiality. ***Partial*** signifies that there is significant leak of information. Here, access to numerous system files is likely, however the assailant (attacker) has no authority of what is accessible, or the extent of information loss is controlled. ***Complete*** represents that the exposure of information is thorough which results in the disclosing of all files belonging to a system. In this circumstance, the assailant (attacker) can view all the information within a system; for example: files, memory et cetera (Mell, Scarfone and Romanosky, 2007).

The Integrity Impact metric determines the impact on integrity, relating to a vulnerability which has been successfully exploited. Here, integrity represents both the certain reliability and consistency towards information. Important to note is that the increased impact on integrity enhances the vulnerability score. This metric has three possible scoring evaluation outcomes which comprises of the following values: None, Partial and Complete (Mell, Scarfone and Romanosky, 2007; Toloudis, Spanos and Angelis, 2016; Scarfone and Mell, 2009; Zhang, Ou and Caragea, 2015). ***None*** signifies that no impact has occurred towards the system's integrity. ***Partial*** represents that the alteration of information or files located within a system is feasible. However, the assailant (attacker) has no regulation over what information or files may be changed, or the range of what the assailant (attacker) can achieve is restricted. ***Complete*** means that that integrity of a system has been entirely compromised. In this situation, the defence of a system has been fully lost which can result in the whole system being put in jeopardy. Here, the assailant (attacker) can change any information or files within the intended system(s) (Mell, Scarfone and Romanosky, 2007).

The Availability Impact metric gauges the impact on availability, relating to a vulnerability which has been successfully exploited. The availability applies to the obtainability of information sources. In this situation, attacks which exhaust disc storage, network bandwidth or processor cycles can all effect a system's availability impact. Furthermore, the increased impact on availability enhances the vulnerability score. This metric has three possible scoring evaluation outcomes which comprises of the following values: None, Partial and Complete (Mell, Scarfone and Romanosky, 2007; Toloudis, Spanos and Angelis, 2016; Scarfone and Mell, 2009; Zhang, Ou and Caragea, 2015). ***None*** indicates that no impact has occurred towards the system's availability. ***Partial*** signifies that there is decreased efficiency or disruption in the availability of assets. ***Complete*** represents an entire cessation of the affected asset(s). Here, the assailant (attacker) has the ability to make the asset or assets totally inaccessible (Mell, Scarfone and Romanosky, 2007).

### 3.3.3 Common Weakness Enumeration (CWE)

The Common Weakness Enumeration or CWE is recognised as a 'community-developed' and publicly available list dictionary of shared software security vulnerabilities. It operates as both a common language and benchmark aimed at software security mechanisms. Additional, it can function as a standard intended for the identification, mitigation and prevention of a vulnerability (CWE Community, 2017).

As previously identified, a vulnerability has an associated type and is assigned an identifier, known as a Common Weakness Enumeration identifier or a CWE ID. The Common Weakness Enumeration (CWE) represents an established ordered or ranked list of categories of vulnerabilities which have been created by security specialists. Every Common Vulnerabilities and Exposures identifier (CVE ID) consist of a Common Weakness Enumeration identifier (CWE ID). Generally, the Common Weakness Enumeration (CWE) is encompassed with a Common Vulnerabilities and Exposures identifier (CVE ID) of a vulnerability (Murtaza et al., 2016).

In total, there are approximately 1,000 diverse CWE identifiers or numbers, however only a small number of distinct CWE identifiers are used within the National Vulnerability Database (Recorded Future, 2015). Due to the large quantity of identifiers, the list of CWEs located within the NVD are catalogued in Appendix A.

### 3.3.4 Common Platform Enumeration (CPE)

The Common Platform Enumeration (CPE) can be defined as a *"structured naming scheme for information technology systems, software and packages"* (National Institute of Standards and Technology (NIST), 2018). The CPE is a division of the Security Content Automation Protocol, or SCAP which is a collection of technical requirements that are maintained by the U.S. Government to encourage automation and standardisation regarding information security (Information-Technology Promotion Agency Japan, 2008). Additionally, the SCAP specification was proposed by the National Institute of Standards and Technology (NIST), to which was implemented into the National Vulnerability Database (NVD) (Sanguino and Uetz, 2017).

As previous stated, the Common Platform Enumeration is a naming standard for the identification of information system platforms which include software applications, hardware and operating systems. The use of this CPE naming scheme can allow software and hardware vendors, users, systems administrators and security personnel to categorise Information Technology platforms which contain vulnerabilities by means of a common language. In addition, the application of the Common Platform Enumeration can also be beneficial towards the management of assets (Information-Technology Promotion Agency Japan, 2008).

At present, two versions of the Common Platform Enumeration definition exist: CPE 2.2 and CPE 2.3. It is version 2.3 of the Common Platform Enumeration which outlines an assemblage [stack] comprising of five definitions, consisting of the CPE Naming Standard and the CPE Dictionary Standard (Sanguino and Uetz, 2017; The MITRE Corporation, 2013).

#### 3.3.4.1 CPE Naming Standard

The Common Platform Enumeration naming standard is characterised through a collection of attributes known as the *Well-Formed CPE Name (WFN).* These CPE naming attributes are as follows: *part, vendor, product, version, update, edition, language, sw_edition, target_sw, target_hw* and *other*. An example of a WFN is shown in Figure 3.3 below (Sanguino and Uetz, 2017; The MITRE Corporation, 2013).

```
part:a, vendor:microsoft, product:internet_explorer, version:8,
update:NA, edition:ANY, language:ANY, sw_edition:ANY,
target_sw:ANY, target_hw:ANY, other:ANY
```

**Figure 3.3** WFN for Microsoft Internet Explorer 8

*Source: Adapted from (Sanguino and Uetz, 2017)*

As shown above in Figure 3.3, the value for the attribute *part: 'a'* specifies the Information Technology (IT) asset is an *application*, where application is denoted through the character *'a'*. The value signified by *'NA'* indicates that attribute is not being used or not applicable, and this value is allocated to attributes to which have no significance towards a software application, hardware or operating system. Lastly, the value *'ANY'* represents that no limitations towards an attribute (Sanguino and Uetz, 2017; The MITRE Corporation, 2013).

The purpose behind the Common Platform Enumeration is to allocate an identification schema towards assets within an IT organization or infrastructure. At present, the CPE supports two standards*: Uniform Resource Identifier (URI)* and *Formatted String*, where the URI is stated in CPE version 2.2 and the Formatted String is defined in CPE version 2.3. A URI or Formatted String identifier is produced from the Well-Formed Name of an Information Technology asset, and this procedure can be identified through the terms: Uniform Resource Identifier binding or Format String binding (Sanguino and Uetz, 2017; The MITRE Corporation, 2013). Examples of both URI and Format String binding are shown below in Figures 3.4 and 3.5.

```
cpe:/a:microsoft:internet_explorer:8:-
```

**Figure 3.4** WFN Bound to a Uniform Resource Identifier (URI)

*Source: Adapted from (Sanguino and Uetz, 2017)*

```
cpe:2.3:a:microsoft:internet_explorer:8:-:*:*:*:*:*:*
```

**Figure 3.5** WFN Bound to a Format String

*Source: Adapted from (Sanguino and Uetz, 2017)*

#### 3.3.4.2 Common Platform Enumeration (CPE) Naming Structure

A Common Platform Enumeration (CPE) Name distinctively classifies Information Technology (IT) platforms, for instance: software applications, hardware and operating systems, and comprises of two features. Firstly, a CPE Name categorises IT platforms, if it encompasses in its name that it is software application, hardware or operating system. Secondly, a CPE Name is achieved through the amalgamation of the *vendor* name and *product* name (Information-Technology Promotion Agency Japan, 2008).

#### 3.3.4.3 Common Platform Enumeration (CPE) Naming Structure Components

The basic CPE Naming convention is as follows:

*cpe:/{part}:{vendor}:{product}:{version}:{update}:{edition}:{language}*

The Part element of the Common Platform Enumeration Name comprises of a solitary character letter value which represents the specific platform. These platforms

consist of: ***a*** for software application, ***h*** for hardware and ***o*** for operating system (Information-Technology Promotion Agency Japan, 2008; Buttner and Ziring, 2008).

The Vendor element of the CPE Name contains the vendor of the platform. Here, the name which is used for the vendor element of the CPE Name should also be the domain name of the vendor. However, if the domain name is dissimilar to that of the vendor's name, the domain name should be used for the vendor element of the CPE Name (Information-Technology Promotion Agency Japan, 2008; Buttner and Ziring, 2008).

The Product element of the CPE Name consists the name of the product in relation to the platform. In the circumstance that the product names and descriptions are multi-worded, full spell of the product muse be used. It is also significant that blank spaces are replaced with underscores (Information-Technology Promotion Agency Japan, 2008; Buttner and Ziring, 2008).

The Version element of the Common Platform Enumeration (CPE) Name is the version of the platform. For the naming of the version component, it is important that the version is written similarly to that of the product element. In this circumstance, the delimiter should be same as how it is done for the product, through the use of periods, dashes et cetera (Information-Technology Promotion Agency Japan, 2008; Buttner and Ziring, 2008).

The Update element belonging to the CPE Name comprises of the update or service pack information of the platform or product. In this instance, the difference between both the update and service pack is subject to how the vendors and products classify this information. Typically, products are originally released with no update or service pack information attached to them. Although, if the information explicitly encompassed through the vendor for the product's initial primary release, then this product information should be included within the update element of the CPE Name (Information-Technology Promotion Agency Japan, 2008; Buttner and Ziring, 2008).

The Edition element of the CPE Name comprises of the edition relating of the platform. The purpose of this edition element is to indicate explicit target software and hardware architectures. For example, if the edition of a certain software or hardware is a free edition or a professional edition (Information-Technology Promotion Agency Japan, 2008; Buttner and Ziring, 2008).

The Language element of the basic structure of the CPE Name signifies the language related with a specified platform. It is significantly important that the value of this element must be in accordance with a valid language code which is determined through *IETF RFC 4646: Tags for Identifying Languages* (Information-Technology Promotion Agency Japan, 2008; A. Phillips and M. Davis, 2006; Buttner and Ziring, 2008).

#### 3.3.4.4 Common Platform Enumeration (CPE) Dictionary

In addition to the Common Platform Enumeration (CPE) Naming Structure as identified above, the CPE stack also incorporates the CPE Dictionary standard. This CPE Dictionary outlines the structure of a repository which comprises of CPE identifiers for categories of Information Technology (IT) platforms or products. Every entry within the CPE Dictionary encompasses the bound formation of a product's Well-Formed Name (WFN), where the bound formation is of the format: Uniform Resource Identifier (URI) or Formatted String (Sanguino and Uetz, 2017; The MITRE Corporation, 2013).

The National Vulnerability Database (NVD) both hosts and preserves the authorized Common Platform Enumeration (CPE) Dictionary which is accessible in XML format (Sanguino and Uetz, 2017). An example of CPE 2.3 entry for Microsoft Internet Explorer 8 contained within the CPE Dictionary is shown below in Figure 3.6. Furthermore, as identified by (Sanguino and Uetz, 2017), the CPE 2.3 entry, as shown in Figure 3.6 comprises of the Uniform Resource Identifier (URI) and Formatted String for the identification of Microsoft Internet Explorer 8 software application.

```
<cpe-item name="cpe:/a:microsoft:ie:8">
  <title xml:lang="en-US">Microsoft Internet Explorer 8</title>
  <cpe-23:cpe23-item name="cpe:2.3:a:microsoft:ie:8:*:*:*:*:*:*:*"/>
</cpe-item>
```

**Figure 3.6** CPE 2.3 Entry for Microsoft Internet Explorer 8 (CPE Dictionary)

*Source: Adapted from (Sanguino and Uetz, 2017)*

## 3.4 The Use of the National Vulnerability Database within Research

The National Vulnerability Database (NVD) has been used in several research studies over the years. To commence, with explicit reference to the National Vulnerability Database itself, (Frei, May, Fiedler and Plattner, 2006) put to use the data or information in the National Vulnerability Database and additional comparable repositories, with the purpose of measuring the variance amongst the detection time of vulnerabilities; the acknowledgement time of attacks; along with the availability time of patches to these vulnerabilities. In this study, an extensive examination of over 14,000 vulnerabilities with the aid of over 80,000 security recommendations or warnings which were published between the years of 1996 to 2006 was undertaken, aimed at investigating security vulnerability's life-cycle. (Frei et al., 2006) also recognised that zero-day vulnerabilities, in relation to their attacks or exploits are instantly accessible within the National Vulnerability Database on the date of the vulnerability's acknowledgement. However, in this circumstance, software suppliers can be unhurried in providing the relevant software patches to mitigate these vulnerabilities. Furthermore, it was also identified that attacks or exploits of a zero-day vulnerability nature are dangerously on the rise.

(Christey and Martin, 2007) published a report called 'Vulnerability Type Distribution in CVE' which focused on vulnerabilities located within the National Vulnerability Database (NVD). It was identified that several attacks had a dramatic upsurge in the year 2006, with such attacks or exploits being of a web application and PHP remote file inclusion nature. Additionally, it was also identified that in the year 2007, the highest ranked vulnerability was that of a buffer overflow.

(Ahmed, Al-Shaer and Khan, 2008) issued a paper which introduced an innovative security metric framework which classifies and measures empirically the most noteworthy security risk factors. These factors comprised of existing vulnerabilities, past vulnerability trend of the remotely available services, forecast of possible vulnerabilities aimed at an established network service and projected severity, along with policy resistance to address circulation in a network. Subsequently, the authors experiments were described using factual six-year vulnerability information

from the NVD, to illustrate the confidence and high accuracy of the metrics used within this study.

(Houmb and Franqueira, 2009) presented a paper which measured the level of risk regarding vulnerabilities, through analysing both the frequency and impact towards the vulnerabilities located within the National Vulnerability Database (NVD), known as a ToE risk level estimation model. Here, a ToE can represent any element of either a system or network or the entire system or network and is used to signify the object that requires being controlled. The ToE risk level estimation model uses the Common Vulnerability Scoring System (CVSS) metric, located within the National Vulnerability Database (NVD) to calculate both the misuse impact and the misuse frequency of vulnerabilities, specifically the resulting severity following an attack or attacks through vulnerability exploitation. From both the misuse impact and misuse frequency the ToE risk level was obtained. Additionally (Houmb and Franqueira, 2009) made use of a Markov model for the purpose of calculating or forecasting a vulnerabilities risk level for a certain period of time.

(Schryen and Rich, 2010), published a paper where an empirical study was performed using comprehensive vulnerability information along with vulnerability patch information. The results shown through this study indicates that it is not the specific software development technique which governs the severity of both the vulnerabilities and vendor's patching conduct, instead the precise application type and policy of specific development institution correspondingly.

(Huang, Tang, Zhang and Tian, 2010) issued a paper in solving the issue of taxonomies covered in software vulnerabilities, through a technique of vulnerability classification constructed through the text classification of NVD information. In this research numerous clustering algorithms were used which included Simplekmean, BisectingKMeans and BatchSom and evaluated by means of the Cluster Overlap Index. Subsequently, 45 key vulnerability clusters were identified and chosen from around 40,000 vulnerabilities in relation to Descriptor Dominance Index. It is with these dominant vulnerability taxonomies which became the focal point for this study.

(Neuhaus and Zimmermann, 2010) published a paper which used the Common Vulnerability and Exposures (CVE) vulnerability information or data up as far as the year 2009, in order to the sort vulnerabilities into their diverse categories. In this paper,

the authors contended that there is significant quantity of Common Weakness Enumeration (CWE) vulnerability categories, in the region of 700 which makes it inconceivable to human. In addition, the authors also clarified that the National Vulnerability Database (NVD) successfully categorises vulnerabilities by means of a small number of Common Weakness Enumeration (CWE) vulnerability categories. They investigated the Common Vulnerability and Exposures (CVE) taken from the National Vulnerability Database (NVD) by means of using topic models by their description, presented in text format. In this instance, this 'topic model' uses an unsupervised machine learning technique known as Latent Dirichlet Allocation (LCA) on the description of Common Vulnerabilities and Exposures (CVE) entries with the aim of developing a purpose-built classification system. Their topic model allows for the identification of dominant topics, along with emerging trends through a technique that is automated.

(Wang et al., 2010) published a research paper which focused on the measurement of vulnerability's similarity, to estimate diverse vulnerabilities through collection of measures. The technique used in this study was grounded on the structural hierarchy of vulnerabilities, where the similarity was defined using recognised mathematical models. For this research, the NVD and Ontology of Vulnerability Management supplied the data essential for similarity measurement, where this calculation may be used in various regions of vulnerability management.

Similar to the report published by (Christey and Martin, 2007) in 2007, (Barlowe, Blackbird and Davis, 2012) and (Symantec, 2014) published separate reports in which made use of the National Vulnerability Database (NVD) for the purpose of mining the progression of diverse categories of vulnerabilities. The difference between the two reports is that the report issued by (Barlowe, Blackbird and Davis, 2012) reveals the progression of vulnerabilities present in both software and hardware; whereas, the report published by (Symantec, 2014) recognises the dominant Zero-Day vulnerabilities.

(Zhang, Caragea and Ou, 2011; Zhang, Ou and Caragea, 2015) orchestrated investigational research of the implementation of data mining techniques on data or information located within the National Vulnerability Database (NVD), with the purpose of forecasting the 'Time To Next Vulnerability' (TTNV) within a system.

Specifically, the estimation of cyber risks by means of using data or information in the National Vulnerability Database (NVD). The authors analysed with several diverse features, that were created through the accessible information within the National Vulnerability Database (NVD). These main features consisted of the: Published Date Time, Common Platform Enumeration (CPE) and the Common Vulnerability Scoring System (CVSS) which were pre-processed into the following: Published Date Time as 'Month' and 'Day'; Two adjacent Common Platform Enumeration (CPE) vulnerabilities differences (version 1, version 2) as 'Versiondiff'; Common Platform Enumeration (CPE) specification as 'Software Name'; adjacent different Published Date Time as 'TTPV' (Time to Previous Vulnerability); adjacent different Published Date Time as 'TTNV' (Time to Next Vulnerability) and Common Vulnerability Scoring System (CVSS) metrics. Additionally, these features were categorised into Predictive data and Predicted data. Initially, Predictive data consisted of the follow pre-processed feature: Month; Day; Versiondiff; Time to Previous Vulnerability (TTPV) and Common Vulnerability Scoring System (CVSS) Metrics (specify the characteristics of the forecasted vulnerabilities). Whereas, the Predicted data comprised of one pre-processed feature: Time to Next Vulnerability (TTNV) which indicated a vulnerabilities risk-level, i.e. zero-day vulnerabilities. The authors used many Machine Learning Algorithms (Regression and Classification techniques) for the intention of investigating the National Vulnerability Database's (NVD's) analytical influence. The authors results presented that the data or information in the National Vulnerability Database mostly had a weak ability in vulnerability forecasting.

(Ghani et al., 2013) orchestrated research which focused on the quantitative comprehension of security vulnerabilities, in the circumstance of limited vulnerability information. An innovative technique was proposed used for the predictive calculation of security vulnerabilities which considers applicable situations (for example, zero-day vulnerabilities), as there may be little to no information available in carrying out distinctive scoring of a vulnerability. The authors proposed a novel systematic approach known as the Vulnerability Assessment Model (VAM), influenced by means of Linear Discriminant Analysis, and which uses the NVD as a training dataset.

(Last, 2015) issues a paper in forecasting vulnerability detection rates aimed at distinct software products. This was achieved through several different phases, comprising of the construction of forecast models intended for vulnerability rates (at an overall level), and a category level (for example: web browser, operating system). Next, these models were used as a feature within the predictive models for distinct software products. For this research, numerous regression models were employed to historical vulnerability information through the NVD for the discovery of past trends within vulnerability data. Subsequently, the author used k-NN classification combined with numerous time series distance calculations, for choosing suitable regression models for a forecast. Results show which time series distance calculations offer the greatest vulnerability discovery estimates.

(Murtaza et al., 2016) published a paper in which examined the use of past vulnerability patterns with the purpose of forecasting upcoming vulnerabilities within software applications and / or programs. The authors also observed whether the trends of vulnerabilities within software applications had any significant meaning or none at all. Here, the authors used the National Vulnerability Database (NVD) as the primary repository of software application and / or program vulnerabilities. Using the data or information within the National Vulnerability Database, the authors data mined vulnerabilities which covered the years: 2009 to 2014. Their results discovered (after the mining of the NVD) that the order of similar vulnerabilities could appear a significant amount of times in one software product, application and / or program. Additionally, the author's results also showed that the quantity of SQL Injection vulnerabilities has to some extent reduced, within the years of 2009 to 2014; whereas vulnerabilities of a cryptographic description, had seen a significant rise throughout these years. Although, the authors did not discover any statistical importance towards the progression of a vulnerability's appearance during these years. During their study, the authors most compelling discovery was that the successive order of vulnerability related circumstances (events) act in accordance with a first order Markov property. This means that the subsequent vulnerability can be forecasted through the preceding vulnerability, and the subsequent vulnerability does not rely upon the past order of vulnerabilities. Through this, the authors discovered that the subsequent vulnerability can be forecasted by use of the preceding vulnerability where it has a precision rate of approximately 90% and a recall rate of approximately 80%. Furthermore, in relation

to the authors use of the National Vulnerability Database (NVD), they did not examine all the elements within the NVD. Elements which include the vulnerability's description, vulnerability metrics (Common Vulnerability Scoring System (CVSS) base metric: access vector, access complexity, authentication, confidentiality impact, integrity impact and availability impact), vulnerability score (Common Vulnerability Scoring System (CVSS) and so forth. Here, the authors identify that that use of these additional elements can further examine vulnerabilities and the sequence towards vulnerabilities.

## 3.5 Summary

The purpose of this chapter was to present an overview of the National Vulnerability Database (NVD). The components which structure the NVD, for instance: The Common Vulnerability and Exposures (CVE), the Common Vulnerability Scoring System (CVSS), the Common Weakness Enumeration (CWE) and the Common Platform Enumeration (CPE) are described in detail. This chapter also identified the use of the National Vulnerability Database within prior research studies.

In the following chapter an information filtering technology known as a recommender system is presented.

# Chapter 4 Recommender Systems

## 4.1 Introduction

Recommender systems as described by (Ricci, Rokach, Shapira and Kantor, 2011) are software tools and techniques which offer suggestions (recommendations) for items which could be of use or benefit to a user. These recommendations are intended at aiding the user of the system in numerous decision-making procedures which in today's lifestyle may comprise of: what music to listen to (Spotify); what movie, film or TV series to watch (Netflix) or what type of items to purchase (Amazon). Commonly, the term 'Item' signifies what the Recommender System suggests to its user or users (Thorat, Goudar and Barve, 2015).

Section 4.2 of this chapter will describe recommender systems and the various techniques which can be employed. Section 4.3 identifies the type of data used within a recommender system. Section 4.4 will offer an overview of the various categories of recommender system which exist today. Section 4.5 will present an overview of the collaborative filtering, and its benefits over alternative recommender system approaches. Section 4.6 will describe the Memory-Based Collaborative Filtering technique, where an overview of its sub categories: User-Based and Item-Based are presented in Sections 4.7 and 4.8. Section 4.9 presents and overview of the state-of-the-art technique, known as Model-Based Collaborative Filtering. Section 4.10 outlines how diverse recommender system techniques can be evaluated. Section 4.11 concludes this chapter with a summary.

## 4.2 Recommender Systems

Recommender systems are a category of Information Filtering Systems which manages the issue of information overloading through filtering essential information from a large quantity of information, according which relates to the interests, likings or the recognised attitude a user has towards an item (Isinkaye, Folajimi and Ojokoh, 2015).

The term ‘Recommender System’ was first brought to attention with the introduction of the first recommender system in 1992, known as Tapestry (Portugal, Alencar and Cowan, 2015). This system was dependent on clear views of individuals (users) which were from a cohesive community, for example a team (workgroup) situation within an office environment (Sarwar, Karypis, Konstan and Riedl, 2001). Furthermore, Tapestry was identified as the first manual Collaborative Filtering system; a term which is still widely used within recommender systems today (Portugal, Alencar and Cowan, 2015; Ekstrand, Riedl and Konstan, 2011).

These systems are mainly targeted to individuals who lack personal knowledge, or capability to categorise the enormous quantity of possible items from which are offered (Ricci et al., 2011). Additionally, the main objective of a recommender system offers personalised recommendations of items to a system’s user, to which they would be of interest in (Lü et al., 2012).

## 4.3 Data

Data and especially the type of data used in recommender systems is extremely important. Recommender systems locate patterns in data for the recommendation of new items through filtering. These recommendations are then provided to users who are either inexperienced, knowledge deficient or completely overcome with the large quantity of favourable items. Data relates to the items to recommend to users, along with the users who in turn will acquire these recommended items. However, data used by recommender systems can be varied, and the use of this data relies on the type of recommender system techniques used (Ricci et al., 2011; Isinkaye, Folajimi and Ojokoh, 2015).

For the classification of data used by recommender systems, (Ricci et al., 2011) outlines three essential features which comprises of the following: (1) items, (2) users and (3) transaction. The term transaction can be described as the relationship amongst users and items that are present within the recommender system.

### 4.3.1 Items

Firstly, an item can be described as an entity or object which is recommended to a user in a recommender system. An item can be represented through its complexity and its worth or usefulness. The worth or value relating to an item can be either positive or negative; positive if the item is of any use to the user, or negative if the item is not suitable to the user or an incorrect choice was made when selecting the item. (Ricci et al., 2011). An example of an item may comprise of music or a movie that is positively favoured by a specific user, would supply a higher liking compared to music or a movie which does not appeal for a certain user.

### 4.3.2 Users

Secondly, a user can have many different aims and traits. Recommender systems can take advantage of user information with the aim of making recommendation more specific to the user in question and to improve the collaboration between the user and recommender system. Recommender systems may use this information for the recommendation of alternative items to users which were chosen either through users with the same similarities or trust levels (Ricci et al., 2011). This information can be organised in several different ways, and the choice of what information to represent is determined by the type of recommender system technique used (de Moura Del Esposte, Campiolo, Kon and Batista, 2016). Comparable to the item example, a user may favour certain items over others. Through this, a user will receive personalised recommendations based on the traits of the user and similar users within the system.

### 4.3.3 Transactions

The third and final feature, transaction; applies to the interaction between both a user and the recommender system itself. Transactions are represented as ‘log-like data’ which collects and accumulates significant information produced through the interaction between human and recommender system. These transactions are valuable for all recommender system algorithms as they use this transactional data which has been collected to recommend new items to the user (Ricci et al., 2011). In addition, the collection of transactions express how items can be valued. In this circumstance, recommender systems use information accumulated through numerous transactions to produce new recommendations to a user (de Moura Del Esposte et al., 2016).

### 4.3.4 Ratings

One important category of transactional data are ratings which can be accumulated implicitly or explicitly. For implicit ratings, the intentions of the recommender system are to try and assume the opinions of the user, built on the behaviours of the user (Ricci et al., 2011). Examples of implicit feedback may comprise of Internet browsing or item purchase history (Hu, Koren and Volinsky, 2008).

For explicit ratings, number of diverse forms that rating can take, these consist of scaler, ordinal, binary and unary. Firstly, Scaler ratings can comprise of numerical ratings which may come in the form of 1-5-star rating or ordinal ratings; which may come in the form of strongly agree, agree, neutral, disagree, strongly agree. Secondly, binary ratings are identical to classification or 1/0 where the rating may be a choice of good/bad, agree/disagree or yes/no. Lastly, unary ratings may signify that a user has viewed or purchased a specific item or positively rated the item. However, the nonappearance of a certain rating may signify, that there is no data or information connecting the user to the item; and that the item was purchased by the user in a different location. Furthermore, another form of explicit rating is tagging, whereby tags connected to a user relates to an items in which is offered by the recommender system (Schafer, Frankowski, Herlocker and Sen, 2007).

## 4.4 Categories of Recommender Systems

As identified within literature, Recommender Systems can be divided into three main categories, as shown in Figure 4.1 below: Collaborative Filtering, Content-Based Filtering and Hybrid Filtering techniques (Isinkaye, Folajimi and Ojokoh, 2015; Zhang et al., 2014; Lü et al., 2012; Portugal, Alencar and Cowan, 2015). Content-Based Filtering techniques establish predictions through the information of its users, while also disregarding the opinions of user, as is with Collaborative Filtering (Isinkaye, Folajimi and Ojokoh, 2015). Collaborative Filtering is a recommendation technique which suggests items through recognising alternative users, who share similar interests which uses the opinion of the similar users to recommend similar items to the active or target user (Sharma, Gopalani and Meena, 2017). Hybrid Filtering however is a combination of more than one recommendation technique

(Thorat, Goudar and Barve, 2015). This recommendation technique can be an amalgamation of Collaborative Filtering with Content-Based Filtering, or through the merger of various Collaborative Filtering techniques (Lü et al., 2012). These three recommender system techniques are illustrated below in Figure 4.1.

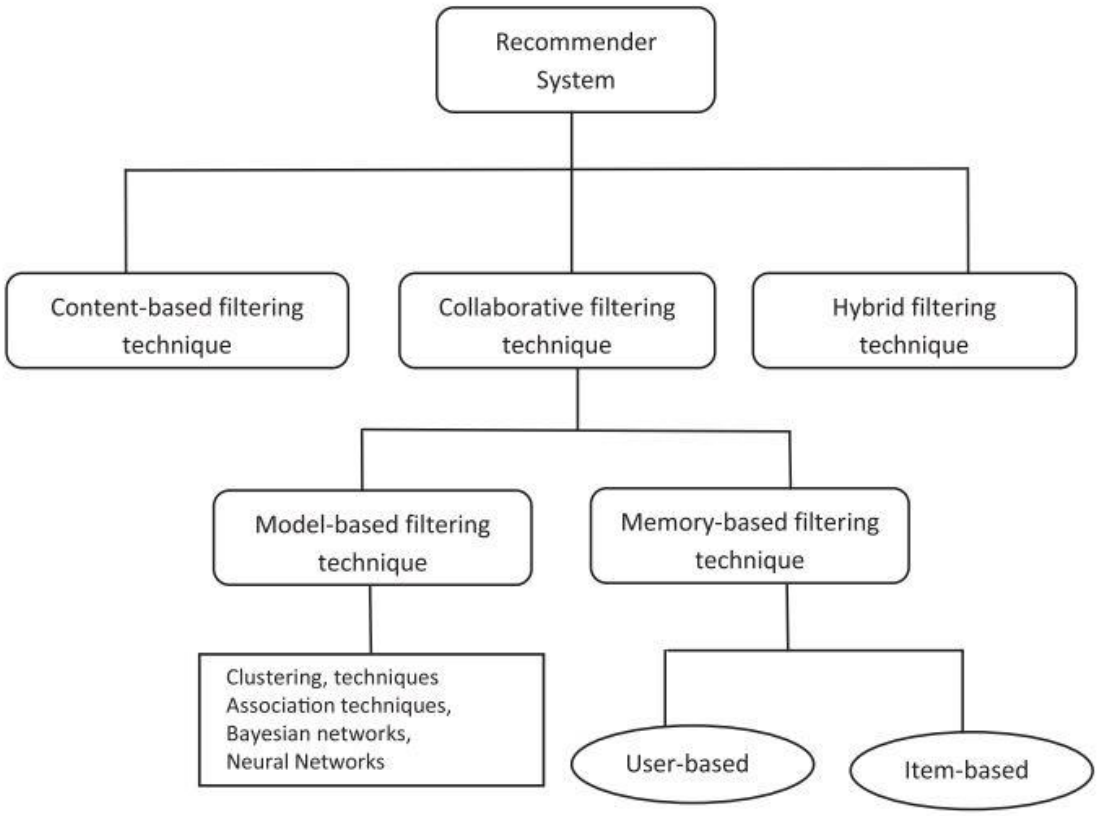



**Figure 4.1** Recommender System Techniques

*Source: Adapted from (Isinkaye, Folajimi and Ojokoh, 2015)*

### 4.4.1 Content-Based Filtering in contrast with Collaborative Filtering

The primary assessments towards information filtering were first grounded on content (Foltz and Dumais, 1992). These types of systems choose which items to recommend which are built on their content. Consequently, the user profile (where each user has an associated profile which comprises of the subset of items which have been rated, and the equivalent rating for each item) is a portrayal of the content that the user is interested in. This category of filtering is particularly successful when recovering or obtaining documents, such as text, where every document is characterised through a group of keywords. However, this category of filtering

systems suffers from several issues detailed as follows (Cacheda, Carneiro, Fernández and Formoso, 2011; Shardanand and Maes, 1995).

Firstly, items should be scrutinised by means of a machine. However, this is problematic when obtaining information such as multimedia where the machine observation of the content (for example, colours and textures) varies significantly through the perception of the user. The annotation of characteristics or attributes by a user partially resolves this issue, as Content-Based filtering is unsatisfactory to deal with a large amount of information made accessible these days. An additional significant issue with Content-Based filtering is its incapability to evaluate the superiority of an item. For instance, this category of filtering system cannot differentiate, in relation to text documents, a good article compared to a bad article, if both articles use an identical set of words. The superiority towards an item is an extremely independent or subjective characteristic which hinges on the ideas, culture and tastes of every person. Because of this, it is difficult for a machine to evaluate. Lastly, Content-Based filtering is deficient in discovering unforeseen items which a user may find of interest. Specifically, appropriate or suitable items which are not evidently associated to the user profile (Cacheda et al., 2011).

Collaborative Filtering techniques however are less sensitive to the issues described for Content-Based techniques. This is because they are not grounded on the content of items, but instead the user's opinion or judgement. Here, this technique will recommend items to a user which have acquired high ratings through alternative users with comparable interests or likings. Through these techniques, items are in fact rated by individuals (Shardanand and Maes, 1995). Therefore, techniques with use Collaborative Filtering are not required to analyse content (meaning it is valid for any sort of item (including nonannotated multimedia content), also the superiority evaluation towards items is also assessed (Cacheda et al., 2011).

Through systems which use Collaborative Filtering techniques, the user profile is the collection of ratings that is appointed towards diverse items. The ratings of a user are kept in a table identified as a rating matrix. This rating matrix is processed to produce recommendations. Conditional on what way the rating matrix data is managed, two categories of Collaborative Filtering techniques of Memory-Based and Model-Based can be distinguished (Cacheda et al., 2011). Furthermore, as identified

in literature, Collaborative Filtering is recognised as the most widely used technique (Sharma, Gopalani and Meena, 2017; Thorat, Goudar and Barve, 2015; Gogna and Majumdar, 2015; Zhang et al., 2014; Sarwar et al., 2001). Therefore, the next section examines Collaborative Filtering in detail.

## 4.5 Collaborative Filtering

Collaborative Filtering is category of recommender system technique which constructs both its predictions and recommendations through the prior ratings and preferences of alternative and comparable users present within the recommender system (Ekstrand, Riedl and Konstan, 2011).

The core concept of Collaborative Filtering techniques is to offer the prediction or recommendation of items based on the opinion of similar users which can be obtained either through explicit or implicit user feedback (Sarwar et al., 2001). In relation to this feedback, (Gogna and Majumdar, 2015) states that explicit feedback can be presented in the form of ratings (i.e. rating score of an item on scale between 1 to 5), whereas implicit feedback can be conditional through a user's behaviour and / or actions (i.e. historical patterns or purchase records).

Collaborate Filtering operates by constructing a user-item rating matrix of favoured (recommended) items of a user. Next it identifies similar users who share common interests and preferences by means of calculating the similarities amongst the targeted user to produce recommendations. These similar users create what is known as a *neighbourhood*, and a user of the system acquires recommendations to particular items which they have not previous rated, but were previously rated (positively) through users within this neighbourhood (Isinkaye, Folajimi and Ojokoh, 2015). A more detailed description of the Collaborative Filtering process is described in the next section.

### 4.5.1 Collaborative Filtering Process Overview

Figure 4.2 displays the process of Collaborative Filtering. This technique operates through a list of *m* users $U = \{U_1, U_2, U_3 \ldots U_m\}$ along with a list of *n* items $I = \{I_1, I_2, I_3 \ldots I_n\}$. Here, *n* signifies that each user has a separate list of items. Every user $u_i$ receives a list of items $I_{ui}$ where the user has given their point of view (assessment),

where a user's assessment can be given either explicitly or implicitly (Sarwar et al., 2001).

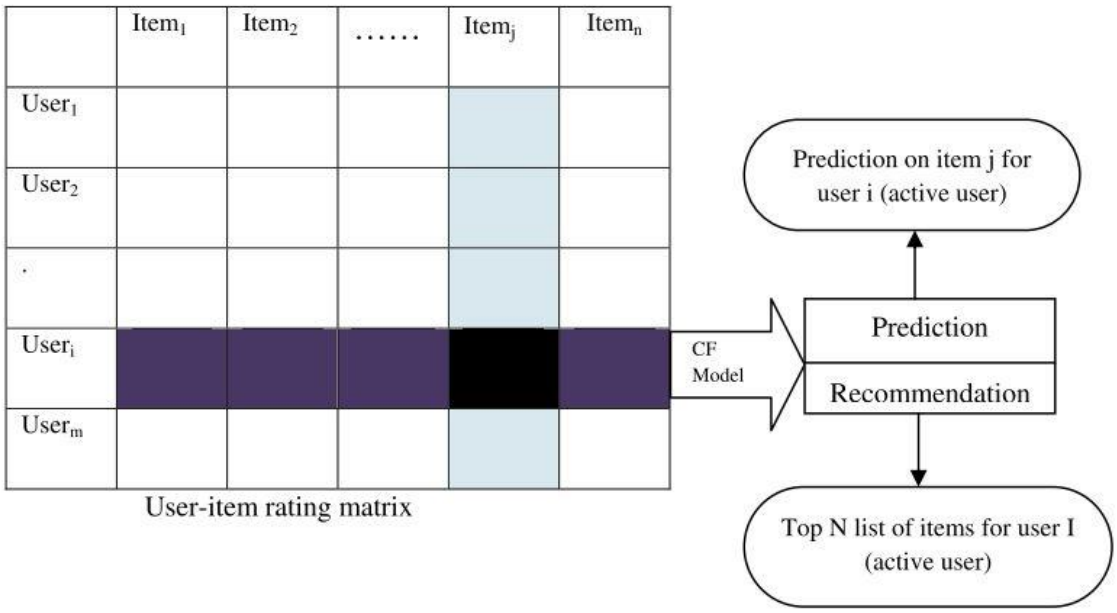



**Figure 4.2** The Collaborative Filtering Process

*Source: Adapted from (Isinkaye, Folajimi and Ojokoh, 2015)*

Figure 4.2 identifies an *active user* of the system $User_i \in U$, whom exists within the system. Collaborative Filtering aims to classify item similarity which can occur in two methods: *Prediction* and *Recommendation.* Initially, **Prediction** is represented as a numerical value that signifies the predicted similarity of item *j* for the active user *i*. Here the predicted numerical value is within a similar range (1 to 5) as the assessment value given by the active user $User_i$. Alternatively, **Recommendation** is characterised as a list of recommended *N* items for which the active user $User_i$ will highly favour. This method of Collaborative Filtering is known as *Top-N recommendation.* Additionally, it is also important to note that in relation to the Top-N list of recommendations is that items must not have previously acquired by the active user (Sarwar et al., 2001).

Furthermore, as identified by (Sarwar et al., 2001), the Collaborative Filtering Process shown in Figure 4.2 characterises the whole user-item information $(User_m \, x \, Item_n)$ as a *Rating Matrix* which is represented as *A.* Here, every entrant within the rating matrix incorporates the desired rating (score) of the *i*th user on the *j*th item. Every respective rating is within a numerical range, additionally this rating can also

be 0 which signifies that a user has not supplied an assessment (rating score) for an item.

As recognised quite extensively within literature (Isinkaye, Folajimi and Ojokoh, 2015; Adomavicius, Manouselis and Kwon, 2015; Sarwar et al., 2001; Lü et al., 2012), Collaborative Filtering Techniques can be divided into two main categories: *Memory-Based* Collaborative Filtering and *Model-Based* Collaborative Filtering.

## 4.6 Memory-Based Collaborative Filtering

Memory-Based Collaborative Filtering, occasionally referred to as Neighbourhood-Based, is a technique which essentially learns the user-item rating matrix and distributes predictions or recommendations grounded on the correlation between the active user and item, along with the remainder of the user-item rating matrix (Lee, Sun and Lebanon, 2012). Memory-Based Collaborative Filtering techniques can be described as *"recommended objects are those that were preferred by users who share similar preferences as the target user, or, those that are similar to the other objects preferred by the target user"* (Lü et al., 2012). In this scenario, the system maintains a memory of past recommendations.

According to (Lee, Sun and Lebanon, 2012; Lü et al., 2012; Schafer et al., 2007) Memory-Based methods are they most widely used Collaborative Filtering techniques. This technique predicts ratings through the suggestion of users who rated similarly to the active user, or else through items that were rated similarly to the targeted item.

Here, it is presumed that is two users share similar ratings on several items that these users will also have comparable ratings on the outstanding items. Conversely, if two items share similar ratings that are rated by a percentage of users, then the two items will receive comparable ratings by the outstanding users (Lee, Sun and Lebanon, 2012).

Similar to Collaborative Filtering techniques being divided into both Memory-Based and Model-Based methods, Memory-Based Collaborative Filtering can be further categorised into two techniques: User-Based and Item-Based Collaborative Filtering (Isinkaye, Folajimi and Ojokoh, 2015; Sarwar et al., 2001; Melville and

Sindhwani, 2010; Lee, Sun and Lebanon, 2012; Ekstrand, Riedl and Konstan, 2011; Sharma, Gopalani and Meena, 2017).

## 4.7 User-Based Collaborative Filtering

User-Based Collaborative Filtering, also called *User-User k-NN Collaborative Filtering* is a technique that identifies alternative users whose prior ratings are similar to the current (active) user, it then uses their ratings against alternative items to predict or recommend to the current (active) user of the system (Ekstrand, Riedl and Konstan, 2011).

User-Based Collaborative Filtering searches for alternative users who share a strong similarity in relation to the items that they have both rated. Here the indicated users' ratings for the item are *weighted* through their closeness of the current (active) user's ratings for the predicting or recommending the current (active) user's preference for an item (Ekstrand, Riedl and Konstan, 2011).

### 4.7.1 Prediction

In relation to Prediction, this technique operates through a three-stage procedure which can be described as follows: First, compute the similarity between the current (active) user and the remainder of users present in the system. Secondly, choose a subgroup of the users which corresponds to their similarity with the current (active) user of the system. The subgroup of users can also be identified as a neighbourhood of users. Lastly, calculate the prediction score by means of using the neighbourhood subgroup ratings (Cacheda et al., 2011).

### 4.7.2 Recommendation

User-Based Collaborative Filtering techniques can also be used for recommendation, typically to compute the Top-N recommended items for a user. Comparable to the prediction approach, the Top-N approach operates through a three-stage procedure which can be described as follows: First, the number of users (*k*) within the user-item-rating matrix that share similarity to the active user are identified. Secondly, after this collection of *k* similar users have been discovered, the union of the items that are rated and / or purchased through these users are computed and attach an accompanying weight to each item grounded on its significance within the

collection. Lastly, through the union in step two, the User-Based technique selects and then recommends the Top-N items which have the highest weight and not previously rated and / or purchased by the active user (Deshpande and Karypis, 2004).

It is observed through (Deshpande and Karypis, 2004), in relation to the User-Based prediction and recommendation, that the approach used to establish the similar users (*k*) and the method used to determine the significance of the diverse items represent the important stages for the overall performance of this technique. The similarity amongst the users is calculated by means of considering them as vectors within the item space and calculating the similarity using a similarity measure, for instance Pearson's Correlation Coefficient or Cosine Vector-Based similarity. However, the significance of each item is established by how often the item was rated and / or purchased through the majority of *k* similar users.

In addition to the Rating Matrix, as shown in Figure 4.2, the User-Based Collaborative Filtering technique requires a similarity function or measure which calculates the similarity amongst two users, along with a procedure for using both the similarities and ratings for the creation of predictions or recommendations for a user (Ekstrand, Riedl and Konstan, 2011).

### 4.7.3 Similarity Computation Between Users for Prediction or Recommendation

To calculate the similarity between the active (current) user and the remainder of users, there are numerous categories of similarity measures which can be used (Cacheda et al., 2011). As identified within literature, the most common of these similarity measures are: ***Pearson's Correlation Coefficient*** and ***Cosine*** (Isinkaye, Folajimi and Ojokoh, 2015; Zhang et al., 2014; Lee, Sun and Lebanon, 2012; Sánchez, Serradilla, Martínez and Bobadilla, 2008).

#### 4.7.3.1 Pearson's Correlation Coefficient Similarity

Pearson Correlation Coefficient is a similarity measure which calculates the *linear relationship between two vectors* (Sharma, Gopalani and Meena, 2017). In other words, this similarity measure *computes the statistical correlation between two user's common ratings to determine their similarity* (Ekstrand, Riedl and Konstan, 2011). The Pearson Correlation Coefficient is not only the first presented similarity measure, it is also identified as being one of the most popular (Cacheda et al., 2011).

$$userSim(u,n) = \frac{\sum_{i \subset CR_{u,n}} (r_{ui} - \bar{r}_u)(r_{ni} - \bar{r}_n)}{\sqrt{\sum_{i \subset CR_{u,n}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \subset CR_{u,n}} (r_{ni} - \bar{r}_n)^2}} \qquad \textit{Equation 4.1}$$

*Source: Adapted from (Schafer et al., 2007)*

The equation for Pearson's Correlation Coefficient, as shown above in Equation 4.1 is calculated by means of comparing the ratings for all items which are rated or scored by both the active (current) user and the neighbour (co-rated items that are in the same neighbourhood as the active user). Here, the user is signified through *u*, the neighbour is represented by *n* and $CR_{u,n}$ symbolises the collection of co-rated items amongst the user *u* and the neighbour *n*. Additionally, Pearson's Correlation Coefficient varies from 1.0 which signifies users with a perfect opinion or preference to -1.0 which signifies users with an imperfect opinion or preference. It is accepted that negative correlations are not considered to be valuable towards the increasing of predictive accuracy (Schafer et al., 2007).

#### 4.7.3.2 Cosine Similarity

Cosine similarity or vector similarity uses a *vector-space* technique that is based on linear algebra instead of a statistical approach (Isinkaye, Folajimi and Ojokoh, 2015). In this similarity measure, users are characterised by means of |*I*|-dimensional vectors, and the similarity between them is calculated through the cosine distance between two rating vectors (Ekstrand, Riedl and Konstan, 2011). In other words, Cosine similarity regards users as *vectors of item ratings* which then calculates the cosine of the angle amongst the vectors of two users. In relation to the similarity measure, a value that is represented as close to 1 signifies the similarity between users, whereas a value which is close to 0 signifies that no similarity is present amongst users (Cacheda et al., 2011).

The Cosine similarity between two users can be represented through the following equation, as shown in Equation 4.2. Additionally, when calculating the cosine similarity, there cannot be negative ratings present and items which have no ratings are considered as having a rating of zero. It has been identified that the Pearson Correlation Coefficient similarity measure usually outperforms the Cosine similarity measure (Melville and Sindhwani, 2010).

$$s(u,v) = \frac{\mathbf{r}_u \cdot \mathbf{r}_v}{\|\mathbf{r}_u\|_2 \|\mathbf{r}_v\|_2} = \frac{\sum_i r_{u,i} r_{v,i}}{\sqrt{\sum_i r_{u,i}^2} \sqrt{\sum_i r_{v,i}^2}}$$

*Equation 4.2*

*Source: Adapted from (Ekstrand, Riedl and Konstan, 2011)*

The User-Based technique has several constraints relating to real-time performance and scalability. Real-Time Top-N recommendations built on an existing group of items, cannot make the most of pre-calculated user-user similarities. While the throughput of User-Based recommendation techniques can be improved by increasing the number of servers operating this category of recommender system approach, it cannot however reduce the latency of every Top-N recommendation. This is essential for near real-time execution. Furthermore, the computational complexity of this Collaborative Filtering technique expands linearly when the number of users significantly increases (Karypis, 2001). Although the user-item matrix is sparse, the user-user similarity matrix tends to be relatively dense. The reason for this is, even a small number of rated and / or purchased items can result in dense user-user similarities. To overcome constraints such as scalability present in User-Based techniques, the Item-Based Collaborative Filtering technique were established (Breese, Heckerman and Kadie, 1998; Kitts, Freed and Vrieze, 2000).

## 4.8 Item-Based Collaborative Filtering

The Item-Based Collaborative Filtering technique is similar to the previously discussed User-Based approach, but rather than searching for neighbours between similar users, Item-Based Collaborative Filtering searches for similarity amongst items (Cacheda et al., 2011).

For the expansion of Collaborative Filtering to a larger user base and to enable or simplify the distribution on numerous of commercial sites and platforms, it was important to develop more scalable techniques, hence Item-Based Collaborative Filtering. This technique is recognised today as a widely used Collaborative Filtering approach (Ekstrand, Riedl and Konstan, 2011).

### 4.8.1 Prediction

In relation to Prediction, Item-Based Collaborative Filtering computes predictions through using the user's individual or personal rating preference for alternative items merged with those items' likenesses towards the intended item, instead of the ratings of alternative user's and their user similarities as is the case with User-Based Collaborative Filtering. The Item-Based Collaborative Filtering (CF) technique operates as follows: instead of using the similarity among the user's ratings for the prediction of item preferences, the Item-Based technique uses the similarity amongst the rating of items (rating patterns towards items). If two items have similar users who like and dislike them, then the items are comparable, and users are predicted to have similar likings towards comparable items (Ekstrand, Riedl and Konstan, 2011).

### 4.8.2 Recommendation

In contrast to the rating prediction, the Item-Based technique can analyse the user-item matrix to discover correlations amongst the diverse items. It uses these associations to compute a Top-N recommendation list. The central motive behind this Top-N approach is that a user will probably supply ratings to items which are like the items that the user has previously rated. Additionally, since the Item-Based Top-N approach is not required to discover the neighbourhood of similar users when a recommendation is sought after, it leads to considerably faster recommender systems (Karypis, 2001).

The Item-Based Top-N recommendation technique uses the item-item similarity to calculate the correlations amongst items. Throughout the building phase, for every item, the *k* most similar items are calculated, and their comparable similarities are captured. Then, for every user which has rated and / or purchased a collection (group of items which have previously been rated and / or purchased through the user for the calculating of Top-N recommendations) of items, this data is used to calculate the Top-N recommended items through the following phases (Karypis, 2001).

Firstly, the collection of possible items to be recommended are identified by capturing the union of the most similar items (*k*) for every item, and then eliminating from the union whichever items are present within the collection. Next, for every item, the similarity to the collection is calculated as the aggregate of the similarities amongst all the items, by applying only the similar items (*k*) to that of each item. Lastly, the items present in the collection are arranged in a non-increasing sequence with regards to that similarity. In addition, the initial *N* items are nominated as the Top-N recommended collection of items for the active user (Karypis, 2001).

### 4.8.3 Similarity Computation Between Items for Prediction or Recommendation

An essential stage of the Item-Based Collaborative Filtering technique is the similarity amongst items and at that point determine the most comparable items. In this circumstance, the essence of calculating the similarity between two items is to: (a) separate the users who have valued (rated) both items and then, (b) implement a similarity measure to calculate the similarity between the two items (Sarwar et al., 2001). This process is shown below in Figure 4.3.

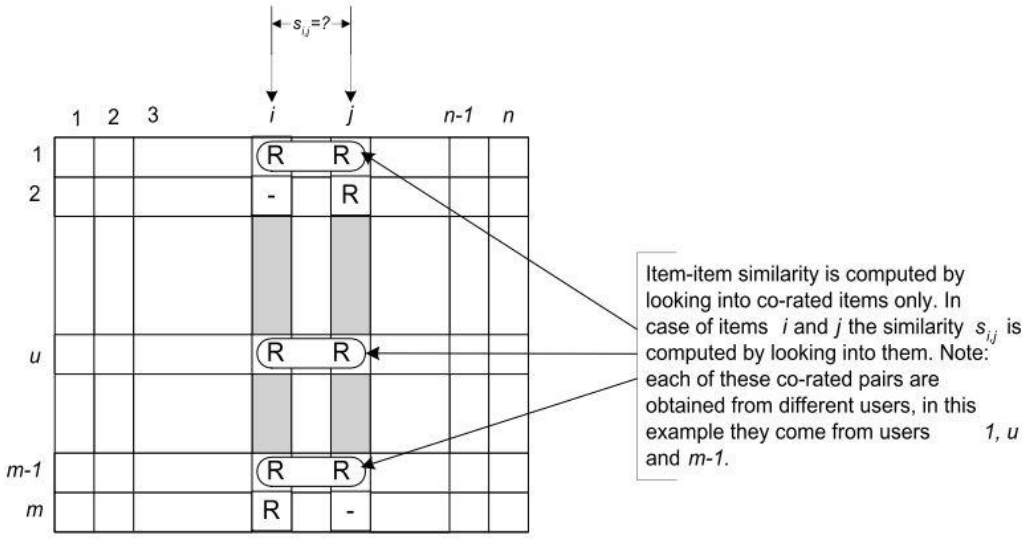



**Figure 4.3** The Separation of the Co-Rated Items and Similarity Computation

*Source: Adapted from (Sarwar et al., 2001)*

Figure 4.3, as seen above demonstrates the procedure of the separation of two items and the process of calculating the similarity of items. In addition, Figure 4.3 displays the rating matrix rows which signifies users and the rating matrix columns which indicates items. Here items are denoted through the characters *i* and *j*, whereas the similarity between these two items is represented through $s_{i,j}$ (Sarwar et al., 2001).

Comparable to the calculation of the similarity of users through User-Based Collaborative Filtering, there are numerous diverse similarity measures which can be employed to compute the similarity of items through an Item-Based Collaborative Filtering technique. As identified through the literature of (Sarwar et al., 2001; Ekstrand, Riedl and Konstan, 2011), two of these similarity measures are: *Correlation-Based Similarity* and *Cosine-Based Similarity*.

These similarity measures were previously discussed in Section 4.7.3 in relation to the User-Based Collaborative Filtering technique. However, for the Item-Based Collaborative Filtering technique, instead of finding the similarity between users, the similarity amongst items are computed through these two similarity measures (Sarwar et al., 2001). In addition, the Pearson's Correlation Coefficient similarity measure does not perform as well as the Cosine-Based similarity measure when used within the Item-Based Collaborative Filtering technique (Ekstrand, Riedl and Konstan, 2011; Sarwar et al., 2001).

### 4.8.4 Advantages and Disadvantages of Memory-Based Collaborative Filtering Technique

In relation to the Memory-Based Collaborative Filtering, (Ricci et al., 2011) identifies several key advantages towards this category of Collaborative Filtering technique which are shown below in Table 4.1 listed below.

**Table 4.1** Key Advantages of Memory-Based Collaborative Filtering

| **Advantage** | **Detailed Explanation** |
|---|---|
| *Simplicity* | Neighbourhood-Based techniques (User-Based and Item-Based) are both instinctive and reasonably simplistic to implement. In their most straightforward configuration, only one parameter (the number of nearest neighbours used within the prediction) demands modification. |
| *Justifiability* | Neighbourhood-Based techniques also make available a concise and instinctive validation towards the calculated predictions. This aids to deliver clarity of the recommendations, henceforth offer more confidence in relation to the recommendations. For instance, in the Item-Based Collaborative Filtering technique, both the group of neighbour items along with the ratings specified by the user for these items can be offered to the user as a reasoning for the recommendation. |
| *Efficiency* | An effective factor of Neighbour-based techniques relates to their efficiency. In contrast to most Model-Based techniques, Neighbourhood-Based techniques require no computationally expensive training stages which requires to be undertaken at regular intervals within considerably large commercial applications. As the stage relating to recommendation is generally more computationally expensive for that of Model-Based techniques, Neighbourhood-Based techniques can be pre-computed within an offline phase. This offers almost immediate recommendations to a user. Additionally, the retaining of the nearest neighbours needs little memory space which makes these techniques scalable towards applications which comprises of millions of users and items. |

| | |
|---|---|
| *Stability* | An additional and useful characteristic relating to Memory-Based Collaborative Filtering techniques is that they are unaffected to a small degree with the addition of users, items and ratings, as is generally recognised within large applications. For example, after the similarity of items have been calculated, an Item-Based technique can quickly construct recommendations to a new user or a group of users, without the need to re-train the Memory-Based technique. Additionally, after the insertion of a small number of ratings for a new item, it is only the similarities between the new item and the alternative items within the Collaborative Filtering technique that needs to be calculated. |

One disadvantage with Memory-Based Collaborative Filtering relates to its inferior *scalability* towards larger applications which use this technique, because the managing of Collaborative Filtering techniques requires organising substantially large amounts of data, that tend to be somewhat inefficient. For instance, major e-commerce suppliers may have a significant quantity of users and items, where the Collaborative Filtering technique employed offers recommended predictions built using the user-item matrix. Memory-Based Collaborative-Filtering techniques also put further emphasis on the recommending the most favoured items, due to a greater amount of rating information being made accessible (Sarwar et al., 2001). Additionally, as acknowledged by (Schafer et al., 2007) nearly all Collaborative Filtering techniques use some pre-processing approach, as this is to lessen the run-time complexity and to assist Memory-Based Collaborative Filtering to improve scalability.

Another related issue to Collaborative Filtering is the *cold start* problem. To provide predictions, Collaborative Filtering techniques require rating information. However, when a new item is added, it lacks rating information. Inherently, this item cannot be recommended to a user until it acquires sufficient ratings. Similarly, the cold start problem also affects new users. Here, Collaborative Filtering techniques cannot produce a user until the target user has supplied plentiful ratings for an item or items. To put it simply, the new user cold start issue occurs when a new user does not occupy a neighbourhood of comparable users (Schafer et al., 2007).

To help lessen the cold start issues present in Collaborative Filtering, (Schafer et al., 2007) identifies the following solutions:

- Have the user of the Collaborative Filtering technique supply a rating for a few primary items before they use the recommender system.
- Present non-personalised recommendations to the user (for instance, list of most favoured items within the recommender system), up until the user has supplied a sufficient quantity of ratings.
- Query the user in describing their likings towards an item (for example, 'I like comedy movies').
- Query the user in supplying demographic information.
- Use the ratings of alternative users who share comparable demographic information as recommendations.

Studies have revealed that Model-Based Collaborative Filtering techniques are more effective than that of Memory-Based Collaborative Filtering techniques, with regards to prediction accuracy (Koren, 2008). However, as identified by (Good et al., 1999), a greater predictive accuracy does not assure users of the recommender system both a satisfying and worthwhile experience. In reality, a significantly important function regarding users of the recommender system is that of unforeseen or unexpected recommendations (Good et al., 1999). For example, if an admirer of a certain category of movie franchise, for instance the Lord of the Rings trilogy, will not become enthusiastic about movie recommendations that relate to Lord of the Rings. Here, unforeseen recommendations assist users of the recommender system discover item(s) of interest which otherwise would have been unnoticed by the user.

## 4.9 Model-Based Collaborative Filtering

To overcome the disadvantages relating to Memory-Based Collaborative Filtering techniques, for example the issue of scalability, Model-Based Collaborative Filtering techniques were established (Sharma, Gopalani and Meena, 2017). As Memory-Based techniques preserves a database of all users' known preferences (ratings) for every all items, and for every prediction, executes a variety of calculations over the whole database of known user ratings for all items. Model-Based techniques first accumulate the users' likings into a descriptive model of users, items and ratings; recommendations are then produced through engaging with the model. Model-Based techniques can provide additional effectiveness beyond its predictive abilities by emphasising certain correlations within the information which provides an inherent justification aimed at recommendations or for making presumptions more unambiguous. Predictions within Model-Based techniques can be computed at a fast speed, once the model is constructed, and it does not need as much performance power compared to Memory-Based Collaborative Filtering techniques. However, the time complexity to accumulate the data or information into a model could be restraining and the addition of a new item could demand a complete recompilation of the model (Pennock, Horvitz, Lawrence and Giles, 2000).

Model-Based Collaborative Filtering techniques use user rating data or information to train models for the prediction of user ratings by means of machine learning or data mining algorithms (Tang and Tong, 2016; Isinkaye, Folajimi and Ojokoh, 2015). This Collaborative Filtering technique provides a parametric model to the training data which can be subsequently used for the prediction of unobserved ratings and distribute recommendations to users (Lee, Sun and Lebanon, 2012). The parameters of the model are calculated offline which uses data or information from the user / item rating matrix (Cacheda et al., 2011). Model-Based Collaborative Filtering techniques construct a model in which acquires knowledge or detects the user-item relations through the influence of low dimensional representations, the user and item feature vectors (Sharma, Gopalani and Meena, 2017). Additionally, Model-Based techniques have the ability to quickly recommend a group of items, this is because they use pre-calculated model and have established to offer recommendations which are comparable to Memory-Based Neighbourhood-Based Collaborative Filtering techniques (Isinkaye, Folajimi and Ojokoh, 2015).

### 4.9.1 Latent Factor Models

This category of Collaborative Filtering techniques is also identified as *Latent Factor Models* which can be recognised as *Matrix Factorisation Models* (Sharma, Gopalani and Meena, 2017). Both Latent Factor and Matrix Factorisation models have emerged as the most modern or state-of-the-art approaches within this category of Collaborative Filtering techniques (Melville and Sindhwani, 2010). The main objective of Latent Factor models is to factorise the user-item rating matrix into low rank user and item factors which signify the preferences of a user and features of an item in a communal latent space, correspondingly. Here, the prediction intended for a user on an item can be computed by means of the dot product relating to the comparable user and items factors (Ning and Karypis, 2011).

Contrary to Memory-Based Neighbourhood-Based techniques which produce recommendations grounded on statistical presumptions of the similarity between users or items, Latent Factor models presume that the similarity amongst both users and items is concurrently influenced through some unobserved lower-dimensional pattern within the data. Additionally, Matrix Factorisation models are recognised as a category of widely effective Latent Factor models (Melville and Sindhwani, 2010).

### 4.9.2 Matrix Factorisation

Matrix Factorisation aims to differentiate both the users and items through their feature vectors of low dimension which are presumed through the rating patterns of a user (Sharma, Gopalani and Meena, 2017). (Zhang et al., 2014; Sharma, Gopalani and Meena, 2017) outlines the general concept behind Matrix Factorisation which is described as follows: For a specific dimension, denoted by the symbol $\boldsymbol{n_f}$, Matrix Factorisation aims to approximate the user-item matrix $\boldsymbol{R}$ as the dot product of two lesser matrices.

$$\mathbf{R} \approx \mathbf{P}\mathbf{Q}^{\mathbf{T}} = \underbrace{\begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \end{bmatrix}}_{n_u \times n_f} \underbrace{\begin{bmatrix} \mathbf{q}_1^{\mathbf{T}} & \mathbf{q}_2^{\mathbf{T}} & \dots \end{bmatrix}}_{n_f \times n_m} \qquad \textit{Equation 4.3}$$

*Source: Adapted from (Zhang et al., 2014)*

As shown above in Equation 4.3, $\boldsymbol{P}$ represents the matrix $n_u \times n_f$ whereas $\boldsymbol{Q}$ signifies the matrix $n_m \times n_f$. Here $\boldsymbol{P}$ is the user factor matrix and $\boldsymbol{Q}$ is the item factor matrix. $\boldsymbol{p_u}$ specifically the u-th row of $\boldsymbol{P}$ denotes a factor vector aimed at the user $u$, also $\boldsymbol{q_i}$ specifically the i-th row of $\boldsymbol{Q}$ symbolises a factor vector for the item $i$. The objective of the Matrix Factorisation algorithm is to acquire knowledge of the $\boldsymbol{P}$ and $\boldsymbol{Q}$ matrices and constructs the dot product as close to the user-item matrix $\boldsymbol{R}$ as possible. Consequently, the prediction score $\hat{r}_{u,\,i}$ for every user and item can be computed through the equation, shown below in Equation 4.4.

$$\hat{r}_{ui} = \mathbf{p_u} \times \mathbf{q_i^T} = \sum_{k=1}^{n_f} p_{u,k} q_{k,i} \qquad \textit{Equation 4.4}$$

*Source: Adapted from (Zhang et al., 2014)*

$\boldsymbol{p_u}$ and $\boldsymbol{q_i}$ represents the user and item feature vector. These feature vectors are acquired through minimising the variance between the actual rating and the predicted rating. The equation shown above in Equation 4.4 can be further expanded through the introduction of regularisation terms or factors to avoid the issue of over-fitting which is shown below in Equation 4.5.

$$(\mathbf{P}^*, \mathbf{Q}^*) = \min_{(\mathbf{P},\mathbf{Q})} \sum_{(u,i)\in\tau} \left(r_{u,i} - \mathbf{p_u q_i^T}\right)^2 + \lambda(\|\mathbf{p_u}\|^2 + \|\mathbf{q_i}\|^2) \qquad \textit{Equation 4.5}$$

*Source: Adapted from (Zhang et al., 2014)*

Presented in Equation 4.5 is the constant parameter $\lambda$ which symbolises the regularisation term or factor used to circumvent over-fitting, where the parameter value generally differs between the values of 0 and 1. The regularisation factor $\lambda$ penalises the square of the Euclidean norm of weights, where this approach also goes by the term *weight decay*. Where $\tau$ represents the training dataset and $\lambda$ signifies the regularisation factor, the equation presented in Equation 4.5 states that inspecting the $\boldsymbol{P}$ and $\boldsymbol{Q}$ matrices on the training dataset $\tau$ can lessen the totality of squared errors. Items which comprise of a high value towards latent factor vector features which are of high regard to the active or target user are then recommended. Furthermore, if the rating data or information is represented in a categorical format, classification Model-Based techniques such as Bayesian, Clustering and Probabilistic Models can be used to represent Collaborative Filtering approaches. In circumstances where the rating data or information is presented as a numerical format, Model-Based techniques such as

Matrix Factorisation, regression and Singular Value Decomposition (SVD) may be used.

### 4.9.3 Relevant Research of Matrix Factorisation Based Models

In recent years, numerous Matrix Factorisation based techniques for the construction of Latent Factor models have been proposed. In 2004, (Hoffman, 2004) implemented the Probabilistic Latent Semantic Analysis (PLSA) used for Collaborative Filtering which has shown to be comparable to non-negative Matrix Factorization. The PLSA technique presents a latent space such that the co-existence of both users and items (for example: a particular user has purchased and / or rated a specific item) can be proclaimed conditionally autonomous.

Similarly, in 2004 and in 2005, (Srebro, Rennie and Jaakkola, 2004) and (Rennie and Srebro, 2005) put forward a Max-Margin Matrix Factorisation technique known as MMMF which necessitates a low-norm factorisation of the user-item matrix that permits unrestrained dimensionality aimed at the latent space. This was employed through lessening the trace-norm of the reassembled user-item matric from the latent factors.

(Pan et al., 2008) and (Hu, Koren and Volinsky, 2008) proposed a Weighted Regularisation Matrix Factorisation technique known as WRMF, that is expressed as a regularised Least-Squares issue. Here, a weighted matrix is used to distinguish the offerings from observed purchase and / or rating activities, along with purchase and / or rating activities which are also unobserved.

Similarly, (Koren, 2008) implemented an intersecting technique which combined the Neighbourhood-Based [Memory-Based] technique and Matrix Factorisation. Through this approach, the similarity of items is concurrently learnt by means of Matrix Factorisation by taking the full advantage of both Collaborative Filtering techniques.

### 4.9.4 Related Work of Matrix Factorisation within a Top-N Recommendation Context

Top-N Collaborative Filtering recommendation has been expressed as a ranking problem, and not a rating prediction issue. In 2009, (Rendle, Freudenthaler, Gantner and Schmidt-Thieme, 2009) proposed a Bayesian Personalised Ranking standard called BPR which is the maximum subsequent estimator from a Bayesian examination, and calculates the dissimilarity amongst the rankings of user purchased and / or rated items along with the remaining items. This BPR technique can be implemented for the Item-Based KNN approach (BPRKNN) and Matrix Factorisation approaches (BPRMF) by way of an overall objective purpose.

In 2010, (Cremonesi, Koren and Turrin, 2010) proposed a straightforward Pure Singular Value Decomposition Based Matrix Factorisation Collaborative Filtering technique, known as PureSVD which defines both users and items by means of the greatest principle singular vectors regarding the user-item matrix.

In 2011, (Ning and Karypis, 2011) established an innovative Top-N recommendation technique known as Sparse Linear Method or SLIM. This approach enhances the conventional Item-Based Collaborative Filtering techniques by means of direct training from the data, a sparse matrix of aggregation coefficients which are comparable to the established item-item similarities. The SLIM technique has shown to achieve superior performance outcomes towards an extensive assortment of datasets, in addition to performing better than alternative state of the art Collaborative Filtering techniques. However, a fundamental constraint of the SLIM technique is that it can only replicate or represent correlations amongst items which have been co-purchased and / or co-rated through a select number of users. Consequently, SLIM is unable to record transitive associations amongst items which are necessary for superior performance outcomes of Item-Based Collaborative Filtering techniques in datasets which are sparse (Kabbur, Ning and Karypis, 2013).

The SLIM Top-N recommendation approach has a comparable linear model to the Item-Based Memory-Based technique. Here, linear models have been used for Top-N recommendation. The Item-Based Collaborative Filtering model represents a KNN item to item cosine similarity matrix, specifically, every row has exactly (the number of neighbourhoods *k)* nonzero values which signify the similarities (cosine)

amongst items and its most analogous neighbours. The primary difference between the Item-Based KNN and SLIM's linear models is that the Item-Based technique is extremely reliant on the pre-specified similarity measure (item-item) utilised to classify the neighbours, however the SLIM technique produces a sparse matrix of aggregation coefficients of the user-item-rating matrix by resolving the issue of optimisation. Hence, the user-item-rating matrix has the potential to covert strong and refined associations across items which might not be directly apprehended through traditional item-item similarity measures (Ning and Karypis, 2011).

The SLIM Collaborative Filtering technique (Ning and Karypis, 2011) has shown to accomplish respectable performance results on a wide variation of datasets, along with performing better than other state of the art techniques. However, an intrinsic constraint of the SLIM technique is that it can only replicate associations amongst items which have been co-rated through many users. Consequently, this technique cannot record transitive relations amongst items which are important for superior performance of Item-Based Collaborative Filtering techniques within sparse datasets. Here, the transitive relation can be described when two such items can be comparable to one another through virtue of an additional item, which is analogous to both items (Kabbur, Ning and Karypis, 2013).

In 2013, (Kabbur, Ning and Karypis, 2013) proposed a Collaborative Filtering technique known as Factored Item Similarity Models or FISM. This technique memorises the item-item similarity matrix by means of a product of two low dimensional latent factor matrices. In this circumstance, the factored rendition of this item-item similarity matrix permits the FISM technique to both record and replicate correlations amongst items, corresponding to immensely sparse datasets. Through their experimental evaluation, it was identified through numerous datasets with different sparsity levels that the Factored Item Similarity Models (FISM) outperforms the Sparse Linear Method (SLIM) and alternative state of the art Collaborative Filtering techniques. Additionally, in relation the FISM Collaborative Filtering Top-N technique, the comparative performance increases through the sparsity of the datasets (Kabbur, Ning and Karypis, 2013).

To overcome the shortcomings of the SLIM technique, the item-oriented FISM Collaborative Filtering technique utilises a factored item similarity model. By learning the similarity matrix through casting the values into a latent space of considerably reduced dimensionality, which implicitly supports the learning of transitive relations amongst items. Therefore the FISM technique is likely to achieve better performance especially on sparse datasets, this is because it can capture associations amongst items that are not co-rated (Kabbur, Ning and Karypis, 2013).

As further recognised through (Kabbur, Ning and Karypis, 2013), users normally offer their feedback to only a minority of items from a possible list of thousands if not millions of items. As a result, the user-item-rating matrix turn out to be extremely sparse. Collaborative Filtering techniques such as SLIM (along with conventional approaches like Item-Based (Deshpande and Karypis, 2004)) that depend on learning the similarity amongst items fail to apprehend the associations amongst items which have not been co-rated through at least one user. Nonetheless, two items can be comparable to each other through the merit of alternative item that is analogous to both items, also known as a transitive association. Collaborative Filtering techniques built using the Matrix Factorisation approach mitigates this issue by means of extruding data against a low dimensional space, thus indirectly learning effective associations among both users and items (counting items that are not co-rated). Although, such Collaborative Filtering techniques are often outdone by means of the Sparse Linear Method or SLIM (Kabbur, Ning and Karypis, 2013).

To overcome the issue identified in the above paragraph, the FISM Collaborative Filtering technique uses a factored item similarity model which has similar intentions used by the NSVD and SVD++ Collaborative Filtering techniques. In this instance, training the similarity matrix by means of extruding the values into a latent space of a considerably lesser dimensionality covertly assists to train the transitive associations amongst items. Consequently, the FISM technique is estimated to achieve improved results on sparse datasets, since it has the capability to learn associations amongst item that are co-rated (Kabbur, Ning and Karypis, 2013).

In relation to the FISM Collaborative Filtering technique, two diverse models were constructed which utilise different loss functions connected optimisation approaches. These FISM techniques are known as *FISMrmse* and *FISMauc*. The FISMrmse technique calculates the loss by utilising the squared error loss function; whereas the FISMauc technique utilises a ranked loss function grounded on Bayesian Personalised Ranking (BPR), that enhances the Area Under the Curve (AUC) (Kabbur, Ning and Karypis, 2013).

## 4.10 Recommender System Evaluation Metrics

The degree of excellence (quality) of a recommender system technique can evaluated by using several diverse categories of measurements (metrics). The sort of metrics to use for the evaluation of a recommender system is subject to the filtering technique employed (Isinkaye, Folajimi and Ojokoh, 2015). Metrics used for evaluating the quality of predictions and recommendations of a recommender system, as recognised by (Cacheda et al., 2011) through the literature of (Herlocker, Konstan, Terveen and Riedl, 2004) are divided into three categories of metrics: *Prediction Accuracy, Classification Accuracy* and *Rank Accuracy*.

### 4.10.1 Prediction Accuracy Metrics

Prediction Accuracy Metrics measure (evaluate) the accuracy of a recommender system technique by comparing the numerical predicted rating score in contrast to the actual user's rating score (Isinkaye, Folajimi and Ojokoh, 2015). This category of accuracy metrics can also be identified as *Statistical Accuracy Metrics* (Isinkaye, Folajimi and Ojokoh, 2015; Sarwar et al., 2001; Sarwar, Karypis, Konstan and Riedl, 2000) or *Rating Accuracy Metrics* (Lü et al., 2012).

Within literature, there are two widely used statistical accuracy metrics used for recommender system predictive evaluation, these are the *Mean Absolute Error (MAE)* and the *Root Mean Square Error (RMSE).* The most popular and commonly used metric as identified is the Mean Absolute Error (MAE) (Isinkaye, Folajimi and Ojokoh, 2015; Melville and Sindhwani, 2010; Sarwar et al., 2001; Cacheda et al., 2011). Both predictive accuracy metrics are described in much greater detail in the below paragraph.

#### 4.10.1.1 Mean Absolute Error (MAE)

The Mean Absolute Error (MAE), as shown in Equation 4.6, evaluates the accuracy of a Collaborative Filtering technique by measuring the predictive value against that of the user's actual rating, for user-item pairs, as found within the testing data set. In this circumstance, for every rating prediction (user-item) pair, the absolute error is determined. By totalling these user-item pairs and then dividing them by the entire quantity of rating-prediction pairs, this results in the Mean Absolute Error (Arsan, Koksal and Bozkus, 2016)

$$MAE = \frac{\sum_{\{u,i\}} |p_{u,i} - r_{u,i}|}{N}$$ *Equation 4.6*

*Source: Adapted from (Melville and Sindhwani, 2010)*

#### 4.10.1.2 Root Mean Square Error (RMSE)

The Root Mean Square Error however, shown below in Equation 4.7 operates differently to that of the Mean Absolute Error. In this instance, after the rating-prediction pair variance is calculated, its power of 2 is determined (used). From here, the total variance is divided by the entire quantity of rating-prediction pairs and square rooting its result (Arsan, Koksal and Bozkus, 2016).

$$RMSE = \sqrt{\frac{\sum_{\{u,i\}} (p_{u,i} - r_{u,i})^2}{N}}$$ *Equation 4.7*

*Source: Adapted from (Melville and Sindhwani, 2010)*

As shown above in Equation 4.6 Mean Absolute Error (MAE) and Equation 4.7 Root Mean Square Error (RMSE):

- $p_{u,i}$ represents the predictive rating
- $r_{u,i}$ signifies the real or true rating
- $N$ denotes the total quantity of rating-prediction pairs

Additionally, as recognised by (Lü et al., 2012), the smaller the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) correlates to a greater accuracy towards forecasting (prediction). However, as shown in Equation 4.7, the Root Mean

Square Error (RMSE) squares the error prior to its summation, it tends to impose a penalty on greater errors more decisively.

Both metrics, MAE and RMSE, as acknowledged by (Lü et al., 2012) evaluates the totality of ratings in the same way, despite their location within a collection of recommendations they may not be best suited towards frequent tasks. For example, locating a small quantity of items to which a specified user may value highly (the discovery of noteworthy items). Nonetheless, because of their clarity, both the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) rating accuracy metrics are commonly used towards recommender system evaluation.

### 4.10.2 Classification and Ranking Accuracy Metrics

Classification Accuracy Metrics measure (evaluate) just how successful a recommender system algorithm is at choosing the best calibre of items from the total collection of all items for a user. These category of metrics considers the method of prediction as a binary task which differentiates items that are predicted (good) from those that are not (bad) (Sarwar et al., 2001). Additionally, these classification of evaluation metrics can also be known as *Decision Support Accuracy Metrics* (Sarwar et al., 2001).

It is recognised by (Cacheda et al., 2011), that this category of accuracy metrics are suitable for the task of discovering *good items* for a user, particularly when their preferences are that of a binary nature (liked or disliked). However, dissimilar to binary preferences, if a user's preferences are of a numeric range (1 to 5), this type of accuracy metrics do not determine the precise order of items within the Top-N recommendation list. These metrics merely evaluate if the recommended items for a user are satisfactory, without taken any regard to what item or items are more desirable.

There are numerous types classification accuracy metrics identified within literature, with the most recognised of these being *ROC* (similarly identified as *AUC), Precision* and *Recall* (Cacheda et al., 2011; Lü et al., 2012; Melville and Sindhwani, 2010; Isinkaye, Folajimi and Ojokoh, 2015).

#### 4.10.2.1 AUC (Area Under ROC Curve)

The Area Under ROC Curve, with *ROC* signifying *Relative Operating Characteristic* is a well-known classification accuracy metric which determines how a recommender system effectively differentiates significant items (valued by a user) from insignificant items (all remaining items) (Lü et al., 2012).

$$\text{AUC} = \frac{n' + 0.5n''}{n} \qquad \textit{Equation 4.8}$$

*Source: Adapted from (Lü et al., 2012)*

To compute the AUC, the most straightforward approach is by measuring the likelihood that the significant (relevant) items will be recommended to a user in contrast to the insignificant (irrelevant) items. As shown above in Equation 4.8, for *n* distinct comparisons (where every comparison denotes to the selection of one significant and insignificant item), if there are *n'* instances where the significant item has a *higher* total than the insignificant item and *n"* instances where the totals are the same, then the AUC is equal to the equation as shown in Equation 4.8. Noticeably, if all the significant items have a greater total than that of the insignificant items, then the AUC is equal to 1 which denotes an impeccable recommendation list; whereas for an irregular recommendation list, the AUC would be equivalent to 0.5. As a result, the extent of which the AUC surpasses 0.5 specifies the capability of a recommender system algorithm in classifying significant items for a user (Lü et al., 2012).

The AUC or Area Underneath an ROC Curve which is also recognised as *Swet's A Measure* can be used as a *single classification accuracy metric* for a recommender system algorithm's capability to differentiate *good items* (relevant) from *bad items* (irrelevant). The area that is represented underneath the curve is similar to the likelihood that the recommender system algorithm will have the ability to correctly select between two items, one randomly chosen item from a group of relevant items and one randomly chosen item from the group of irrelevant items (Herlocker et al., 2004). Additionally, the area underneath the curve represents the *Recall* of the recommender system algorithm against the *Fallout* (correlation amongst item hits and item misses) (Cacheda et al., 2011).

#### 4.10.2.2 Precision and Recall

Precision can be characterised as the ratio of relevant items chosen, to the quantity of items chosen (recommended). The equations for both this classification accuracy metric is shown below in Equation 4.9 (Herlocker et al., 2004).

$$P = \frac{N_{rs}}{N_s} \qquad \textit{Equation 4.9}$$

*Source: Adapted from (Herlocker et al., 2004)*

Recall is described as the ratio of relevant items chosen, to the total quantity of irrelevant items obtainable. The equations for both this classification accuracy metric is shown below in Equation 4.10 (Herlocker et al., 2004).

$$R = \frac{N_{rs}}{N_r} \qquad \textit{Equation 4.10}$$

*Source: Adapted from (Herlocker et al., 2004)*

**Table 4.2** Precision Recall

| | **Selected** | **Not Selected** | **Total** |
|---|---|---|---|
| **Relevant** | $N_{rs}$ | $N_{rn}$ | $N_r$ |
| **Irrelevant** | $N_{is}$ | $N_{in}$ | $N_i$ |
| **Total** | $N_s$ | $N_n$ | N |

Both the Precision and Recall are calculated from a two by two table, as shown above in Table 4.2. In relation to the evaluation of recommendation algorithms, it is desirable that both the Precision and Recall accuracy metrics have a preferably high value. However, both the Precision and Recall metrics are inversely linked, to the extent that when the Precision increases the Recall generally lessens, and vice versa (Cacheda et al., 2011).

Ranking Accuracy Metrics calculate the capability of a recommendation technique to present a Top-N recommended collection of ordered items which matches how a user would have grouped similar items. Contrary to Classification Accuracy Metrics, Ranking Accuracy Metrics are more suitable for evaluating recommendation

techniques which will be used to offer ranked Top-N recommendation lists to a user, especially within domains where the preferences of a user in recommendations are of a non-binary nature (Herlocker et al., 2004).

This category of accuracy metrics however does not aim to calculate the capability of a recommendation technique to correctly predict the rating score intended for an individual item. Ranking Accuracy Metrics are not the same as Prediction Accuracy Metrics, and are not suitable for the evaluation of the *Annotation in Context* task (Herlocker et al., 2004). The task of Annotation in Context is described through (Cacheda et al., 2011) as the prediction of a specified item to a user. Here, the user first choses the item or items of which are of interest. Then the recommendation technique predicts the rating that the user would give for that specific item. Additionally, as acknowledged by (Herlocker et al., 2004), if a recommendation technique will be presenting the prediction score of ratings, then it is imperative to also evaluation the following technique through the use of a Prediction Accuracy Metric, as previously described earlier in evaluation of recommender system techniques.

As intended for Rating-Based Collaborative Filtering technique, the accepted evaluation standard is the prediction of rating accuracy. In this context, the most widely used prediction accuracy measures include the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE), as described in the *Predictive Accuracy Metrics* above. Here, both evaluation measures are subject to the difference between the true or actual rating and its predicted rating. Ranking-Based however places emphases on the ranking of items as opposed to the prediction of ratings (Wang, Sun, Gao and Ma, 2014).

According to literature, two commonly used ranking oriented evaluation measures are the *Normalised Discounted Cumulative Gain (NDCG)* and the *Mean Average Precision (MAP)* (Steck, 2013; Wang et al., 2014). Both measures are favoured in information retrieval for the evaluation of ranked results, where information (i.e. documents) are allocated graded relevance judgements in the NDCG and binary relevance judgements in MAP. Additionally, in the context of Collaborative Filtering techniques, the rating of items given by a user(s) can certainly be used as a form of relevance judgements (Wang et al., 2014; Liu and Yang, 2008).

#### 4.10.2.3 Normalised Discounted Cumulative Gain (NDCG)

Firstly, the NDCG measure is evaluated over an approximate number *k* of the top items located on the ranked list of items. To further explain the NDCG measure, let *Q* be the group of users used for testing and *R(u, p)* be the rating which is allocated by *u* to the item on the *p-th* position of the ranked list, that is constructed for the user *u*. In this context, the NDCG at the *k-th* position with consideration to the group of users *U:* shown below in Equation 4.11, where $Z_u$ is the calculated normalisation factor so the NDCG for the optimum ranking holds a value of 1. The evaluation values of the NDCG measure ranges between value of 0 to 1, where a higher value signifies a superior ranking efficiency. The NDCG evaluation measure is especially sensitive towards the ratings of the highest ranked items which is shown through the discounting factor *log(1 + p)*, that increases with the position in the ranking. It is through this feature which makes it highly preferable for evaluating the ranking superiority within recommender systems. This is because most users rarely look beyond the first few items presented in a recommendation list, therefore the most relevant items positioned higher on the recommendation list are far more significant than items positioned lower on the list (Liu and Yang, 2008).

$$\mathrm{NDCG}(Q,k) = \frac{1}{|Q|}\sum_{u\in Q} Z_u \sum_{p=1}^{k} \frac{2^{R(u,p)}-1}{\log(1+p)} \qquad \textit{Equation 4.11}$$

*Source: Adapted from (Liu and Yang, 2008)*

#### 4.10.2.4 Mean Average Precision (MAP)

Secondly, the Mean Average Precision (MAP) which is also identified as the Average Precision (AP) can be acknowledged as the average of the precision value achieved for the group of Top-N items that occur after each relevant item is extracted (Dev and Mohan, 2016). As shown in Equation 4.12, where *P@n* signifies the precision within the recommendation list's Top-N results of the ranked list of items intended for a user. The Average Precision (AP) for a user *u* can be described as the average of the *P@n* values aimed at all relevant items, where *rel(n)* represents the binary function which maps an item to either being relevant (1) or irrelevant (0). Additionally, the Mean Average Precision (MAP) takes the mean of the Average Precision (AP) values over the group of users *U* (Wang et al., 2014). Comparable to the NDCG evaluation measure, the MAP or AP evaluation value ranges between value

of 0 to 1, where a higher value signifies a superior ranking efficiency and 1 represents the ideal ranking of the Top-N items (Dev and Mohan, 2016).

$$AP_u = \frac{\sum_{p=1}^{N} \left(P@n \times rel_{u,p}\right)}{\#\text{ relevant items for user u}}, \qquad \textit{Equation 4.12}$$

*Source: Adapted from (Wang et al., 2014)*

# 4.11 Summary

The purpose of this chapter was to present an overview of recommender systems. The type of data used, the various categories of recommender system techniques and how recommender systems are evaluated are described in detail. For this thesis, the Collaborative Filtering technique is identified to be the suitable recommender system approach. Not only is this approach the most widely used recommender system technique, evaluation of the Collaborative Filtering technique is made possible using specific metrics which is not possible through approaches such as Content-Based Filtering.

Collaborative Filtering techniques can be used for the prediction or recommendation of an item to a user. For this thesis, the recommendation approach, known as Top-N Collaborative Filtering will be used. This approach presents users with a Top-N recommendation list of items suggested for a specific user. The Memory-Based techniques of User-Based and Item-Based, along with state-of-the-art Model-based techniques are chosen for this thesis. In addition, these Top-N Collaborative Filtering approaches are to be evaluated using decision support evaluation metrics. This category of evaluation metric measures the position of items recommended for a user within a Top-N recommendation list.

In the next chapter, relating to the Methodology, the data collection, data pre-processing, data analysis, recommender system tools and techniques used. In addition, the experimental setup and evaluation metrics used within the experiments are presented.

# Chapter 5 Methodology

## 5.1 Introduction

The major aim of this research project is to use various categories of recommender system techniques for the identification and similarity-based ranking of cyber security information. This cyber security information relates to software and hardware vulnerabilities. Here the hypothesis is that the similarity-based ranking of this cyber security information can increase the user satisfaction of security personnel through a ranked list of recommended security information.

Section 5.2 of this chapter will describe the employed data collection and information retrieval process. Section 5.3 outlines the data pre-processing and data analysis tools and techniques used. Section 5.4 presents the implemented recommender system tools and approaches. Section 5.5 will describe the setup and design regarding experiments. Section 5.6 outlines the evaluation measures used to assess the experiments. Section 5.7 will conclude this chapter with a summary.

## 5.2 Data Collection / Information Retrieval

Cyber security data was kindly supplied by the Security Research Group from the University of Trento, Italy. Through their Security Research Group, numerous cyber security datasets were made available which have been used in numerous research projects, for instance: *Comparing Vulnerability Severity and Exploits Using Case-Control Studies* (Allodi and Massacci, 2014). The National Vulnerability Database (NVD) dataset was used through the research of (Allodi and Massacci, 2014). This dataset, along with others used within their research have been collected from publicly available repositories, for example: data from the National Vulnerability Database can be obtained in many different formats (XML and most recently JSON).

The data supplied through the Security Research Group of the University of Trento was presented in CSV format, compared to the raw format nature of the National Vulnerability Database. For this research project, the NVD dataset was used, where the NVD can be described as a repository or reference database for disclosed

vulnerabilities and is coined the phrase *"The Universe of Vulnerabilities"* by (Allodi and Massacci, 2014).

### 5.2.1 Open Vulnerability Data: National Vulnerability Database and CVE Details

The National Vulnerability Database (NVD) dataset as supplied by the University of Trento comes in CSV format and comprises of 63,736 rows of vulnerability data, where vulnerabilities are documented from the years 1999 up to October 2014. The attributes of the NVD dataset as supplied by the University of Trento is shown in Table 5.1 below, with an excerpt of the dataset shown in Figure B.1 of Appendix B.

**Table 5.1** National Vulnerability Database Attributes

| **National Vulnerability Database Attributes** | |
|---|---|
| **Common Vulnerability and Exposures (CVE) ID** | Identifier of the vulnerability |
| **Publication Date** | First publication date of the vulnerability |
| **Modification Date** | Date of last updated vulnerability entry |
| **Common Vulnerability Scoring System (CVSS) Score** | CVSS version 2.0 risk score of the vulnerability |
| **Common Vulnerability Scoring System (CVSS) Impact** | CVSS version 2.0 impact score of the vulnerability |
| **Common Vulnerability Scoring System (CVSS) Exploitability** | CVSS version 2.0 exploitability score of the vulnerability |
| **Common Vulnerability Scoring System (CVSS) Access Vector** | CVSS exploitability assessment: Access Vector |
| **Common Vulnerability Scoring System (CVSS) Access Complexity** | CVSS exploitability assessment: Access Complexity |
| **Common Vulnerability Scoring System (CVSS) Authentication** | CVSS exploitability assessment: Authentication |
| **Common Vulnerability Scoring System (CVSS) Confidentiality** | CVSS impact assessment: Confidentiality |
| **Common Vulnerability Scoring System (CVSS) Integrity** | CVSS impact assessment: Integrity |
| **Common Vulnerability Scoring System (CVSS) Availability** | CVSS impact assessment: Availability |
| **Affected Software** | Software that is affected by the vulnerability |
| **Edition** | Latest edition of the software and hardware |

| | |
|---|---|
| **Last Version** | Latest version of the software and hardware |
| **Vendor** | Vendor of the software |
| **Description** | English description of the vulnerability |
| **Category** | Classification of the software and hardware |

As the National Vulnerability Database dataset provided by the University of Trento contains Common Vulnerability and Exposures (CVE), Common Vulnerability Scoring System (CVSS) and Common Platform Enumeration (CPE) information, a small number of attributes however were unavailable in this dataset, for instance the Common Weakness Enumeration (CWE) identifier. To obtain the CWE identifier, data was acquired from the CVE Details (https://www.cvedetails.com) website. This website is an unrestricted Common Vulnerability and Exposures security vulnerability database or information source.

Vulnerability information was acquired by its date which range from the years 1999 to 2014. This was to correlate with the National Vulnerability Database data provided by the University of Trento. Comparable to the National Vulnerability data, information from CVE Details was stored within a CSV format which comprises of 66,399 rows of vulnerability information. The attributes of the CVE Details dataset are shown below in Table 5.2, with an excerpt of the dataset shown in Figure B.2 of Appendix B.

**Table 5.2** CVE Details Attributes

| **CVE Details Attributes** | |
|---|---|
| **Common Vulnerability and Exposures (CVE) ID** | Identifier of the vulnerability |
| **Common Weakness Enumeration (CWE) ID** | Identifier for software weaknesses and vulnerabilities |
| **# of Exploits** | Number of exploits for the software weakness or vulnerability |
| **Vulnerability Type(s)** | Category of software weakness or vulnerability |
| **Publish Date** | First publication date of the vulnerability |
| **Update Date** | Date of last updated vulnerability entry |
| **Score** | CVSS version 2.0 base score of the vulnerability |
| **Gained Access Level** | Status of access required to gain entry to exploit software and hardware |

| | |
|---|---|
| **Access** | CVSS exploitability assessment: Access Vector |
| **Complexity** | CVSS exploitability assessment: Access Complexity |
| **Authentication** | CVSS exploitability assessment: Authentication |
| **Conf.** | CVSS impact assessment: Confidentiality |
| **Integ.** | CVSS impact assessment: Integrity |
| **Avail.** | CVSS impact assessment: Availability |

As shown in Table 5.2, relating to the CVE Details Attributes, several of these attributes share a common pattern with the attributes found in Table 5.1, relating to the National Vulnerability Database Attributes. As both datasets share comparable attributes, it can be observed that some attribute do not appear in both datasets. For instance, the Common Weakness Enumeration (CWE) identifier appears in the CVE Details dataset and not in the National Vulnerability Database dataset, provided through the University of Trento.

As some attributes appear in one dataset and not the other, for instance the CWE identifier, both the National Vulnerability Database and CVE Details datasets were merged together by means of the Common Vulnerability and Exposures (CVE) identifier. Here the CVE identifier acts as the primary attribute which connects both datasets together. Additionally, instances (rows) of both datasets which share a CVE identifier were only included into the merger of both the National Vulnerability Database and CVE Details datasets. A comparable approach in relation to the merging of vulnerability information by their common identifier was achieved through the research of (Bozorgi, Saul, Savage and Voelker, 2010).

The purpose for merging both datasets together was to produce one central dataset, containing all the attributes belonging to both datasets, through their matching CVE identifiers. Through this central dataset, the removal of rows which only comprised of a CVE identifier and where subsequent associated attribute data relating to that CVE identifier was absent, could be achieved with greater control compared to removing these separately in both datasets. Signifying, that every row contained within this dataset will contain a CVE identifier and associated attribute information contained in both the National Vulnerability Database and CVE Details datasets, however this central dataset allows straightforward dataset regulation while

minimising absent data. Additionally, instead of having to access two independent datasets, this central dataset allows the selection of required attributes to be accessed through one data source. This allows the attribute data (merged from the National Vulnerability Dataset and CVE Details datasets) present in this central dataset, to always match its associated CVE identifier.

## 5.3 Data Pre-Processing and Data Analysis

In relation to the merger of the National Vulnerability Database and CVE Details datasets, and the further pre-processing of this newly combined dataset, the programming tool *RStudio* was used. RStudio is an open-source integrated development environment (IDE) for *R* (RStudio, 2018). R is a programming language and environment for statistical computing and graphics (The R Foundation, 2018).

As stated in the above paragraph, once the National Vulnerability Database and CVE Details datasets were merged, the new combined dataset comprising of all attributes shown in both Table 5.1 and Table 5.2 was further pre-processed. The dataset was pre-processed to only comprise of software and hardware vulnerability information instances (rows), where the *publication date* of the vulnerability started from the year 2005 or '2005-01-01'. Meaning, that vulnerability information present in the NVD and CVE Details dataset ranges from the years 2005 up to 2014. As this pre-processing step is taken from research by (Zhang, Caragea and Ou, 2011), it was identified that vulnerability information prior to 2005 is regarded to be unstable. This is shown in Figure 5.1 below.

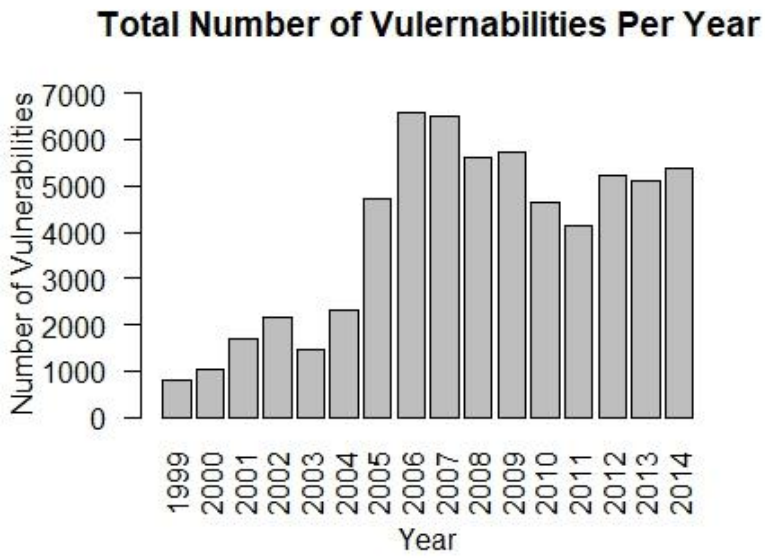



**Figure 5.1** Trend of Vulnerability Numbers

*Source: Adapted from (Zhang, Caragea and Ou, 2011)*

Due to the vast amount of software and hardware vulnerabilities contained within the NVD and CVE Details 2005 to 2014 dataset, additional pre-processing was undertaken in relation to *Vendors*. As shown in Figure 5.2 below, it presents the top 20 vendors located within the NVD and CVE Details 2005 to 2014 dataset.

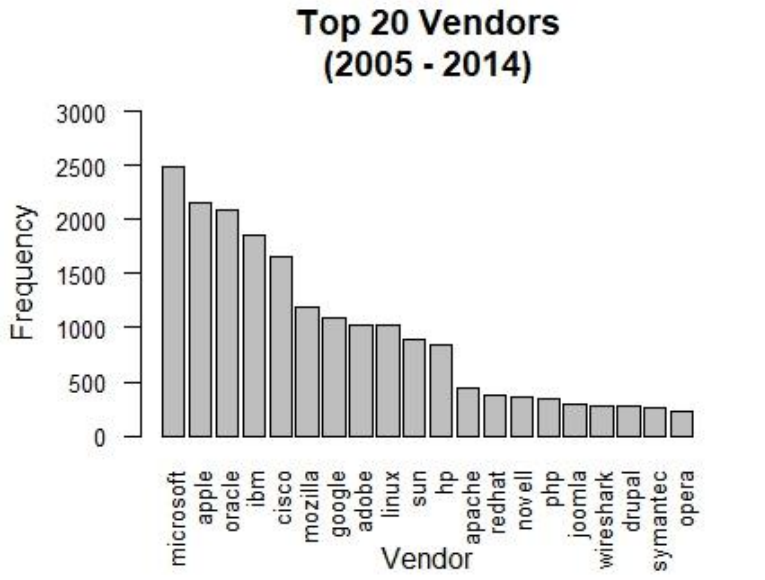



**Figure 5.2** Top 20 Vendors (2005 to 2014)

Given the large number of vendors present in the data, the following list of major vendors were selected: *Linux*, *Microsoft*, *Mozilla*, *Google*, *Apple*, *Sun* and *Cisco*. The Vendor attribute, as shown in Table 5.1, relates to the third component of the Common Platform Enumeration (CPE), located within the National Vulnerability Database. The data pre-processing stages which comprise of the selection of vulnerability instance where the publication date of vulnerability starts from the year 2005 and the selection of specified vendors present in the NVD was identified through the research of (Zhang, Caragea and Ou, 2011). The frequency of these specified vendors is shown in both Figure 5.3 and Table 5.3 below.

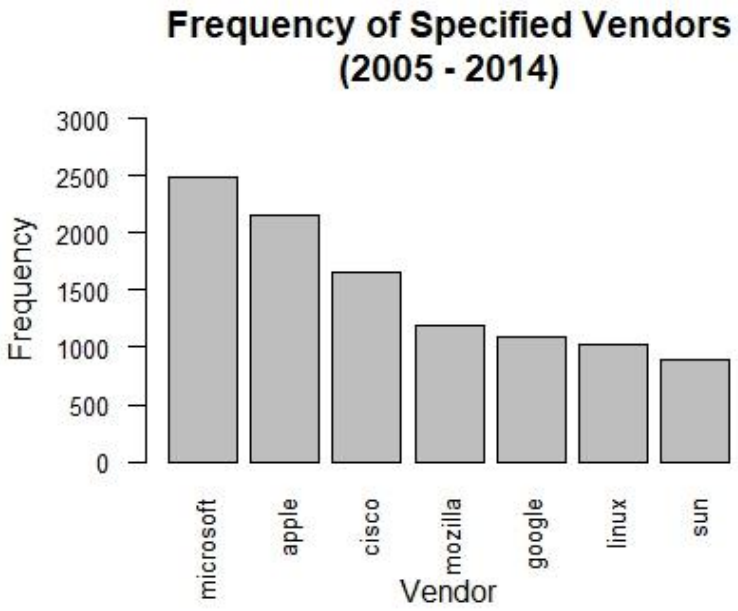



**Figure 5.3** Frequency of Specified Vendors (2005 to 2014)

**Table 5.3** Table of the Frequency of Specified Vendors from the years 2005 to 2014

| **Frequency of Specified Vendors (per Figure 5.3)** | | | | | | |
|---|---|---|---|---|---|---|
| **Microsoft** | **Apple** | **Cisco** | **Mozilla** | **Google** | **Linux** | **Sun** |
| 2485 | 2148 | 1662 | 1192 | 1085 | 1021 | 886 |

In conjunction with the specified vendors, as acknowledged in the above paragraph, and as seen in both Figure 5.4 and Table 5.4, the top 20 affected software and hardware products were identified. To construct an accurate list of the twenty most affected software and hardware products, both the *Vendor* and *Affected Software* attributes were combined, where both the vendor and affected software attributes are separated through *"|"*. The top 20 affected software and hardware products are shown below in Figure 5.4, with the frequency of affected software and hardware shown in Table 5.4.

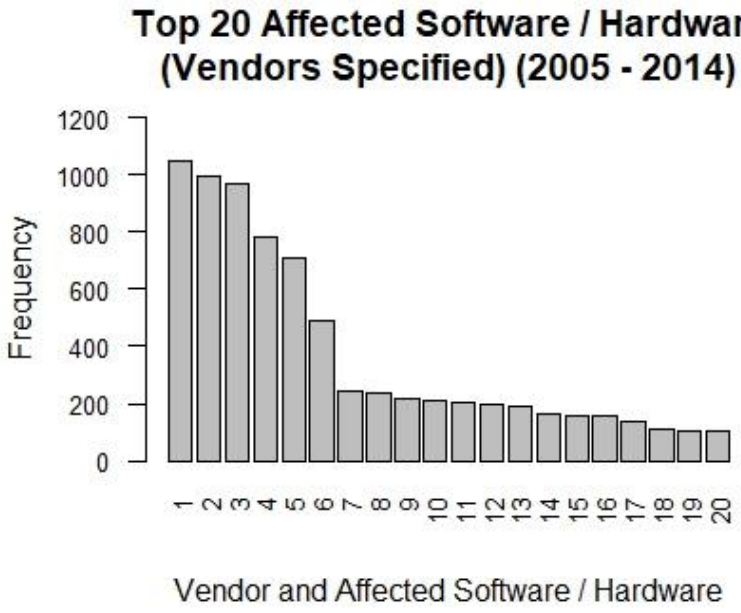



**Figure 5.4** Top 20 Affected Software / Hardware (Specified Vendors) from 2005 to 2014

**Table 5.4** Top 20 Affected Software / Hardware – From Highest (1) to Lowest (20)

| | | | | |
|---|---|---|---|---|
| ***Mozilla \| Firefox*** | ***Linux \| Linux_ Kernel*** | ***Google \| Chrome*** | ***Microsoft \| Internet_Explorer*** | ***Apple \| Mac_OS_X*** |
| 1046 | 997 | 971 | 783 | 706 |
| ***Apple \| Safari*** | ***Cisco \| IOS*** | ***Apple \| iTunes*** | ***Microsoft \| Windows_2000*** | ***Apple \| iPhone_OS*** |
| 492 | 243 | 238 | 220 | 207 |
| ***Apple \| QuickTime*** | ***Sun \| JDK*** | ***Microsoft \| Windows_7*** | ***Microsoft \| Windows_2003_Server*** | ***Sun \| Solaris*** |
| 206 | 196 | 191 | 163 | 159 |
| ***Microsoft \| Office*** | ***Sun \| Sunos*** | ***Microsoft \| Excel*** | ***Cisco \| Adaptive_Security_Software _Software*** | ***Cisco \| Unified_Communications _Manager*** |
| 156 | 139 | 112 | 103 | 103 |

As Collaborative Filtering uses the user-item-rating matrix to construct the prediction of ratings and item recommendation, avoiding the necessity for accumulating considerable amount of information regarding both users and items (Wang et al., 2014), a user-item-rating dataset was constructed from the pre-processed NVD and CVE Details 2005 to 2014 specified vendors dataset. To coincide with this user-item-rating, the following attributes were selected for the construction of the initial dataset: For a *User*, the *Vendor* and *Affected Software* attributes were chosen. As *LibRec* (explained in Section 5.4) requires the users, items and ratings to be stored in their own separate columns [one column for users, one column for items and one column for ratings], the vendor and affected software attributes were combined into one column. The merger of the vendor and affected software attributes, for the construction of the user column was achieved by combining both attributes into a newly created column, where the vendor and affected software attributes were separated through "|". For example: *Mozilla | Firefox*. In addition, the Vendor attribute relates to the *Vendor* component of the *Common Platform Enumeration (CPE),* and the Affected Software attribute is equivalent to the *Product* component of the *Common Platform Enumeration (CPE),* both located within the NVD. For an *Item*, the *Common Weakness Enumeration (CWE)* identifier attribute was selected. As for the *Rating*, the *Common Vulnerability Scoring Systems (CVSS)* score attribute was chosen.

Identical to the data pre-processing stages used for the construction of the above User-Item-Rating dataset, two additional datasets were created in the same manner as the initial dataset. However, the only change made to the datasets was in relation to the *user* column. In the initial dataset, the user column was built through the merger of the *Vendor* and *Affected Software* attributes. For the second dataset, the user column comprised of the Vendor and Affected Software attributes as found in the initial dataset, were combined with the *Edition* and *Last Version* attributes, as found in the NVD. The purpose for the addition of these attributes was to apply the additional CPE information relating to the vulnerable software and hardware. Unlike the initial dataset which consists of the Vendor and Affected Software, the addition of the Edition and Last Version of the vulnerable software and hardware gives an exact description of what specific software and hardware are affected by vulnerabilities. An example of this dataset is as follows: *Mozilla | Firefox | N\A | 9.0.1*. For the third dataset, the user column comprised of the *Affected Software, Edition* and *Last Version* attributes, as located within the NVD. In contrast to the second dataset, the Vendor attribute was not included in the User column of this dataset, to examine if the absence of this attribute increases or decreases recommendation accuracy to that of the second dataset. An example of the third dataset is shown as follows: *Firefox | N\A | 9.0.1*. Comparable to the combining of the user column attributes for the initial dataset (Vendor and Affected Software), the user column attributes for both dataset two and dataset three were separated through "|". Additionally, comparable to the construction of the initial dataset, both the *item* and *rating* columns remained identical for the two newly created datasets (item: CWE identifier and rating: CVSS Score). These three datasets are shown in Table 5.5 below.

**Table 5.5** User-Item-Rating Datasets and Attributes

| **Datasets** | | | |
|---|---|---|---|
| **#** | **User** | **Item** | **Rating** |
| **1** | Vendor \| Affected Software<br>**606** | CWE ID<br>**30** | CVSS Score<br>**7484** |
| **2** | Vendor \| Affected Software \| Edition \| Version<br>**1960** | CWE ID<br>**30** | CVSS Score<br>**7484** |
| **3** | Affected Software \| Edition \| Version<br>**1959** | CWE ID<br>**30** | CVSS Score<br>**7484** |

The three datasets as identified in Table 5.5 were individually stored in the format of separate Comma Separated Values (CSV) files, as this is data file format used by LibRec for the reading and pre-processing of data. The *User* column of each of these three datasets is presented in text format, and both the *Item* and *Rating* columns are presented in numerical or integer format. As LibRec reads and processes the data in numerical format, the *User* column of each of the datasets was converted from text format to numerical or integer format. In addition, within the *Item* column of each of the three datasets which signifies the CWE ID, numerous CWE IDs were unavailable. Here, rows in each of the three datasets where the CWE ID was unavailable (marked with *NA*) were removed from the dataset(s). The purpose for the removal of these unavailable CWE IDs was to make all the columns within each of the datasets fully complete (*with no missing values in the User, Item and Rating columns; as both the User and Rating columns contained no missing values*), and CWE ID *NA* item values would not be recommended to a user of the recommender system.

To reiterate, in relation to the *User-Item-Ratings* of the three datasets, identified in Table 5.5: The Vendor | Affected Software (dataset 1), Vendor | Affected Software | Edition | Version (dataset 2) and the Affected Software | Edition | Version (dataset 3) signifies the *User* attribute as this represents the type of software and hardware which is affected by a vulnerability; the CWE ID indicates the numerical value for the software and hardware weakness and vulnerability which represents the *Item* attribute. Lastly, the CVSS Score specifies the base risk score of the software and hardware weakness and vulnerability which signifies the *Rating* attribute. Excerpts of these three User-Item-Rating datasets are shown in Appendix C, however only the User-Item-Rating attributes ending with **.1** were used and employed into the recommender systems techniques. This is due to LibRec only permitting to use of numerical datasets. The R code relating to the acknowledged data pre-processing stages are shown in Appendix D.

## 5.4 Recommender System Tools and Techniques

The programming tool which was used to experimentally research the use of cyber security information implemented within a recommendation environment is known as *LibRec* (*https://www.librec.net)*. LibRec is an innovative open source Java library of recommender system techniques, comprising of approximately 70+ recommendation algorithms. This recommender system library implements a collection of state-of-the-art recommendation algorithms, in addition to traditional recommender system techniques which have the capability of effectively solving both rating prediction and Top-N item ranking issues. LibRec also implements several different evaluation measures or metrics, used to assess the prediction of rating and the recommendation of items to a user (Guibing Guo, Jie Zhang, 2015). In addition, Recommender Systems are a category of Machine Learning techniques which are used to offer personalised recommendations to its user (LibRec, 2018).

In relation to the experiments, undertaken in this research, the following Collaborative Filtering recommendation techniques were implemented, using the LibRec Java library. The Java code which used the LibRec library is shown in Appendix E.

- **Memory-Based Collaborative Filtering**
    - User-Based (UserKNN) (Konstan et al., 1997)
    - Item-Based (ItemKNN) (Deshpande and Karypis, 2004)
- **Model-Based Collaborative Filtering (Matrix Factorisation)**
    - FISM
        - FISMrmse (Kabbur, Ning and Karypis, 2013)
        - FISMauc (Kabbur, Ning and Karypis, 2013)
- **Other**
    - SLIM (Ning and Karypis, 2011)

## 5.5 Experimental Setup / Design

For the following experiments, 80% randomly rated items were selected for the training dataset and the remaining 20% were selected for the testing dataset. Each Collaborative Filtering algorithm was executed 5 times, and the average performance results were recorded (Wang et al., 2014). This approach is known as K-Fold Cross-Validation, and the valuation used within the subsequent experiments was identified through the research of (Tang and Tong, 2016; Gogna and Majumdar, 2015; Zhang et al., 2014).

(Ricci et al., 2011) outlines the Cross-Validation process. Both the training and testing datasets are constructed through the existing dataset. The recommendation model is trained by using the dataset and then tested through instances within the testing dataset. Next, distinct training and testing datasets are chosen to commence the process of training and test, and this process is repeated *K* times. Lastly, the *average* performance of the *K* learned recommendation models are documented. For *K-Fold Cross-Validation*, the dataset is separated into *k* folds. One of the folds is used for testing the recommendation model, and the outstanding *n-1* folds are used for training the recommendation model. From here, the Cross-Validation process is repeated *n* times where each *n* subsamples are used precisely once as a validation dataset.

To verify that the results of the following experiments were statistically accurate, every recommendation algorithm was executed five different times, where each execution used a different randomised subdivision of the training and testing datasets. The recommendation algorithm results presented are the averages over these five algorithm executions. Lastly, in these experiments, the value of $N = 10$ was used as the number of chosen items to be recommended to a user through the *Top-N* recommendation techniques (Karypis, 2001). This *N* value, in addition to the research of (Karypis, 2001) has been also used through the research of (Ning and Karypis, 2011; Kabbur, Ning and Karypis, 2013; Kang, Peng and Cheng, 2016).

## 5.6 Experimental Evaluation Measures

As acknowledged within recommender system research, commonly used rating prediction accuracy measures comprise of the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE). Both of these accuracy measures depend on the variance between the true or actual rating and the predicted rating (Wang et al., 2014).

For the following experiments, focus is placed on the evaluation of item classification and item rankings rather than the prediction of ratings. To evaluate Top-N Collaborative Filtering algorithms, one classification-oriented evaluation measure was used: *Area Under the ROC Curve (AUC)*. In addition, two ranking oriented evaluation measures were also used: *Normalised Discounted Cumulative Gain (NDCG)* and *Mean Average Precision (MAP).*

In addition to the Java code shown in Appendix E, Java Properties files were also used. This was for the setup of the various Collaborative Filtering techniques, together with the configuration of Top-N evaluation metrics used within experiments, as shown in Appendix F.

## 5.7 Summary

In this chapter, NVD data was kindly supplied by the Security Research Group from the University of Trento, Italy. Analysis was performed on this data to identify certain trends within the data itself. Comparable with the analysing of the data, the NVD data was pre-processed to fit the desired dataset type to be executed by numerous Top-N recommender system techniques. The data used by Collaborative Filtering comprises of a user, item and a rating. To fit this type of dataset, attributes from the NVD were identified to being comparable to a user, item and rating. The user component comprises of the NVD attributes: Vendor, Affected Software and Hardware, Edition and Version. The item component contains the CWE ID NVD attribute and the rating component comprises of the CVSS Score. In this context of cyber security, a user is presented with a Top-N recommendation list of vulnerabilities (item) which affect specific software and hardware (user) which can be based on the similarity of the user, item and its associated rating (CVSS Score: item).

This chapter presented the Top-N Collaborative Filtering techniques to be employed in relation to the upcoming experiments which use the three datasets acknowledged in Table 5.5. The state-of-the-art techniques of SLIM and FISM are identified to offer a much greater quality of recommendations, compared to the traditional approaches of User-Based and Item-Based. In addition, the experimental setup and the evaluation metrics used were acknowledged in this chapter.

The following chapter outlines the set of experiments which use the traditional Memory-Based techniques, in addition to more state-of-the-art Collaborative Filtering approaches by means of a Top-N recommendation context, using the three user-item-rating datasets as seen in Table 5.5.

# Chapter 6 Experiments and Results

## 6.1 Introduction

Collaborative Filtering is recognised as the most widely used technique (Sharma, Gopalani and Meena, 2017; Thorat, Goudar and Barve, 2015; Gogna and Majumdar, 2015; Zhang et al., 2014; Sarwar et al., 2001). Collaborative Filtering can be further divided into Memory-Based and Model-Based techniques. In Memory-Based approaches two of the most widely used similarity measures are Pearson's Correlation Coefficient and Cosine (Isinkaye, Folajimi and Ojokoh, 2015; Zhang et al., 2014; Lee, Sun and Lebanon, 2012; Sánchez et al., 2008). Memory-Based techniques can be used for both prediction and recommendation.

To overcome the disadvantages relating to Memory-Based techniques Model-Based approaches were developed (Sharma, Gopalani and Meena, 2017). Model-Based techniques use user rating data to train models for the prediction of user ratings through machine learning algorithms (Tang and Tong, 2016; Isinkaye, Folajimi and Ojokoh, 2015). Latent Factor and Matrix Factorisation models have emerged as the state-of-the-art approaches within the category of Collaborative Filtering (Melville and Sindhwani, 2010). Model-Based techniques generate a higher cost when producing recommendations. Therefore, the quality of the recommendations are significantly improved and achieve the best performance (Ning and Karypis, 2011).

This chapter is presented as four subsections. Section 6.2 outlines the experimental parameters and results for the Item-Based and User-Based Top-N Memory-Based techniques using the three datasets presented in Table 5.5. Section 6.3 defines the experimental parameters and presents the experimental outcomes for the novel SLIM Top-N technique using the same three datasets. Section 6.4 outlines the parameters and displays the results for the FISM Top-N Collaborative Filtering technique.

## 6.2 Memory-Based Top-N Collaborative Filtering Neighbourhood Size Sensitivity via Similarity Measures

The experiments in this section evaluate the model size sensitivity, the number of K Nearest Neighbours (KNNs) for the Memory-Based Collaborative Filtering techniques. The methods used are User-Based and Item-Based Collaborative Filtering. The value of KNNs used for the following experiments are as follows: 10, 20, 30 ,40 and 50 identified through (Karypis, 2001).

In addition to the KNN parameter values Pearson's Correlation Coefficient and Cosine-Based similarity measures are used. As previously mentioned in Section 5.5, the following experiments uses 5 K-Fold Cross Validation and the top 10 (Top-N) chosen items are to be recommended for a user.

For the evaluation in relation to the accuracy of the Top-N Memory-Based Collaborative Filtering Recommender System algorithms three-evaluation metrics as employed. These are the Area Under the ROC Curve (AUC), the Normalised Discounted Cumulative Gain (NDCG) and Mean Absolute Precision (MAP). The results shown in the following experiments display the average performance results of the 5 K-Fold Cross Validation process.

### 6.2.1 Experiment A: Item-Based Neighbourhood Size Sensitivity via Pearson's Correlation Coefficient Similarity Measure

The first experiment employed the Item-Based Memory-Based Collaborative Filtering technique uses a number of KNN value parameters assigned to the Recommender System algorithm. In addition to these KNN values, a similarity measure was also assigned to the Item-Based algorithm. For this experiment the Pearson's Correlation Coefficient (PCC) similarity measure was implemented. The results for the Item-Based Neighbourhood Model Size Sensitivity using the Pearson's Correlation Coefficient are shown below in Tables 6.1, 6.2 and 6.3.

**Table 6.1** Item-Based (Pearson) Top-N evaluation results for the first software / hardware vulnerability dataset

| **Dataset: 1** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Vendor \| Affected Software* | | *CWE ID* | | *CVSS Score* |
| **Memory-Based Collaborative Filtering Techniques, Parameters and Results** | | | | | |
| ***CF Technique*** | ***Similarity Measure*** | ***# KNNs*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **Item-Based CF** | Pearson | 10 | 0.75366 | 0.34820 | 0.23856 |
| | | 20 | 0.75331 | 0.34656 | 0.23658 |
| | | 30 | 0.75331 | 0.34656 | 0.23658 |
| | | 40 | 0.75331 | 0.34656 | 0.23658 |
| | | 50 | 0.75331 | 0.34656 | 0.23658 |

**Table 6.2** Item-Based (Pearson) Top-N evaluation results for the second software / hardware vulnerability dataset

| **Dataset: 2** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Vendor \| Affected Software \| Edition \| Version* | | *CWE ID* | | *CVSS Score* |
| **Memory-Based Collaborative Filtering Techniques, Parameters and Results** | | | | | |
| ***CF Technique*** | ***Similarity Measure*** | ***# KNNs*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **Item-Based CF** | Pearson | 10 | 0.80339 | 0.37552 | 0.25556 |
| | | 20 | 0.80339 | 0.37552 | 0.25556 |
| | | 30 | 0.80339 | 0.37552 | 0.25556 |
| | | 40 | 0.80339 | 0.37552 | 0.25556 |
| | | 50 | 0.80339 | 0.37552 | 0.25556 |

**Table 6.3** Item-Based (Pearson) Top-N evaluation results for the third software / hardware vulnerability dataset

| **Dataset: 3** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Affected Software \| Edition \| Version* | | *CWE ID* | | *CVSS Score* |
| **Memory-Based Collaborative Filtering Techniques & Parameters** | | | | | |
| ***CF Technique*** | ***Similarity Measure*** | ***# KNNs*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **Item-Based CF** | Pearson | 10 | 0.80225 | 0.37419 | 0.25475 |
| | | 20 | 0.80225 | 0.37444 | 0.25508 |
| | | 30 | 0.80225 | 0.37444 | 0.25508 |
| | | 40 | 0.80225 | 0.37444 | 0.25508 |
| | | 50 | 0.80225 | 0.37444 | 0.25508 |

The highest AUC metric value was achieved through the second dataset, with a score of 0.80339. This value outperforms the first dataset and is marginally higher than that of the third dataset. The AUC ranges between 0.5 to 1, with a value of 0.5 indicating a random recommendation list and 1 signifying a perfect recommendation

list. The AUC metric value achieved through the Item-Based Collaborative Filtering which uses the Pearson's Correlation Coefficient obtains a notably high AUC metric score.

The NDCG and MAP evaluation metrics asses the position of relevant software and hardware vulnerabilities (representing items) within the Top-N (10) recommendation list of software and hardware products (representing users). In addition, both evaluation metrics range between the values of 0 to 1, where a higher value indicates a superior ranking effectiveness. The NDCG measures some specified number *n* of the topmost items located in the ranked recommendation list; whereas the MAP evaluates the average precision for the relevant items ranging from 1 to *n.* This metric assumes a user will examine the recommendation list from the first item and advance from one item to the next with certain probability, up until the end of the recommendation list (Lü et al., 2012). In relation to the NDCG and MAP, the highest metric values was obtained by means of the second dataset with results of 0.37552 and 0.25556. Comparable to that of the AUC metric value achieved, these results outperform the first dataset and are marginally greater than that obtained through the third dataset.

### 6.2.2 Experiment B: Item-Based Neighbourhood Size Sensitivity via Cosine Similarity Measure

The second experiment uses the same Memory-Based Collaborative Filtering technique (Item-Based) and sequence number of KNN value parameters (10, 20, 30, 40 and 50). In the previous experiment the Similarity Measure used was Pearson's Correlation Coefficient whereas in this experiment the Similarity Measure used is the Vector-Based Cosine similarity.

Shown below in Tables 6.4, 6.5 and 6.6 are the results for the Item-Based Neighbourhood Model Size Sensitivity which uses the Vector-Based Cosine Similarity Measure and is evaluated using the AUC, NDCG and MAP evaluation metrics for the three datasets.

**Table 6.4** Item-Based (Cosine) Top-N evaluation results for the first software / hardware vulnerability dataset

| **Dataset: 1** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Vendor \| Affected Software* | | *CWE ID* | | *CVSS Score* |
| **Memory-Based Collaborative Filtering Techniques, Parameters and Results** | | | | | |
| ***CF Technique*** | ***Similarity Measure*** | ***# KNNs*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **Item-Based CF** | Cosine | 10 | 0.69668 | 0.25800 | 0.17108 |
| | | 20 | 0.69707 | 0.26554 | 0.18066 |
| | | 30 | 0.69707 | 0.26554 | 0.18066 |
| | | 40 | 0.69707 | 0.26554 | 0.18066 |
| | | 50 | 0.69707 | 0.26554 | 0.18066 |

**Table 6.5** Item-Based (Cosine) Top-N evaluation results for the second software / hardware vulnerability dataset

| **Dataset: 2** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Vendor \| Affected Software \| Edition \| Version* | | *CWE ID* | | *CVSS Score* |
| **Memory-Based Collaborative Filtering Techniques, Parameters and Results** | | | | | |
| ***CF Technique*** | ***Similarity Measure*** | ***# KNNs*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **Item-Based CF** | Cosine | 10 | 0.71682 | 0.27588 | 0.18936 |
| | | 20 | 0.71684 | 0.27600 | 0.18950 |
| | | 30 | 0.71684 | 0.27600 | 0.18950 |
| | | 40 | 0.71684 | 0.27600 | 0.18950 |
| | | 50 | 0.71684 | 0.27600 | 0.18950 |

**Table 6.6** Item-Based (Cosine) Top-N evaluation results for the third software / hardware vulnerability dataset

| **Dataset: 3** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Affected Software \| Edition \| Version* | | *CWE ID* | | *CVSS Score* |
| **Memory-Based Collaborative Filtering Techniques & Parameters** | | | | | |
| ***CF Technique*** | ***Similarity Measure*** | ***# KNNs*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **Item-Based CF** | Cosine | 10 | 0.71607 | 0.27626 | 0.19040 |
| | | 20 | 0.71611 | 0.27634 | 0.19049 |
| | | 30 | 0.71611 | 0.27634 | 0.19049 |
| | | 40 | 0.71611 | 0.27634 | 0.19049 |
| | | 50 | 0.71611 | 0.27634 | 0.19049 |

The results shown in Tables 6.4 to 6.6 show the highest AUC value was obtained through the second dataset with a result of 0.71684. This result, although achieved using the same dataset as the previous experiment the AUC results of the Item-Based technique using the Cosine similarity measure were significantly lower to

that of the same techniques which used the Pearson's similarity measure. Therefore, this technique is not as successful at distinguishing items consisting of software and hardware vulnerabilities that are relevant to a user, where the user is recognised as the software and hardware product affected by the vulnerability through a completed and ranked recommendation list.

The highest NDCG and MAP metric scores were achieved when using the third dataset. The results in metric values were 0.27634 and 0.19049. Respectively comparing this to the results of the previous experiment were obtained through the second dataset. The results of this experiment show that the Item-Based technique using the Cosine similarity measure does not perform as well as the equivalent Collaborative Filtering technique using Pearson's similarity measure. This signifies that the NDCG and MAP ranked evaluation results for this Collaborative Filtering technique was not as successful in assessing the position of relevant software and hardware vulnerabilities (items) within the Top-N recommendation list ($N$ = 10) of software and hardware products (users) compared to that shown in the previous experiment.

### 6.2.3 Experiment C: User-Based Neighbourhood Size Sensitivity via Pearson's Correlation Coefficient Similarity Measure

Experiment C is similar to the first two experiments (Experiments A and B) in that the same sequence number of KNN value parameters are used, along with the Pearson's Correlation Coefficient and Vector-Based Cosine similarity measures. The alteration to this experiment however lies within the Recommender System algorithm used. In the previous experiments (Experiments A and B) the Item-Based Memory-Based Collaborative Filtering technique was used. For this experiment the Recommender System algorithm the User-Based technique is chosen.

Tables 6.7, 6.8 and 6.9 presents results in relation to the User-Based Memory-Based Collaborative Filtering Neighbourhood Model Size Sensitivity (KNN value parameters: 10, 20, 30, 40 and 50) using the Pearson's Correlation Coefficient.

**Table 6.7** User-Based (Pearson) Top-N evaluation results for the first software / hardware vulnerability dataset

| **Dataset: 1** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Vendor \| Affected Software* | | *CWE ID* | | *CVSS Score* |
| **Memory-Based Collaborative Filtering Techniques, Parameters and Results** | | | | | |
| ***CF Technique*** | ***Similarity Measure*** | ***# KNNs*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **User-Based CF** | Pearson | 10 | 0.78957 | 0.48368 | 0.39167 |
| | | 20 | 0.78141 | 0.47209 | 0.38604 |
| | | 30 | 0.77527 | 0.45815 | 0.37225 |
| | | 40 | 0.77276 | 0.44610 | 0.35855 |
| | | 50 | 0.77166 | 0.44446 | 0.35711 |

**Table 6.8** User-Based (Pearson) Top-N evaluation results for the second software / hardware vulnerability dataset

| **Dataset: 2** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Vendor \| Affected Software \| Edition \| Version* | | *CWE ID* | | *CVSS Score* |
| **Memory-Based Collaborative Filtering Techniques, Parameters and Results** | | | | | |
| ***CF Technique*** | ***Similarity Measure*** | ***# KNNs*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **User-Based CF** | Pearson | 10 | 0.82217 | 0.48406 | 0.37928 |
| | | 20 | 0.81535 | 0.48470 | 0.38608 |
| | | 30 | 0.81172 | 0.48036 | 0.38302 |
| | | 40 | 0.80983 | 0.47576 | 0.37833 |
| | | 50 | 0.80820 | 0.47338 | 0.37619 |

**Table 6.9** User-Based (Pearson) Top-N evaluation results for the third software / hardware vulnerability dataset

| **Dataset: 3** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Affected Software \| Edition \| Version* | | *CWE ID* | | *CVSS Score* |
| **Memory-Based Collaborative Filtering Techniques & Parameters** | | | | | |
| ***CF Technique*** | ***Similarity Measure*** | ***# KNNs*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **User-Based CF** | Pearson | 10 | 0.82134 | 0.48266 | 0.37791 |
| | | 20 | 0.81407 | 0.48364 | 0.38514 |
| | | 30 | 0.80988 | 0.47637 | 0.37903 |
| | | 40 | 0.80844 | 0.47224 | 0.37476 |
| | | 50 | 0.80641 | 0.46971 | 0.37272 |

The best AUC metric score was achieved using the second dataset with a value of 0.82217. This technique outperforms the Item-Based Pearson's and Cosine techniques from the past two experiments.

The highest values of the NDCG and MAP obtained were 0.48470 and 0.39167 of the second and first datasets. For the three datasets, the User-Based Pearson Collaborative Filtering technique shows an increase comparison with the Item-Based techniques, particularly when Pearson's Correlation Coefficient is used.

### 6.2.4 Experiment D: User-Based Neighbourhood Size Sensitivity via Cosine Similarity Measure

In Experiment C the User-Based Memory-Based Collaborative Filtering technique was introduced using a sequence number of KNN value parameters and similarity measure as parameters. In Experiment D, the same arrangement of KNN values are used (from 10 to 50, in increments of 10). The modification in relation to this experiment relates to the similarity measure used. In this experiment, Experiment D employs the Cosine similarity measure is used. The key parameter changes in both Memory-Based Collaborative Filtering techniques (Item-Based and User-Based) are the number of KNN values and Similarity Measure employed.

**Table 6.10** User-Based (Cosine) Top-N evaluation results for the first software / hardware vulnerability dataset

| **Dataset: 1** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Vendor \| Affected Software* | | *CWE ID* | | *CVSS Score* |
| **Memory-Based Collaborative Filtering Techniques, Parameters and Results** | | | | | |
| ***CF Technique*** | ***Similarity Measure*** | ***# KNNs*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **User-Based CF** | Cosine | 10 | 0.81105 | 0.46576 | 0.34552 |
| | | 20 | 0.81461 | 0.51499 | 0.40381 |
| | | 30 | 0.81461 | 0.52651 | 0.41734 |
| | | 40 | 0.81461 | 0.54135 | 0.43682 |
| | | 50 | 0.81461 | 0.54126 | 0.43707 |

**Table 6.11** User-Based (Cosine) Top-N evaluation results for the second software / hardware vulnerability dataset

| **Dataset: 2** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Vendor \| Affected Software \| Edition \| Version* | | *CWE ID* | | *CVSS Score* |
| **Memory-Based Collaborative Filtering Techniques, Parameters and Results** | | | | | |
| ***CF Technique*** | ***Similarity Measure*** | ***# KNNs*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **User-Based CF** | Cosine | 10 | 0.83650 | 0.44448 | 0.31529 |
| | | 20 | 0.83696 | 0.46605 | 0.34142 |
| | | 30 | 0.83704 | 0.48100 | 0.36008 |
| | | 40 | 0.83704 | 0.48365 | 0.36303 |
| | | 50 | 0.83704 | 0.49709 | 0.38024 |

**Table 6.12** User-Based (Cosine) Top-N evaluation results for the first software / hardware vulnerability dataset

| **Dataset: 3** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Affected Software \| Edition \| Version* | | *CWE ID* | | *CVSS Score* |
| **Memory-Based Collaborative Filtering Techniques & Parameters** | | | | | |
| ***CF Technique*** | ***Similarity Measure*** | ***# KNNs*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **User-Based CF** | Cosine | 10 | 0.83606 | 0.44590 | 0.31756 |
| | | 20 | 0.83675 | 0.46492 | 0.34048 |
| | | 30 | 0.83675 | 0.47718 | 0.35559 |
| | | 40 | 0.83675 | 0.48370 | 0.36368 |
| | | 50 | 0.83675 | 0.49716 | 0.38090 |

As identified through the evaluation results displayed through Tables 6.10 to 6.12, the dataset which achieved the highest AUC was the second dataset with a value of 0.83704. This evaluation score is marginally greater than that of the previous experiment.

The highest values of the NDCG and MAP obtained were 0.54135 and 0.43707 by use of the first dataset. The User-Based Cosine technique not only outperforms the Item-Based approaches, but also the User-Based Pearson technique.

## 6.3 Sparse Linear Methods (SLIM) Top-N Collaborative Filtering via regularisation Parameter Alteration

In the experiments presented in Section 6.2 the traditional Collaborative Filtering Techniques User-Based and Item-Based were implemented. The two parameters of key interest in those experiments were, the type of similarity measure used, and the number of neighbourhood (KNN) values.

Comparable to the experiments undertaken in Section 6.2, the following SLIM experiments applied 5 K-Fold Cross Validation to evaluate to the accuracy performance of the SLIM technique, along with the top 10 (Top-N) chosen items are to be recommended for a user. Additionally, for the evaluation Top-N recommendation accuracy of the SLIM technique, the AUC, NDCG and MAP evaluation metrics were used.

Shown below in Tables 6.13, 6.14 and 6.15 are the results of the SLIM technique through the alteration of the l1-norm (λ) and l2-norm (β) regularisation hyper-parameters. These parameters values were identified and obtained through the work of (Kabbur, Ning and Karypis, 2013).

**Table 6.13** Sparse Linear Methods (SLIM) Top-N evaluation results for the first software / hardware vulnerability dataset

| **Dataset: 1** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Vendor \| Affected Software* | | *CWE ID* | | *CVSS Score* |
| **SLIM Collaborative Filtering Technique, Parameters & Results** | | | | | |
| ***CF Technique*** | ***Reg l1 (λ)*** | ***Reg l2 (β)*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **SLIM** | 20 | 0.1 | 0.79015 | 0.45601 | 0.36220 |
| | 18 | 0.01 | 0.79044 | 0.45914 | 0.36596 |
| | 14 | 0.0001 | 0.78757 | 0.45383 | 0.36135 |
| | 8 | 0.001 | 0.79279 | 0.48223 | 0.39043 |
| | 8 | 0.1 | 0.79279 | 0.48257 | 0.39089 |
| | 12 | 0.0001 | 0.79039 | 0.45836 | 0.36481 |
| | 12 | 0.1 | 0.79043 | 0.45891 | 0.36536 |
| | 12 | 0.001 | 0.79039 | 0.45836 | 0.36481 |
| | 2 | 0.1 | 0.80240 | 0.51424 | 0.41213 |

**Table 6.14** Sparse Linear Methods (SLIM) Top-N evaluation results for the second software / hardware vulnerability dataset

| **Dataset: 2** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Vendor \| Affected Software \| Edition \| Version* | | *CWE ID* | | *CVSS Score* |
| **SLIM Collaborative Filtering Technique, Parameters & Results** | | | | | |
| ***CF Technique*** | ***Reg l1 (λ)*** | ***Reg l2 (β)*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **SLIM** | 20 | 0.1 | 0.82347 | 0.48336 | 0.38194 |
| | 18 | 0.01 | 0.82347 | 0.48367 | 0.38236 |
| | 14 | 0.0001 | 0.82346 | 0.48401 | 0.38277 |
| | 8 | 0.001 | 0.82589 | 0.49459 | 0.39411 |
| | 8 | 0.1 | 0.82589 | 0.49459 | 0.39411 |
| | 12 | 0.0001 | 0.82348 | 0.48525 | 0.38427 |
| | 12 | 0.1 | 0.82348 | 0.48525 | 0.38427 |
| | 12 | 0.001 | 0.82348 | 0.48525 | 0.38427 |
| | 2 | 0.1 | 0.83408 | 0.51740 | 0.41241 |

**Table 6.15** Sparse Linear Methods (SLIM) Top-N evaluation results for the third software / hardware vulnerability dataset

| **Dataset: 3** | **User** | | **Item** | | **Rating** |
|---|---|---|---|---|---|
| | *Affected Software \| Edition \| Version* | | *CWE ID* | | *CVSS Score* |
| **SLIM Collaborative Filtering Technique, Parameters & Results** | | | | | |
| ***CF Technique*** | ***Reg l1 (λ)*** | ***Reg l2 (β)*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **SLIM** | 20 | 0.1 | 0.82336 | 0.48202 | 0.38037 |
| | 18 | 0.01 | 0.82334 | 0.48248 | 0.38095 |
| | 14 | 0.0001 | 0.82336 | 0.48178 | 0.37997 |
| | 8 | 0.001 | 0.82652 | 0.49602 | 0.39556 |
| | 8 | 0.1 | 0.82652 | 0.49602 | 0.39556 |
| | 12 | 0.0001 | 0.82338 | 0.48313 | 0.38176 |
| | 12 | 0.1 | 0.82338 | 0.48313 | 0.38176 |
| | 12 | 0.001 | 0.82338 | 0.48313 | 0.38176 |
| | 2 | 0.1 | 0.83423 | 0.51723 | 0.41272 |

The highest AUC metric score was achieved using the third dataset with a value of 0.83423. This evaluation score is lower than that obtained through the User-Based Cosine technique. However, this result is higher than that achieved by the User-Based Pearson approach and both Item-Based techniques. The NDCG and MAP evaluation metrics show that the highest results achieved were 0.51740 and 0.41272 using the second and third datasets. This technique obtained lower NDCG and MAP scores compared to the User-Based Cosine technique (using the first dataset), but greater than User-Based Pearson approach and both Item-Based techniques.

## 6.4 Factored Item Similarity Models (FISM) Top-N Collaborative Filtering via Parameter Alteration

In the previous set of experiments, three categories of Collaborative Filtering techniques were used. In Section 6.1 the User-Based and Item-Based Memory-Based techniques were employed. The experiments consisted of evaluating the classification and ranking accuracy of the neighbourhood model size sensitivity through various similarity measures. Following these Memory-Based experiments Section 6.2 employed the SLIM technique. The experiments comprised of evaluating the accuracy through the alteration of the l1-norm (λ) and l2-norm (β) regularisation hyper-parameters as identified by (Kabbur, Ning and Karypis, 2013).

For these experiments the Factored Item Similarity Models (FISM) Collaborative Filtering technique was used. Similar to the experiments presented in Section 6.1 and Section 6.2 these experiments applied 5 K-Fold Cross Validation to evaluate to the accuracy performance of the FISM techniques, along with the top 10 (Top-N) chosen items are to be recommended for a user. Additionally, for the evaluation Top-N recommendation accuracy of both FISM techniques, the evaluation metrics were used for the three datasets.

The results of the FISMrmse technique through the alteration of the latent factor (k), regularisation weight (β) and learning rate (n) hyper-parameters are shown in Tables 6.16, 6.17 and 6.18. The results of the FISMauc technique through the alteration of these hyper-parameters are shown in Tables 6.19, 6.20 and 6.21.

**Table 6.16** Factored Item Similarity Models (FISM), FISMrmse Top-N evaluation results for the first software / hardware vulnerability dataset

| **Dataset: 1** | **User** | | **Item** | | **Rating** | |
|---|---|---|---|---|---|---|
| | *Vendor \| Affected Software* | | *CWE ID* | | *CVSS Score* | |
| **FISMrmse Collaborative Filtering Technique, Parameters & Results** | | | | | | |
| ***CF Technique*** | ***Latent Factors (k)*** | ***Regularisation Weight (β)*** | ***Learning Rate (n)*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **FISMrmse** | 96 | 0.00002 | 0.001 | 0.82795 | 0.56361 | 0.44613 |
| | 64 | 0.0008 | 0.01 | 0.82802 | 0.56458 | 0.44710 |
| | 96 | 0.0008 | 0.001 | 0.82795 | 0.56361 | 0.44613 |
| | 192 | 0.00002 | 0.001 | 0.82792 | 0.56399 | 0.44635 |
| | 192 | 0.00006 | 0.001 | 0.82792 | 0.56399 | 0.44635 |
| | 128 | 0.00006 | 0.001 | 0.82820 | 0.56440 | 0.44706 |
| | 192 | 0.0001 | 0.001 | 0.82792 | 0.56399 | 0.44635 |
| | 192 | 0.00002 | 0.0005 | 0.82792 | 0.56399 | 0.44635 |
| | 160 | 0.002 | 0.001 | 0.82804 | 0.56452 | 0.44676 |

**Table 6.17** Factored Item Similarity Models (FISM), FISMrmse Top-N evaluation results for the second software / hardware vulnerability dataset

| **Dataset: 2** | **User** | | **Item** | | **Rating** | |
|---|---|---|---|---|---|---|
| | *Vendor \| Affected Software \| Edition \| Version* | | *CWE ID* | | *CVSS Score* | |
| **FISMrmse Collaborative Filtering Technique, Parameters & Results** | | | | | | |
| ***CF Technique*** | ***Latent Factors (k)*** | ***Regularisation Weight (β)*** | ***Learning Rate (n)*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **FISMrmse** | 96 | 0.00002 | 0.001 | 0.85359 | 0.55260 | 0.43165 |
| | 64 | 0.0008 | 0.01 | 0.85343 | 0.55243 | 0.43153 |
| | 96 | 0.0008 | 0.001 | 0.85359 | 0.55260 | 0.43165 |
| | 192 | 0.00002 | 0.001 | 0.85372 | 0.55217 | 0.43140 |
| | 192 | 0.00006 | 0.001 | 0.85372 | 0.55217 | 0.43140 |
| | 128 | 0.00006 | 0.001 | 0.85347 | 0.55228 | 0.43146 |
| | 192 | 0.0001 | 0.001 | 0.85372 | 0.55217 | 0.43140 |
| | 192 | 0.00002 | 0.0005 | 0.85372 | 0.55217 | 0.43140 |
| | 160 | 0.002 | 0.001 | 0.85415 | 0.55272 | 0.43168 |

**Table 6.18** Factored Item Similarity Models (FISM), FISMrmse Top-N evaluation results for the third software / hardware vulnerability dataset

| **Dataset: 3** | **User** | | **Item** | | **Rating** | |
|---|---|---|---|---|---|---|
| | *Affected Software \| Edition \| Version* | | *CWE ID* | | *CVSS Score* | |
| **FISMrmse Collaborative Filtering Technique, Parameters & Results** | | | | | | |
| ***CF Technique*** | ***Latent Factors (k)*** | ***Regularisation Weight (β)*** | ***Learning Rate (n)*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **FISMrmse** | 96 | 0.00002 | 0.001 | 0.85337 | 0.55194 | 0.43089 |
| | 64 | 0.0008 | 0.01 | 0.85263 | 0.55160 | 0.43072 |
| | 96 | 0.0008 | 0.001 | 0.85337 | 0.55194 | 0.43089 |
| | 192 | 0.00002 | 0.001 | 0.85375 | 0.55249 | 0.43116 |
| | 192 | 0.00006 | 0.001 | 0.85375 | 0.55249 | 0.43116 |
| | 128 | 0.00006 | 0.001 | 0.85274 | 0.55188 | 0.43081 |
| | 192 | 0.0001 | 0.001 | 0.85375 | 0.55249 | 0.43116 |
| | 192 | 0.00002 | 0.0005 | 0.85375 | 0.55249 | 0.43116 |
| | 160 | 0.002 | 0.001 | 0.85348 | 0.55228 | 0.43098 |

As identified through the results of Tables 6.16 to 6.18 the dataset which achieved the highest AUC was 0.85415 through the second dataset. This evaluation metric score shows an increase by any Top-N Collaborative Filtering technique as presented in the previous experiments.

Through the NDCG and MAP Top-N metrics, it was identified that the highest results were achieved through the first dataset, with values of 0.56458 and 0.44710. Similar to AUC result, the FISMrmse Collaborative Filtering technique obtained greater NDCG and MAP evaluation values to that presented in the previous experiments.

**Table 6.19** Factored Item Similarity Models (FISM), FISMauc Top-N evaluation results for the first software / hardware vulnerability dataset

| **Dataset: 1** | **User** | | **Item** | | **Rating** | |
|---|---|---|---|---|---|---|
| | *Vendor \| Affected Software* | | *CWE ID* | | *CVSS Score* | |
| **FISMauc Collaborative Filtering Technique, Parameters & Results** | | | | | | |
| ***CF Technique*** | ***Latent Factors (k)*** | ***Regularisation Weight (β)*** | ***Learning Rate (n)*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **FISMauc** | 64 | 0.001 | 0.0001 | 0.82820 | 0.55658 | 0.44108 |
| | 144 | 0.00002 | 0.00005 | 0.82736 | 0.55575 | 0.44030 |
| | 144 | 0.00008 | 0.00001 | 0.82127 | 0.53053 | 0.41655 |
| | 192 | 0.00001 | 0.0001 | 0.82859 | 0.56438 | 0.44969 |
| | 240 | 0.00002 | 0.0001 | 0.82587 | 0.55668 | 0.44072 |
| | 160 | 0.0004 | 0.0005 | 0.82792 | 0.55528 | 0.43782 |
| | 144 | 0.00008 | 0.0001 | 0.82639 | 0.55555 | 0.43969 |
| | 160 | 0.00002 | 0.0005 | 0.82792 | 0.55528 | 0.43782 |
| | 176 | 0.0002 | 0.001 | 0.82590 | 0.53802 | 0.41833 |

**Table 6.20** Factored Item Similarity Models (FISM), FISMauc Top-N evaluation results for the second software / hardware vulnerability dataset

| **Dataset: 2** | **User** | | **Item** | | **Rating** | |
|---|---|---|---|---|---|---|
| | *Vendor \| Affected Software \| Edition \| Version* | | *CWE ID* | | *CVSS Score* | |
| **FISMauc Collaborative Filtering Technique, Parameters & Results** | | | | | | |
| ***CF Technique*** | ***Latent Factors (k)*** | ***Regularisation Weight (β)*** | ***Learning Rate (n)*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **FISMauc** | 64 | 0.001 | 0.0001 | 0.85342 | 0.55180 | 0.43140 |
| | 144 | 0.00002 | 0.00005 | 0.85392 | 0.55199 | 0.43157 |
| | 144 | 0.00008 | 0.00001 | 0.85236 | 0.54816 | 0.42900 |
| | 192 | 0.00001 | 0.0001 | 0.85438 | 0.55280 | 0.43169 |
| | 240 | 0.00002 | 0.0001 | 0.85349 | 0.55224 | 0.43147 |
| | 160 | 0.0004 | 0.0005 | 0.85366 | 0.55184 | 0.43155 |
| | 144 | 0.00008 | 0.0001 | 0.85382 | 0.55249 | 0.43201 |
| | 160 | 0.00002 | 0.0005 | 0.85366 | 0.55184 | 0.43155 |
| | 176 | 0.0002 | 0.001 | 0.85532 | 0.52700 | 0.39961 |

**Table 6.21** Factored Item Similarity Models (FISM), FISMauc Top-N evaluation results for the third software / hardware vulnerability dataset

| **Dataset: 3** | **User** | | **Item** | | **Rating** | |
|---|---|---|---|---|---|---|
| | *Affected Software \| Edition \| Version* | | *CWE ID* | | *CVSS Score* | |
| **FISMauc Collaborative Filtering Technique, Parameters & Results** | | | | | | |
| ***CF Technique*** | ***Latent Factors (k)*** | ***Regularisation Weight (β)*** | ***Learning Rate (n)*** | ***AUC*** | ***NDCG*** | ***MAP*** |
| **FISMauc** | 64 | 0.001 | 0.0001 | 0.85387 | 0.55249 | 0.43190 |
| | 144 | 0.00002 | 0.00005 | 0.85397 | 0.55156 | 0.42996 |
| | 144 | 0.00008 | 0.00001 | 0.84992 | 0.54552 | 0.42672 |
| | 192 | 0.00001 | 0.0001 | 0.85390 | 0.55261 | 0.43169 |
| | 240 | 0.00002 | 0.0001 | 0.85349 | 0.55228 | 0.43159 |
| | 160 | 0.0004 | 0.0005 | 0.85249 | 0.55132 | 0.43068 |
| | 144 | 0.00008 | 0.0001 | 0.85381 | 0.55169 | 0.43004 |
| | 160 | 0.00002 | 0.0005 | 0.85249 | 0.55132 | 0.43068 |
| | 176 | 0.0002 | 0.001 | 0.85478 | 0.52196 | 0.39383 |

The highest AUC achieved through the second dataset with a score of 0.85532. It was observed that this result is greater than that of the FISMrmse technique which was also obtained through the second dataset.

The highest NDCG and MAP results were achieved by means of the first dataset, with values of 0.56438 and 0.44969. The FISMauc technique obtained a higher NDCG evaluation score to that of the FISMrmse approach, however this technique did achieve a slightly lower MAP result.

## 6.5 Summary

In this chapter Top-N Collaborative Filtering techniques were investigated, through the use of three software and hardware vulnerability cyber datasets. The traditional Memory-Based Collaborative Filtering approaches of User-Based and Item-Based were employed using the similarity measures of Pearson's Correlation Coefficient and Cosine. State-of-the-art Top-N Collaborative Filtering techniques were also implemented to investigate the quality of recommendation results using evaluation metrics of AUC, NDCG and MAP.

The next chapter discusses the experimental results presented in this chapter in more detail.

# Chapter 7 Discussion

## 7.1 Introduction

As recommender system can be divided into three main techniques: Collaborative Filtering, Content-Based Filtering and Hybrid Filtering, this research project focuses on the Collaborative Filtering technique. As identified in chapter 4, Collaborative Filtering is recognised as the most widely used recommender system technique. In addition, this commonly used technique can be further divided into two sub-categories of Memory-Based and Model-Based Collaborative Filtering.

Section 7.2 of this chapter will discuss the key findings form this study through a series of research questions. Section 7.3 concludes this chapter with a summary.

## 7.2 Key Findings

For this research project both Memory-Based and Model-Based techniques were used within a Top-N (Recommendation) Collaborative Filtering approach. As presented in chapter 6, the Memory-Based techniques of User-Based and Item-Based were used. These employed the two Similarity Measures of Pearson's Correlation Coefficient (PCC) and Cosine (COS) Vector-Based similarity through the variance of the Neighbourhood Size Sensitivity. In addition, two state-of-the-art Top-N Collaborative Filtering technique were used: Sparse Linear Methods (SLIM) and Factored Item Similarity Models (FISM). The above Collaborative Filtering techniques were trained and tested using the three User-Item-Rating cyber security datasets. To reiterate these three datasets are shown below in Table 7.1.

**Table 7.1** User-Item-Rating Datasets and Attributes

| **Datasets** | | | |
|---|---|---|---|
| **#** | **User** | **Item** | **Rating** |
| **1** | Vendor \| Affected Software<br>**606** | CWE ID<br>**30** | CVSS Score<br>**7484** |
| **2** | Vendor \| Affected Software \| Edition \| Version<br>**1960** | CWE ID<br>**30** | CVSS Score<br>**7484** |
| **3** | Affected Software \| Edition \| Version<br>**1959** | CWE ID<br>**30** | CVSS Score<br>**7484** |

The Top-N recommendation accuracy for the employed Collaborative Filtering techniques were evaluated using the Area Under the Curve (AUC), Normalised Discounted Cumulative Gain (NDCG) and the Mean Absolute Precision (MAP).

The following sections answer the research questions set out in this research project. In addition, to compare the outcomes of the following research questions statistical tests were performed. The statistical tests compared the averages obtained from the five-fold cross validation of the Collaborative Filtering techniques used. For these tests the paired t-test was used with a confidence interval of 95% (Deshpande and Karypis, 2004). The paired t-test observes the average difference among the performance scores of algorithms A and algorithm B which is normalised through the standard deviation of the score variance (Ricci et al., 2011).

### 7.2.1 Research Question 1

*What similarity measure achieves the highest Top-N accuracy using the Item-Based Collaborative Filtering technique through the three software and hardware vulnerability datasets?*

Results for experiment A show that the second software and hardware vulnerability dataset employed into the Item-Based Pearson's Correlation Coefficient technique achieved the highest level of accuracy for all three-evaluation metrics.

Results from experiment B show that the second dataset obtained the highest AUC accuracy through the Item-Based technique using the Cosine similarity measure. For both the NDCG and MAP evaluation metrics the third dataset achieved the highest accuracy.

In answering the first research question the Item-Based technique using Pearson's Correlation Coefficient achieved the highest AUC, NDCG and MAP evaluation results. These three highest Top-N recommendation results were achieved using the second software and hardware vulnerability dataset.

The AUC scores of the Item-Based Pearson's Correlation Coefficient compared to the Item-Based Cosine obtained a paired t-test p-value of less than 0.0001, making the difference to be statistically significant at a 95% confidence interval. The NDCG scores obtained a p-value of less than 0.0001 which considers the

difference to be statistically significant at a 95% confidence interval. The MAP scores obtained a p-value of less than 0.0001, meaning that the difference is expressed to be statistically significant at a 95% confidence interval.

### 7.2.2 Research Question 2

*What similarity measure achieves the highest Top-N accuracy using the User-Based Collaborative Filtering technique through the three software and hardware vulnerability datasets?*

Results from experiment C show that the second software and hardware vulnerability dataset achieved the highest AUC Top-N evaluation accuracy. The second dataset also achieved the highest NDCG evaluation accuracy. The highest MAP accuracy however was obtained through the first dataset. Additionally, the User-Based technique using Pearson's Correlation Coefficient achieves improved results compared to the Item-Based techniques (Pearson's and Cosine similarity measures) of experiments A and B.

Results from experiment D identifies that the second dataset obtained the highest AUC evaluation accuracy. In relation to the NDCG and MAP evaluation metrics, the highest scores where achieved through the first dataset.

In answering the second research question the User-Based using the Cosine Vector-Based similarity measure technique achieved the highest AUC result. The highest NDCG and MAP results were also achieved using the Cosine similarity measure. In addition, the highest AUC result was obtained through the second dataset, whereas both the highest NDCG and MAP evaluation results were achieved through the first dataset.

The AUC scores of the User-Based Cosine compared to the Item-Based Pearson's Correlation Coefficient obtained a paired t-test p-value of 0.0008, making the difference to be statistically significant at a 95% confidence interval. The NDCG scores obtained a p-value of 0.0715 which considers the difference to be not statistically significant at a 95% confidence interval. The MAP scores obtained a p-value of 0.2079, meaning that the difference is expressed to be not statistically significant at a 95% confidence interval.

### 7.2.3 Research Question 3

*What Memory-Based Collaborative Filtering technique achieves the highest Top-N accuracy through the three software and hardware vulnerability datasets?*

In answering the third research question, both the Item-Based and User-Based Memory-Based Collaborative Filtering techniques from research questions one and two were observed. For this research question the User-Based technique which used the Cosine similarity measure achieved the highest Top-N evaluation accuracy across all three metrics. Not only does it obtain better results than the User-Based technique which uses Pearson's Correlation Coefficient but improves on the Item-Based technique (for both similarity measures) which employ the three software and hardware vulnerability datasets for all three Top-N evaluation metrics.

The AUC scores of the User-Based Cosine compared to the Item-Based Pearson's Correlation Coefficient obtained a paired t-test p-value of less than 0.0001, making the difference to be statistically significant at a 95% confidence interval. The NDCG scores obtained a p-value of 0.0005 which considers the difference to be statistically significant at a 95% confidence interval. The MAP scores obtained a p-value of 0.0008, meaning that the difference is expressed to be statistically significant at a 95% confidence interval.

The AUC scores of the User-Based Cosine compared to the Item-Based Cosine obtained a paired t-test p-value of less than 0.0001, making the difference to be statistically significant at a 95% confidence interval. The NDCG scores obtained a p-value of less than 0.0001 which considers the difference to be statistically significant at a 95% confidence interval. The MAP scores obtained a p-value of 0.0002, meaning that the difference is expressed to be statistically significant at a 95% confidence interval.

### 7.2.4 Research Question 4

*Does the use of state-of-the-art Top-N Collaborative Filtering techniques outperform Memory-Based techniques through the employment of the three software and hardware vulnerability datasets?*

The fourth research question puts primary focus on the more advanced Collaborative Filtering techniques, compared to the traditional Memory-Based techniques. This research question puts focus on the Sparse Linear Methods (SLIM) and Factored Item Similarity Models (FISM). These state-of-the-art Collaborative Filtering techniques focus on the Top-N recommendation problem which is said to improve on the alternative Collaborative Filtering techniques.

#### 7.2.4.1 SLIM Top-N Collaborative Filtering Technique

The results of the SLIM technique show that the highest AUC Top-N evaluation accuracy was obtained through the third software and hardware vulnerability dataset. In contrast the User-Based Cosine technique achieved the highest AUC result through the second software and hardware dataset.

The highest NDCG Top-N accuracy was achieved through the second dataset. Through the third research question the User-Based Cosine technique obtained the highest NDCG evaluation accuracy through the first dataset. This NDCG evaluation score is less than what was achieved through the User-Based technique using the Cosine Vector-Based similarity measure.

The highest MAP evaluation accuracy result was achieved by means of the third dataset. As acknowledged through the third research question, that the User-Based Cosine technique obtained the highest MAP accuracy through the first dataset.

It was identified through the Top-N evaluation results that the User-Based approach using the Cosine Vector-Based similarity measure achieved the highest AUC, NDCG and MAP evaluation results to that of the SLIM technique. However, the Top-N evaluation results for the User-Based Cosine technique are close to that of the SLIM technique, for all the three software and hardware datasets. The User-Based and Item-Based techniques using Pearson's Correlation Coefficient similarity measure also obtained lower results for all the three datasets. The Item-Based technique that implemented the Cosine similarity measure however achieved the lowest Top-N

results. This technique (Item-Based Cosine) achieved lower Top-N accuracy results than any of the Memory-Based techniques and SLIM techniques, throughout the three software and hardware vulnerability datasets.

Through the results of (Ning and Karypis, 2011), the SLIM technique achieved greater results than alternative Top-N Collaborative Filtering approaches. This technique increased the recommendation accuracy of the traditional User-Based and Item-Based Top-N approaches. As identified in this research, the SLIM technique achieved higher accuracy results than that of the Item-Based and User-Based Pearson approaches, with only the User-Based Cosine technique achieving similar outcomes to that of the SLIM Top-N technique.

The AUC scores of SLIM compared to User-Based Cosine obtained a paired t-test p-value of less than 0.0001, making the difference to be statistically significant at a 95% confidence interval. The NDCG scores obtained a p-value of 0.0704 which considers the difference to be not statistically significant at a 95% confidence interval. The MAP scores obtained a p-value of 0.2107, meaning that the difference is expressed to be not statistically significant at a 95% confidence interval.

#### 7.2.4.2 FISM Top-N Collaborative Filtering Technique

This research question continues by observing the Top-N evaluation accuracy results for the state-of-the-art Factored Item Similarity Models (FISM) Collaborative Filtering technique. For this research question, the two FISM techniques of FISMrmse and FISMauc were employed.

The results of the FISMrmse technique identify that the highest AUC evaluation accuracy was achieved through the second software and hardware vulnerability dataset. This technique improved the AUC accuracy of both the User-Based Cosine and SLIM techniques. Additionally, it was observed that the AUC results for both the second and third software and hardware datasets showed change throughout. However, the first dataset obtained the lowest AUC results throughout.

The NDCG and MAP Top-N evaluation results show that the first software and hardware dataset achieved the highest evaluation scores for both metrics. In relation to both Top-N evaluation metrics. For both metric results it was observed that there was very little difference throughout the three datasets. The FISMrmse technique

improved the NDCG and MAP Top-N accuracy to that of the User-Based Cosine and SLIM techniques.

The results of the second FISM Top-N Collaborative Filtering technique FISMauc identifies that the highest AUC evaluation accuracy was achieved through the second software and hardware dataset. The FISMauc technique in contrast to the FISMrmse approach achieved improved AUC results. Similar to the AUC evaluation results for the FISMrmse approach, the AUC results for the FISMauc technique identify that there is little variance shown in the second and third datasets throughout. In addition to the FISMrmse approach, the first dataset obtained the lowest AUC result.

For the NDCG and MAP Top-N evaluation metrics for the FISMauc technique results show that the highest NDCG and MAP was achieved through the first software and hardware dataset. These results show a small difference throughout the three software and hardware datasets. Through observing these evaluation results to those of the FISMrmse, it is identified that the results are similar for the three datasets.

In answering the fourth research question, the Top-N evaluation metric results identify that the FISM technique performed better than that of the SLIM technique. Results show that the FISM technique increased the AUC, NDCG and MAP for all the three software and hardware vulnerability datasets, in contrast to the SLIM Top-N technique. In relation to the two FISM technique, the notable AUC results occurred in the second and third datasets, with the highest been achieved through the FISMauc approach (second dataset), in contrast to the SLIM technique which was obtained through the third dataset. The NDCG and MAP evaluation results shows little change towards the three datasets for both FISM techniques. The highest NDCG score was achieved though the FISMrmse technique (first dataset) and the highest MAP score was achieved through the FISMauc approach (first dataset).

The FISM technique similar to the SLIM approach out-performed the User-Based and Item-Based techniques in regards Top-N Collaborative Filtering. As previously identified through the first three research questions, the User-Based technique which using the Cosine Vector-Based similarity measure achieved the highest evaluation results than any other Memory-Based technique. However, results from both the User-Based Cosine and SLIM techniques fall short of the Top-N

evaluation metrics scores obtained through the FISM Collaborative Filtering technique.

Through results of (Kabbur, Ning and Karypis, 2013), the FISM technique achieved improved results than that of the SLIM technique (Ning and Karypis, 2011) and alternative Collaborative Filtering approaches. Additionally, the respective performance gains improves with the sparsity of the datasets, as shown through the experiments of (Kabbur, Ning and Karypis, 2013). As identified in this research, the FISM technique achieved higher Top-N accuracy results than that of the SLIM technique and the traditional approaches of Item-Based and User-Based which is shown through the research of (Kabbur, Ning and Karypis, 2013).

The AUC scores of FISM compared to the User-Based Cosine obtained a paired t-test p-value of less than 0.0001, making the difference to be statistically significant at a 95% confidence interval. The NDCG scores obtained a p-value of 0.0298 which considers the difference to be statistically significant at a 95% confidence interval. The MAP scores obtained a p-value of 0.1705, meaning that the difference is expressed to be not statistically significant at a 95% confidence interval.

The AUC scores of FISM compared to SLIM obtained a paired t-test p-value of less than 0.0001, making the difference to be statistically significant at a 95% confidence interval. The NDCG scores obtained a p-value of less than 0.0001 which considers the difference to be statistically significant at a 95% confidence interval. The MAP scores also obtained a p-value of less than 0.0001, meaning that the difference is expressed to be statistically significant at a 95% confidence interval.

## 7.3 Summary

In this chapter, a discussion of the experimental results from chapter 6 is presented. The experimental results shown in this chapter were distinguished through numerous research questions. The first three research question put focus on the Top-N Memory-Based Collaborative Filtering techniques of Item-Based and User-Based. These traditional approaches employed the similarity measures of Pearson's Correlation Coefficient and Cosine, using software and hardware vulnerability user-item-rating data. For the three Top-N evaluation metrics AUC, NDCG and MAP. The

User-Based Cosine technique achieved the highest level of Top-N evaluation accuracy across all three metrics for the three software and hardware vulnerability datasets.

This chapter also put focus on state-of-the-art Top-N Collaborative Filtering techniques. In this study, it was investigated that the FISM approach achieved greater recommendation evaluation results to that of the SLIM approach and User-Based Cosine technique. The FISM technique accomplished the highest Top-N evaluation accuracy for all three Top-N metrics for the three software and hardware vulnerability datasets. In addition, this study shows that the use of state-of-the-art Top-N Collaborative Filtering technique outperform the traditional Memory-Based approaches in the recommendation of vulnerabilities which affect software and hardware, grounded on the similarity of software and hardware systems, its related vulnerability and its associated rating score.

# Chapter 8 Conclusion and Suggestions for Future Work

The FISM Top-N Collaborative Filtering technique through the results in chapter 6 not only out-performed the traditional Memory-Based approaches (User-Based and Item-Based) but also the state-of-the-art SLIM Top-N technique for all the three user-item-rating software and hardware datasets constructed through attributes of the National Vulnerability Database (NVD). Additionally, it was also observed that the User-Based technique which employed the Cosine Vector-Based similarity measure obtained greater Top-N evaluation results than that of the alternative Memory-Based approaches.

The FISM Top-N evaluation results obtained through the NVD datasets used in this research project offers security personnel the highest level of Top-N accuracy achieved which relate to vulnerabilities affecting software and hardware assets. In other words, this provides security personnel a Top-N recommendation list of software and hardware vulnerabilities based on the similarity of a vulnerable asset, its vulnerability type and accompanying vulnerability severity score. Through the various evaluation metrics used different conclusions were presented, in relation to the Top-N recommendation list.

In addition to the methods and techniques used in the research project, areas of where future work can be applied are as follows:

- Implementation of additional software and hardware product (user) information. Incorporation of this data into the present datasets may increase Top-N results.
- Employment of alternative Similarity Measures used within the Memory-Based Collaborative Filtering techniques (User-Based and Item-Based). This is to observe if the implemented of various similarity measures will improve the Top-N evaluation metric results, relating to the three software and hardware datasets.
- Implementation of alternative state-of-the-art Top-N Model-Based Collaborative Filtering techniques. Due to the extensive variety of Top-

N approaches, further research and investigating will be undertaken to explore if the employment of such techniques will improve the Top-N evaluation achieved through this research.

- Implementation of diverse Top-N evaluation metrics, both relating to classification-based and ranked-based used for assessing various observations through the recommendation list.
- Use of various data splitting approaches. In this research the K-Fold Cross Validation (CV) was implemented. Other approaches which can be applied include Leave-One-Out CV

# References

A. Phillips, E. and M. Davis, E., 2006. *Request for Comments: 4646 - Tags for Identifying Languages*. Available at: <https://www.ietf.org/rfc/rfc4646.txt>.

Adomavicius, G., Manouselis, N. and Kwon, Y.O., 2015. Multi-Criteria Recommender Systems. In: *Recommender Systems Handbook*, Second Edi. [online] pp.769–798. Available at: <https://www.researchgate.net/publication/226204259_Multi-Criteria_Recommender_Systems>.

Ahmad, K., Vivekananda, S. and Pradesh, U., 2011. Classification Of Internet Security Attacks. In: *5th National Conference; INDIACom-2011 Computing For Nation Development*. [online] pp.1–4. Available at: <https://www.researchgate.net/publication/262494946_Classification_of_Internet_Security_Attacks>.

Ahmed, M.S., Al-Shaer, E. and Khan, L., 2008. A Novel Quantitative Approach For Measuring Network Security. In: *The 27th Conference on Computer Communications. IEEE INFOCOM 2008.* [online] pp.76–80. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/4509855/>.

Allodi, L. and Massacci, F., 2014. Comparing Vulnerability Severity And Exploits Using Case-Control Studies. *ACM Transactions on Information and System Security (TISSEC)*, [online] 17(1), pp.1–20. Available at: <https://dl.acm.org/citation.cfm?id=2630069>.

Arsan, T., Koksal, E. and Bozkus, Z., 2016. Comparison Of Collaborative Filtering Algorithms With Various Similarity Measures For Movie Recommendation. *International Journal of Computer Science, Engineering and Applications*, [online] 6(3), pp.1–20. Available at: <http://airccj.org/csecfp/library/jvol.php?last=IJCSEA&volname=6&volno=3>.

Banks, S., 2015. *Cyber Attack Is Real And Costly Threat To Small Business*. [online] Golsan Scruggs. Available at: <http://www.golsanscruggs.com/cyber-attacks-are-a-real-threat/>.

Barlowe, B., Blackbird, J. and Davis, W.S., 2012. *The Evolution Of Malware And*

*The Threat Landscape - A 10-Year Review*. [online] Available at: <http://www.isaca.org/Knowledge-Center/Blog/Lists/Posts/Post.aspx?ID=184>.

Bertino, E., Martino, L.D., Paci, F. and Squicciarini, A.C., 2010. Web Services Threats, Vulnerabilitieis, And Countermeasures. In: *Security for Web Services and Service-Oriented Architectures*, 1st ed. [online] Springer-Verlag Berlin Heidelberg, p.25–44 (226). Available at: <http://www.springer.com/gp/book/9783540877417>.

Bhuddtham, T. and Watanapongse, P., 2016. Time-Related Vulnerability Lookahead Extension To The CVE. In: *2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. [online] pp.1–6. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/7748927/>.

Bozorgi, M., Saul, L.K., Savage, S. and Voelker, G.M., 2010. Beyond Heuristics: Learning To Classify Vulnerabilities And Predict Exploits. In: *KDD '10 Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [online] pp.105–114. Available at: <https://dl.acm.org/citation.cfm?id=1835821>.

Breese, J.S., Heckerman, D. and Kadie, C., 1998. Empirical Analysis Of Predictive Algorithms For Collaborative Filtering. In: *UAI'98 Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. pp.43–52.

Buttner, A. and Ziring, N., 2008. *Common Platform Enumeration (CPE) – Specification 2.1*. Available at: <https://cpe.mitre.org/files/cpe-specification_2.1.pdf>.

Cacheda, F., Carneiro, V., Fernández, D. and Formoso, V., 2011. Comparison Of Collaborative Filtering Algorithms: Limitations Of Current Techniques And Proposals For Scalable, High-Performance Recommender Systems. *ACM Transactions on the Web (TWEB)*, [online] 5(1), pp.1–33. Available at: <https://dl.acm.org/citation.cfm?id=1921593>.

Christey, S. and Martin, R.A., 2007. *Vulnerability Type Distributions In CVE*. [online] *MITRE, Common Weakness Enumeration (CWE$^{TM}$)*. Available at: <https://cwe.mitre.org/documents/vuln-trends/index.html>.

Cremonesi, P., Koren, Y. and Turrin, R., 2010. Performance Of Recommender

Algorithms On Top-N Recommendation Tasks. In: *RecSys '10 Proceedings of the fourth ACM conference on Recommender systems*. pp.39–46.

CVE Community, 2017a. *Common Vulnerabilities And Exposures: About CVE*. [online] The MITRE Corporation. Available at: <https://cve.mitre.org/about/>.

CVE Community, 2017b. *Common Vulnerabilities and Exposures (CVE): About CVE Identifiers*. [online] The MITRE Corporation. Available at: <https://cve.mitre.org/cve/identifiers/>.

CVE Community, 2017c. *Common Vulnerabilities and Exposures (CVE): Frequently Asked Questions - What Is CVE?* [online] The MITRE Corporation. Available at: <https://cve.mitre.org/about/faqs.html>.

CVE Details, 2018. *CVE Details: The Ultimate Security Vulnerability Datasource*. [online] Available at: <https://www.cvedetails.com/>.

CWE Community, 2017. *Common Weakness Enumeration (CWE): Home / Index*. [online] The MITRE Corporation. Available at: <https://cwe.mitre.org/index.html>.

Deshpande, M. and Karypis, G., 2004. Item-Based Top- N Recommendation Algorithms. *ACM Transactions on Information Systems (TOIS)*, [online] 22(1), pp.143–177. Available at: <https://dl.acm.org/citation.cfm?id=245126>.

Dev, A. V. and Mohan, A., 2016. Recommendation System For Big Data Applications Based On Set Similarity Of User Preferences. In: *2016 International Conference on Next Generation Intelligent Systems (ICNGIS)*. [online] p.6. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/7854058/>.

Ekstrand, M.D., Riedl, J.T. and Konstan, J.A., 2011. Collaborative Filtering Recommender Systems. *Foundations and Trends® in Human–Computer Interaction*, [online] 4(2), pp.81–173. Available at: <http://files.grouplens.org/papers/FnT CF Recsys Survey.pdf>.

European Union Agency for Network and Information Security (ENISA), 2018. *ENISA Threat Landscape Report 2017*. [online] Available at: <https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2017>.

Foltz, P.W. and Dumais, S.T., 1992. Personalized Information Delivery: An

Analysis Of Information Filtering Methods. *Communications of the ACM - Special issue on information filtering*, 35(12), pp.51–60.

Frei, S., May, M., Fiedler, U. and Plattner, B., 2006. Large-Scale Vulnerability Analysis. In: *LSAD '06 Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*. [online] pp.131–138. Available at: <http://dl.acm.org/citation.cfm?id=1162671>.

Frühwirth, C. and Männistö, T., 2009. Improving CVSS-Based Vulnerability Prioritization And Response With Context Information. In: *3rd International Symposium on Empirical Software Engineering and Measurement, 2009. ESEM 2009.* [online] pp.535–544. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/5314230/>.

Gallon, L., 2011. Vulnerability Discrimination Using CVSS Framework. In: *2011 4th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2011 - Proceedings*. [online] pp.1–6. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/5720656/>.

Ghani, H., Luna, J., Khelil, A., Alkadri, N. and Suri, N., 2013. Predictive Vulnerability Scoring In The Context Of Insufficient Information Availability. In: *2013 International Conference on Risks and Security of Internet and Systems (CRiSIS)*. [online] pp.1–8. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/6766359/>.

Gogna, A. and Majumdar, A., 2015. A Comprehensive Recommender System Model: Improving Accuracy For Both Warm And Cold Start Users. *IEEE Access*, [online] 3, pp.2803–2813. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/7361739/>.

Good, N., Schafer, J. Ben, Konstan, J.A., Borchers, A., Sarwar, B., Herlocker, J. and Riedl, J., 1999. Combining Collaborative Filtering With Personal Agents For Better Recommendations. In: *AAAI '99/IAAI '99 Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*. [online] pp.439–446. Available at: <https://dl.acm.org/citation.cfm?id=315352>.

Guibing Guo, Jie Zhang, Z.S. and N.Y.-S., 2015. LibRec: A Java Library For

Recommender Systems. In: *Proceedings of the 23rd Conference on User Modelling, Adaptation and Personalization (UMAP)*. [online] pp.1–4. Available at: <https://www.librec.net/>.

Herlocker, J.L., Konstan, J.A., Terveen, L.G. and Riedl, J.T., 2004. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems (TOIS)*, [online] 22(1), pp.5–53. Available at: <https://dl.acm.org/citation.cfm?id=963772&dl=ACM&coll=DL&CFID=1024067268&CFTOKEN=46169432>.

Hoffman, T., 2004. Latent Semantic Models For Collaborative Filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1), pp.89–115.

Holm, H., Ekstedt, M. and Andersson, D., 2012. Empirical Analysis Of System-Level Vulnerability Metrics Through Actual Attacks. *IEEE Transactions on Dependable and Secure Computing*, [online] 9(6), pp.825–837. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/6259801/>.

Houmb, S.H. and Franqueira, V.N.L., 2009. Estimating ToE Risk Level Using CVSS. In: *International Conference on Availability, Reliability and Security, 2009. ARES '09.* [online] pp.718–725. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/5066553/>.

Hu, Y., Koren, Y. and Volinsky, C., 2008. Collaborative Filtering For Implicit Feedback. In: *ICDM '08 Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. [online] pp.263–272. Available at: <https://dl.acm.org/citation.cfm?id=1510528.1511352>.

Huang, S., Tang, H., Zhang, M. and Tian, J., 2010. Text Clustering On National Vulnerability Database. In: *2010 Second International Conference on Computer Engineering and Applications (ICCEA)*. [online] pp.295–299. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/5445659/>.

Information-Technology Promotion Agency Japan, 2008. *IPA / ISEC ： Vulnerabilities ： CPE ( Common Platform Enumeration ) Overview*. [online] IPA/ISEC：Vulnerabilities：CPE (Common Platform Enumeration) Overview. Available at: <https://www.ipa.go.jp/security/english/vuln/CPE_en.html>.

Information Systems Audit and Control Association (ISACA), 2016. *Cybersecurity Fundamentals Glossary (ISACA). ISACA: Information Systems Audit and Control Association*, Available at: <http://www.isaca.org/knowledge-center/documents/glossary/cybersecurity_fundamentals_glossary.pdf>.

International Organisaton for Standardisation (ISO) and International Electrotechnical Commission (IEC), 2004. ISO/IEC 13335-1: 2004 - Information Technology — Security Techniques — Management Of Information And Communications Technology Security. [online] p.28. Available at: <www.iso.org>.

International Organisaton for Standardisation (ISO) and International Electrotechnical Commission (IEC), 2005. ISO/IEC 27002: 2005 - Information Technology — Security Techniques — Code Of Practice For Information Security Controls. [online] 1, p.128. Available at: <www.iso.org>.

International Organisaton for Standardisation (ISO) and International Electrotechnical Commission (IEC), 2011. *ISO/IEC 27005: 2011 - Information Technology — Security Techniques —Information Security Risk Management*. [online] Available at: <http://www.iso.org/iso/catalogue_detail?csnumber=56742>.

International Organisaton for Standardisation (ISO) and International Electrotechnical Commission (IEC), 2013. *ISO/IEC 27002: 2013 - Information Technology — Security Techniques — Code Of Practice For Information Security Controls*. [online] Available at: <www.iso.org>.

International Telecommunication Union (ITU), 2008. *Overview Of Cybersecurity*. [online] *Series X: Data Networks, Open System Communications and Security - Telecommunication Security*, Available at: <https://www.itu.int/rec/T-REC-X.1205>.

International Telecommunication Union (ITU) and Wamala (CISSP), D.F., 2012. *ITU National Cybersecurity Strategy Guide*. [online] Available at: <http://www.itu.int/ITU-D/cyb/cybersecurity/docs/ITUNationalCybersecurityStrategyGuide.pdf>.

Isinkaye, F.O., Folajimi, Y.O. and Ojokoh, B.A., 2015. Recommendation Systems: Principles, Methods And Evaluation. *Egyptian Informatics Journal*, [online] 16(3), pp.261–273. Available at:

<http://www.sciencedirect.com/science/article/pii/S1110866515000341>.

Johnson, C., Badger, L., Waltermire, D., Snyder, J. and Skorupka, C., 2016. *NIST Special Publication 800-150: Guide To Cyber Threat Information Sharing*. [online] Available at: <http://csrc.nist.gov/publications/PubsSPs.html>.

Jouini, M., Rabai, L.B.A. and Aissa, A. Ben, 2014. Classification Of Security Threats In Information Systems. In: *5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014)*. [online] Elsevier Masson SAS, pp.489–496. Available at: <http://0-www.sciencedirect.com.acpmil13web.ancheim.ie/science/article/pii/S1877050914006528>.

Kabbur, S., Ning, X. and Karypis, G., 2013. FISM: Factored Item Similarity Models For Top-N Recommender Systems. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. [online] pp.659–667. Available at: <http://glaros.dtc.umn.edu/gkhome/node/1068>.

Kang, Z., Peng, C. and Cheng, Q., 2016. Top-N Recommender System via Matrix Completion. In: *AAAI'16 Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. [online] pp.179–184. Available at: <https://arxiv.org/abs/1601.04800>.

Karypis, G., 2001. Evaluation Of Item-Based Top-N Recommendation Algorithms. *CIKM '01 Proceedings of the Tenth Iternational Conference on Information and Knowledge Management*, [online] pp.1–13. Available at: <https://dl.acm.org/citation.cfm?id=502627>.

Kitts, B., Freed, D. and Vrieze, M., 2000. Cross-Sell: A Fast Promotion-Tunable Customer-Item Recommendation Method Based On Conditionally Independent Probabilities. In: *KDD '00 Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. pp.437–446.

Klimburg, A., 2012. *National Cyber Security Framework Manual*. [online] *National Cyber Security Framework Manual, NATO CCD COE*. NATO CCD COE Publications. Available at: <https://ccdcoe.org/publications/books/NationalCyberSecurityFrameworkManual.pdf>.

Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R. and Riedl, J., 1997. GroupLens: Applying Collaborative Filtering To Usenet News. *Communications of the ACM*, [online] 40(3), pp.77–87. Available at: <https://dl.acm.org/citation.cfm?id=245126>.

Koren, Y., 2008. Factorization Meets The Neighborhood: A Multifaceted Collaborative Filtering Model. In: *KDD '08 Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. [online] pp.426–434. Available at: <https://dl.acm.org/citation.cfm?id=1401944>.

Last, D., 2015. Using Historical Software Vulnerability Data To Forecast Future Vulnerabilities. In: *Resilience Week (RWS), 2015*. [online] pp.120–126. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/7287429/>.

Lee, J., Sun, M. and Lebanon, G., 2012. A Comparative Study Of Collaborative Filtering Algorithms. *arXiv:1205.3193 [cs.IR]*, [online] pp.1–27. Available at: <https://arxiv.org/abs/1205.3193>.

LibRec, 2018. *LibRec: Documentation*. [online] Available at: <https://www.librec.net/dokuwiki/doku.php#introduction>.

Liu, N.N. and Yang, Q., 2008. EigenRank: A Ranking-Oriented Approach to Collaborative Filtering. In: *SIGIR '08 Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. [online] pp.83–90. Available at: <http://www.mendeley.com/research/eigenrank-rankingoriented-approach-collaborative-filtering>.

Lü, L., Medo, M., Yeung, C.H., Zhang, Y.-C., Zhang, Z.-K. and Zhou, T., 2012. Recommender Systems. *Physics Reports 519 (2012)*, [online] 519(1), pp.1–49. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S037015731200082 8%5Cnhttp://www.sciencedirect.com/science/article/pii/S0370157312000828>.

Mell, P., Scarfone, K. and Romanosky, S., 2007. A Complete Guide To The Common Vulnerability Scoring System Version 2.0. *FIRST (Forum of Incident Response and Security Teams)*, [online] pp.1–24. Available at: <https://www.nist.gov/publications/complete-guide-common-vulnerability-scoring-system-version-20>.

Melville, P. and Sindhwani, V., 2010. Encyclopaedia of Machine Learning: Recommender Systems. In: C. Sammut and G.I. Webb, eds., *Encyclopaedia of Machine Learning*. [online] Springer US, pp.829–838. Available at: <https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-30164-8_705>.

de Moura Del Esposte, A., Campiolo, R., Kon, F. and Batista, D., 2016. A Collaboration Model To Recommend Network Security Alerts Based On The Mixed Hybrid Approach. In: *The Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*. [online] ResearchGate, p.15. Available at: <http://www.sbrc2016.ufba.br/>.

Murtaza, S.S., Khreich, W., Hamou-Lhadj, A. and Bener, A.B., 2016. Mining Trends And Patterns Of Software Vulnerabilities. *Journal of Systems and Software*, [online] 117, pp.218–228. Available at: <http://dx.doi.org/10.1016/j.jss.2016.02.048>.

National Institute of Standards and Technology, 2014. Framework For Improving Critical Infrastructure Cybersecurity. *National Institute of Standards and Technology: Cybersecurity Framework*, [online] 1(Februrary), p.39. Available at: <https://www.nist.gov/sites/default/files/documents/cyberframework/cybersecurity-framework-021214.pdf>.

National Institute of Standards and Technology (NIST), 2012. *NIST Special Publication 800-30 Revision 1: Guide For Conducting Risk Assessments*. [online] Available at: <http://csrc.nist.gov/publications/PubsSPs.html>.

National Institute of Standards and Technology (NIST), 2016. *National Vulnerability Database (NVD): Summary*. [online] NIST: National Vulnerability Database (NVD). Available at: <https://www.nist.gov/programs-projects/national-vulnerability-database-nvd>.

National Institute of Standards and Technology (NIST), 2018. *Official Common Platform Enumeration (CPE) Dictionary*. [online] Available at: <https://nvd.nist.gov/products/cpe>.

National Institute of Standards and Technology (NIST) and Kissel, R.L., 2013. *Glossary Of Key Information Security Terms (NIST)*. [online] Available at: <https://www.nist.gov/node/579721>.

Neuhaus, S. and Zimmermann, T., 2010. Security Trend Analysis With CVE Topic Models. In: *2010 IEEE 21st International Symposium on Software Reliability Engineering (ISSRE)*. [online] pp.111–120. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/5635130/>.

Ning, X. and Karypis, G., 2011. SLIM: Sparse Linear Methods For Top-N Recommender Systems SLIM : Sparse Linear Methods. In: *ICDM '11 Proceedings of the 2011 IEEE 11th International Conference on Data Mining*. [online] pp.497–506. Available at: <https://dl.acm.org/citation.cfm?id=2118303>.

Open Web Application Security Project (OWASP), 2018. *OWASP Top 10 - 2017: The Ten Most Critical Web Application Security Risks*. [online] Available at: <https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf>.

Pan, R., Zhou, Y., Cao, B., Liu, N.N., Lukose, R., Scholz, M. and Yang, Q., 2008. One-Class Collaborative Filtering. In: *ICDM '08 Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. pp.502–511.

Pennock, D.M., Horvitz, E., Lawrence, S. and Giles, C.L., 2000. Collaborative Filtering By Personality Diagnosis: A Hybrid Memory and Model-Based Approach. In: *UAI '00 Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*. [online] pp.473–480. Available at: <https://dl.acm.org/citation.cfm?id=720062>.

Portugal, I., Alencar, P. and Cowan, D., 2015. The Use Of Machine Learning Algorithms In Recommender Systems: A Systematic Review. *arXiv:1511.05263 [cs.SE]*, [online] pp.1–16. Available at: <https://arxiv.org/abs/1511.05263>.

Recorded Future, 2015. *Anticipating Cyber Vulnerability Exploits Using Machine Learning*. [online] Available at: <http://www.dataversity.net/new-report-anticipating-cyber-vulnerability-exploits-using-machine-learning/>.

Rendle, S., Freudenthaler, C., Gantner, Z. and Schmidt-Thieme, L., 2009. BPR: Bayesian Personalized Ranking From Implicit Feedback. In: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI2009)*. [online] pp.452–461. Available at: <https://arxiv.org/abs/1205.2618>.

Rennie, J.D.M. and Srebro, N., 2005. Fast Maximum Margin Matrix Factorization

For Collaborative Prediction. In: *ICML '05 Proceedings of the 22nd international conference on Machine learning*. pp.713–719.

Ricci, F., Rokach, L., Shapira, B. and Kantor, P.B., 2011. *Recommender Systems Handbook*. [online] *Recommender Systems Handbook*. Springer. Available at: <http://www.cs.ubbcluj.ro/~gabis/DocDiplome/SistemeDeRecomandare/Recommender_systems_handbook.pdf>.

RStudio, 2018. *Take Control Of Your R Code*. [online] Available at: <https://www.rstudio.com/products/rstudio/>.

Rufi, A.W., 2006. Chapter 1: Vulnerabilities, Threats, And Attacks. In: *Network Security 1 and 2 Companion Guide (Cisco Networking Academy)*. [online] Cisco Press and Companion Guide, pp.1–49. Available at: <http://www.ciscopress.com/store/network-security-1-and-2-companion-guide-cisco-networking-9781587131622>.

Sánchez, J.L., Serradilla, F., Martínez, E. and Bobadilla, J., 2008. Choice Of Metrics Used In Collaborative Filtering And Their Impact On Recommender Systems. In: *2008 2nd IEEE International Conference on Digital Ecosystems and Technologies, (IEEE-DEST 2008)*. [online] pp.432–436. Available at: <http://ieeexplore.ieee.org/document/4635147/>.

Sanguino, L.A.B. and Uetz, R., 2017. Software Vulnerability Analysis Using CPE And CVE. *Computer & Security (Cryptography and Security)*, [online] pp.1–29. Available at: <https://arxiv.org/abs/1705.05347>.

Sarwar, B., Karypis, G., Konstan, J. and Riedl, J., 2001. Item-Based Collaborative Filtering Recommendation Algorithms. In: *WWW '01 Proceedings of the 10th iIternational conference on World Wide Web*. [online] pp.285–295. Available at: <http://dl.acm.org/citation.cfm?id=372071>.

Sarwar, B.M., Karypis, G., Konstan, J.A. and Riedl, J.T., 2000. Application Of Dimensionality Reduction In Recommender System - A Case Study. *WebKDD-2000 Workshop*, [online] p.12. Available at: <http://glaros.dtc.umn.edu/gkhome/node/122>.

Scarfone, K. and Mell, P., 2009. An Analysis Of CVSS Version 2 Vulnerability

Scoring. In: *3rd International Symposium on Empirical Software Engineering and Measurement, 2009. ESEM 2009.* [online] pp.516–525. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/5314220/>.

Schafer, J. Ben, Frankowski, D., Herlocker, J. and Sen, S., 2007. Collaborative Filtering Recommender Systems. In: *The Adaptive Web*. [online] pp.291–324. Available at: <http://dl.acm.org/citation.cfm?id=1768208>.

Schryen, G. and Rich, E., 2010. Increasing Software Security through Open Source or Closed Source Development? Empirics Suggest that We have Asked the Wrong Question. In: *Proceedings of the Annual Hawaii International Conference on System Sciences*. [online] pp.1–10. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/5428450/>.

Shardanand, U. and Maes, P., 1995. Social Information Filtering: Algorithms For Automating "Word Of Mouth". In: *CHI '95 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. pp.210–217.

Sharma, R., Gopalani, D. and Meena, Y., 2017. Collaborative Filtering-Based Recommender System: Approaches And Research Challenges. In: *2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT)*. [online] pp.1–6. Available at: <http://ieeexplore.ieee.org/document/7977363/>.

Shirey, R., 2007. *Request For Comments: 4949 - Internet Security Glossary, Version 2*. *Request For Comments: 4949 - Internet Security Glossary, Version 2*, Available at: <https://tools.ietf.org/html/rfc4949>.

von Solms, R. and van Niekerk, J., 2013. From Information Security To Cyber Security. *Computers & Security (2013)*, [online] 38(October 2013), pp.97–102. Available at: <http://dx.doi.org/10.1016/j.cose.2013.04.004>.

Srebro, N., Rennie, J.D.M. and Jaakkola, T.S., 2004. Maximum-Margin Matrix Factorization. In: *NIPS'04 Proceedings of the 17th International Conference on Neural Information Processing Systems*. pp.1329–1336.

Steck, H., 2013. Evaluation Of Recommendations: Rating-Prediction And Ranking. In: *Proceedings of the 7th ACM conference on Recommender systems*. [online]

pp.213–220. Available at: <https://dl.acm.org/citation.cfm?id=2507160>.

Stine, K., Kissel, R., Barker, W.C., Fashlsing, J. and Gulick, J., 2008. *NIST SP 800-60 Volume I Revision 1: Appendices To Guide For Mapping Types Of Information And Information Systems To Security Categories*. [online] *NIST Special Publication 800-60 Volume I Revision 1*, Available at: <http://csrc.nist.gov/publications/PubsSPs.html>.

Swanson, M., Bowen, P., Phillips, A.W., Gallup, D. and Lynes, D., 2010. *NIST Special Publication 800-34 Revision 1: Contingency Planning Guide For Federal Information Systems*. [online] Available at: <http://csrc.nist.gov/publications/PubsSPs.html>.

Symantec, 2014. *Symantec - Internet Security Threat Report 2014*. [online] *Symantec 2013 Trends*, Available at: <http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf>.

Tang, Y. and Tong, Q., 2016. BordaRank: A Ranking Aggregation Based Approach To Collaborative Filtering. In: *Computer and Information Science (ICIS), 2016 IEEE/ACIS 15th International Conference*. [online] p.6. Available at: <http://ieeexplore.ieee.org/document/7550761/>.

The MITRE Corporation, 2013. *CPE Specifications*. [online] Available at: <https://cpe.mitre.org/specification/#naming>.

The R Foundation, 2018. *Introduction To R*. [online] Available at: <https://www.r-project.org/about.html>.

Thorat, P.B., Goudar, R.M. and Barve, S., 2015. Survey On Collaborative Filtering, Content-based Filtering And Hybrid Recommendation System. *International Journal of Computer Applications (0975 – 8887)*, [online] 110(4), pp.31–36. Available at: <https://pdfs.semanticscholar.org/51f4/78339cc86f20fff222baf17be5bab7f29bc3.pdf>.

Threat Analysis Group, 2010. *Threat, Vulnerability, Risk - Commonly Mixed Up Terms*. [online] Threat, Vulnerability, Risk - Commonly Mixed Up Terms. Available

at: <http://www.threatanalysis.com/2010/05/03/threat-vulnerability-risk-commonly-mixed-up-terms/>.

Toloudis, D., Spanos, G. and Angelis, L., 2016. Associating The Severity Of Vulnerabilities With Their Description. *Advanced Information Systems Engineering Workshops*, [online] 249, pp.231–242. Available at: <https://www.researchgate.net/publication/303822811_Associating_the_Severity_of_Vulnerabilities_with_their_Description>.

Wang, J.A., Zhou, L., Guo, M., Wang, H. and Camargo, J., 2010. Measuring Similarity For Security Vulnerabilities. In: *Proceedings of the 43rd Annual Hawaii International Conference on System Sciences 2010*. [online] p.10. Available at: <http://0-ieeexplore.ieee.org.acpmil13web.ancheim.ie/document/5428666/>.

Wang, S., Sun, J., Gao, B.J. and Ma, J., 2014. VSRank: A Novel Framework for Ranking-Based Collaborative Filtering. *ACM Transactions on Intelligent Systems and Technology (TIST) - Special Section on Urban Computing*, [online] 5(3), p.51:1--51:24. Available at: <https://dl.acm.org/citation.cfm?id=2542048>.

Whitman, M.E. and Mattord, H.J., 2011. *Principles Of Information Security*. 4th ed. [online] *Principles of Information Security*. Course Technology Press. Available at: <http://dl.acm.org/citation.cfm?id=1983557>.

Zhang, R., Liu, Q., Chun-Gui, Wei, J.-X. and Huiyi-Ma, 2014. Collaborative Filtering For Recommender Systems. In: *2014 Second International Conference on Advanced Cloud and Big Data*. [online] pp.301–308. Available at: <http://ieeexplore.ieee.org/document/7176109/>.

Zhang, S., Caragea, D. and Ou, X., 2011. An Empirical Study On Using The National Vulnerability Database To Predict Software Vulnerabilities. In: *Database and Expert Systems Applications - 22nd International Conference*. [online] pp.1–15. Available at: <https://www.researchgate.net/publication/221465277_An_Empirical_Study_on_Using_the_National_Vulnerability_Database_to_Predict_Software_Vulnerabilities>.

Zhang, S., Ou, X. and Caragea, D., 2015. Predicting Cyber Risks Through National Vulnerability Database. *Information Security Journal: A Global Perspective*, [online] 24(4), pp.1–13. Available at:

<https://www.researchgate.net/publication/285370086_Predicting_Cyber_Risks_through_National_Vulnerability_Database>.

# Appendix A Common Weakness Enumerations (CWEs) within the National Vulnerability Database (NVD)

| CWE-ID | Name | Description |
|---|---|---|
| 16 | Configuration | Weaknesses in this category are typically introduced during the configuration of the software. |
| 17 | Code | Weaknesses in this category are typically introduced during code development, including specification, design, and implementation. |
| 18 | Source Code | Weaknesses in this category are typically found within source code. |
| 19 | Data Handling | Weaknesses in this category are typically found in functionality that processes data. |
| 20 | Input Validation | The product does not validate or incorrectly validates input that can affect the control flow or data flow of a program. |
| 21 | Path Equivalence | Weaknesses in this category can be used to access files outside of a restricted directory (path traversal) or to perform operations on files that would otherwise be restricted (path equivalence). |
| 22 | Path Traversal | The software uses external input to construct a pathname that is intended to identify a file or directory that is located underneath a restricted parent directory, but the software does not properly neutralize special elements within the pathname that can cause the pathname to resolve to a location that is outside of the restricted directory. |
| 59 | Link Following | The software attempts to access a file based on the filename, but it does not properly prevent that filename from identifying a link or shortcut that resolves to an unintended resource. |
| 74 | Injection | The software constructs all or part of a command, data structure, or record using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify how it is parsed or interpreted when it is sent to a downstream component. |
| 77 | Command Injection | The software constructs all or part of a command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could |

| | | |
|---|---|---|
| | | modify the intended command when it is sent to a downstream component. |
| 78 | OS Command Injections | The software constructs all or part of an OS command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended OS command when it is sent to a downstream component. |
| 79 | Cross-Site Scripting (XSS) | The software does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users. |
| 89 | SQL Injection | The software constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component. |
| 91 | XML Injection (aka Blind XPath Injection) | The software does not properly neutralize special elements that are used in XML, allowing attackers to modify the syntax, content, or commands of the XML before it is processed by an end system. |
| 93 | Improper Neutralization of CRLF Sequences ('CRLF Injection') | The software uses CRLF (carriage return line feeds) as a special element, e.g. to separate lines or records, but it does not neutralize or incorrectly neutralizes CRLF sequences from inputs. |
| 94 | Code Injection | The software constructs all or part of a code segment using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the syntax or behaviour of the intended code segment. |
| 119 | Buffer Errors | The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer. |
| 125 | Out-of-bounds Read | The software reads data past the end, or before the beginning, of the intended buffer. |
| 134 | Format String Vulnerability | The software uses a function that accepts a format string as an argument, but the format string originates from an external source. |
| 189 | Numeric Errors | Weaknesses in this category are related to improper calculation or conversion of numbers. |
| 190 | Integer Overflow or Wraparound | The software performs a calculation that can produce an integer overflow or wraparound, when the logic assumes that the resulting value will always be larger than the original value. This can introduce other weaknesses when the |

| | | |
|---|---|---|
| | | calculation is used for resource management or execution control. |
| 191 | Integer Underflow (Wrap or Wraparound) | The product subtracts one value from another, such that the result is less than the minimum allowable integer value which produces a value that is not equal to the correct result. |
| 199 | Information Management Errors | Weaknesses in this category are related to improper handling of sensitive information. |
| 200 | Information Leak / Disclosure | An information exposure is the intentional or unintentional disclosure of information to an actor that is not explicitly authorized to have access to that information. |
| 254 | Security Features | Software security is not security software. Here we're concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management. |
| 255 | Credentials Management | Weaknesses in this category are related to the management of credentials. |
| 264 | Permissions, Privileges, and Access Control | Weaknesses in this category are related to the management of permissions, privileges, and other security features that are used to perform access control. |
| 284 | Improper Access Control | The software does not restrict or incorrectly restricts access to a resource from an unauthorized actor. |
| 287 | Authentication Issues | When an actor claims to have a given identity, the software does not prove or insufficiently proves that the claim is correct. |
| 295 | Improper Certificate Validation | The software does not validate, or incorrectly validates, a certificate. |
| 297 | Improper Validation of Certificate with Host Mismatch | The software communicates with a host that provides a certificate, but the software does not properly ensure that the certificate is actually associated with that host. |
| 310 | Cryptographic Issues | Weaknesses in this category are related to the use of cryptography. |
| 332 | Insufficient Entropy in PRNG | The lack of entropy available for, or used by, a Pseudo-Random Number Generator (PRNG) can be a stability and security threat. |
| 345 | Insufficient Verification of Data Authenticity | The software does not sufficiently verify the origin or authenticity of data, in a way that causes it to accept invalid data. |
| 352 | Cross-Site Request Forgery (CSRF) | The web application does not, or cannot, sufficiently verify whether a well-formed, valid, consistent request was intentionally provided by the user who submitted the request. |

| | | |
|---|---|---|
| 362 | Race Conditions | The program contains a code sequence that can run concurrently with other code, and the code sequence requires temporary, exclusive access to a shared resource, but a timing window exists in which the shared resource can be modified by another code sequence that is operating concurrently. |
| 369 | Divide by Zero | The product divides a value by zero. |
| 384 | Session Fixation | Authenticating a user, or otherwise establishing a new user session, without invalidating any existing session identifier gives an attacker the opportunity to steal authenticated sessions. |
| 388 | Error Handling | This category includes weaknesses that occur when an application does not properly handle errors that occur during processing. |
| 399 | Resource Management Errors | Weaknesses in this category are related to improper management of system resources. |
| 400 | Uncontrolled Resource Consumption ('Resource Exhaustion') | The software does not properly restrict the size or amount of resources that are requested or influenced by an actor which can be used to consume more resources than intended. |
| 415 | Double Free | The product calls free() twice on the same memory address, potentially leading to modification of unexpected memory locations. |
| 416 | Use After Free | Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code. |
| 426 | Untrusted Search Path | The application searches for critical resources using an externally-supplied search path that can point to resources that are not under the application's direct control. |
| 428 | Unquoted Search Path or Element | The product uses a search path that contains an unquoted element, in which the element contains whitespace or other separators. This can cause the product to access resources in a parent path. |
| 434 | Unrestricted Upload of File with Dangerous Type | The software allows the attacker to upload or transfer files of dangerous types that can be automatically processed within the product's environment. |
| 476 | NULL Pointer Dereference | A NULL pointer dereference occurs when the application dereferences a pointer that it expects to be valid, but is NULL, typically causing a crash or exit. |
| 502 | Deserialization of Untrusted Data | The application deserializes untrusted data without sufficiently verifying that the resulting data will be valid. |
| 611 | Improper Restriction of | The software processes an XML document that can contain XML entities with URIs that resolve |

| | | |
|---|---|---|
| | XML External Entity Reference ('XXE') | to documents outside of the intended sphere of control, causing the product to embed incorrect documents into its output. |
| 798 | Use of Hard-coded Credentials | The software contains hard-coded credentials, such as a password or cryptographic key which it uses for its own inbound authentication, outbound communication to external components, or encryption of internal data. |
| 824 | Access of Uninitialized Pointer | The program accesses or uses a pointer that has not been initialized. |

# Appendix B National Vulnerability Database (NVD) and CVE Details Datasets (Excerpts)

| CVE_ID | Pub_date | Mod_date | CVSS_Score | CVSS_Imp | CVSS_Expl | CVSS_AV | CVSS_AC | CVSS_Au | CVSS_Conf | CVSS_Integ | CVSS_Avail | Aff_Sw | Edition | Last_Version | Vendor | Description | Category |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CVE-2014-4123 | 15/10/2014 | 16/10/2014 | 6 | 6 | 8 N | M | N | P | P | P | P | internet_explorer | N\A | 9 | microsoft | Microsoft In BROWSER | |
| CVE-2014-4124 | 15/10/2014 | 16/10/2014 | 6 | 6 | 8 N | M | N | P | P | P | P | internet_explorer | N\A | 9 | microsoft | Microsoft In BROWSER | |
| CVE-2014-4126 | 15/10/2014 | 16/10/2014 | 9 | 10 | 8 N | M | N | C | C | C | C | internet_explorer | - | 11 | microsoft | Microsoft In BROWSER | |
| CVE-2014-4127 | 15/10/2014 | 16/10/2014 | 9 | 10 | 8 N | M | N | C | C | C | C | internet_explorer | N\A | 9 | microsoft | Microsoft In BROWSER | |
| CVE-2014-4128 | 15/10/2014 | 16/10/2014 | 9 | 10 | 8 N | M | N | C | C | C | C | internet_explorer | N\A | 9 | microsoft | Microsoft In BROWSER | |
| CVE-2014-4129 | 15/10/2014 | 16/10/2014 | 9 | 10 | 8 N | M | N | C | C | C | C | internet_explorer | N\A | 8 | microsoft | Microsoft In BROWSER | |
| CVE-2014-4130 | 15/10/2014 | 16/10/2014 | 9 | 10 | 8 N | M | N | C | C | C | C | internet_explorer | - | 11 | microsoft | Microsoft In BROWSER | |
| CVE-2014-4132 | 15/10/2014 | 16/10/2014 | 9 | 10 | 8 N | M | N | C | C | C | C | internet_explorer | - | 11 | microsoft | Microsoft In BROWSER | |
| CVE-2014-4133 | 15/10/2014 | 16/10/2014 | 9 | 10 | 8 N | M | N | C | C | C | C | internet_explorer | N\A | 7 | microsoft | Microsoft In BROWSER | |
| CVE-2014-4134 | 15/10/2014 | 16/10/2014 | 9 | 10 | 8 N | M | N | C | C | C | C | internet_explorer | N\A | 8 | microsoft | Microsoft In BROWSER | |

**Figure B.1** Excerpt of the National Vulnerability Database (NVD) Dataset

| CVE_ID | CWE_ID | # of Exploits | Vulnerability Type(s) | Publish Date | Update Date | Score | Gained Access Level | Access | Complexity | Authentication | Conf. | Integ. | Avail. | Desc. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CVE-2014-4123 | 264 | | +Priv | 15/10/2014 | 30/10/2015 | 6.8 | User | Remote | Medium | Not required | Partial | Partial | Partial | Microsoft Internet Explorer 7 th |
| CVE-2014-4124 | 264 | | +Priv | 15/10/2014 | 30/10/2015 | 6.8 | User | Remote | Medium | Not required | Partial | Partial | Partial | Microsoft Internet Explorer 7 th |
| CVE-2014-4126 | 20 | | DoS Exec Code Mem. Corr. | 15/10/2014 | 09/12/2016 | 9.3 | None | Remote | Medium | Not required | Complete | Complete | Complete | Microsoft Internet Explorer 10 |
| CVE-2014-4127 | 399 | | DoS Exec Code Mem. Corr. | 15/10/2014 | 30/10/2015 | 9.3 | None | Remote | Medium | Not required | Complete | Complete | Complete | Microsoft Internet Explorer 6 th |
| CVE-2014-4128 | 20 | | DoS Exec Code Mem. Corr. | 15/10/2014 | 09/12/2016 | 9.3 | None | Remote | Medium | Not required | Complete | Complete | Complete | Microsoft Internet Explorer 6 th |
| CVE-2014-4129 | 20 | | DoS Exec Code Mem. Corr. | 15/10/2014 | 30/10/2015 | 9.3 | None | Remote | Medium | Not required | Complete | Complete | Complete | Microsoft Internet Explorer 8 a |
| CVE-2014-4130 | 20 | | DoS Exec Code Mem. Corr. | 15/10/2014 | 30/10/2015 | 9.3 | None | Remote | Medium | Not required | Complete | Complete | Complete | Microsoft Internet Explorer 11 |
| CVE-2014-4132 | 20 | | DoS Exec Code Mem. Corr. | 15/10/2014 | 09/12/2016 | 9.3 | None | Remote | Medium | Not required | Complete | Complete | Complete | Microsoft Internet Explorer 11 |
| CVE-2014-4133 | 20 | | DoS Exec Code Mem. Corr. | 15/10/2014 | 30/10/2015 | 9.3 | None | Remote | Medium | Not required | Complete | Complete | Complete | Microsoft Internet Explorer 6 a |
| CVE-2014-4134 | 20 | | DoS Exec Code Mem. Corr. | 15/10/2014 | 30/10/2015 | 9.3 | None | Remote | Medium | Not required | Complete | Complete | Complete | Microsoft Internet Explorer 6 th |

**Figure B.2** Excerpt of the CVE Details Dataset

# Appendix C Text / Numeric Representations of the User-Item-Rating Datasets (Excerpts)

| *User* | | *Item* | | *Rating* | |
|---|---|---|---|---|---|
| Vendor_Affected_Software | Vendor_Affected_Software.1 | CWE_ID | CWE_ID.1 | CVSS_Score | CVSS_Score.1 |
| microsoft \| windows_7 | 621 | 189 | 189 | 9 | 9 |
| microsoft \| windows_7 | 621 | 399 | 399 | 7 | 7 |
| microsoft \| internet_explorer | 522 | 399 | 399 | 9 | 9 |
| microsoft \| internet_explorer | 522 | 119 | 119 | 9 | 9 |
| mozilla \| firefox | 666 | 399 | 399 | 10 | 10 |
| mozilla \| firefox | 666 | 200 | 200 | 6 | 6 |
| google \| chrome | 420 | 399 | 399 | 4 | 4 |
| google \| chrome | 420 | 189 | 189 | 7 | 7 |
| apple \| mac_os_x | 44 | 200 | 200 | 4 | 4 |
| apple \| mac_os_x | 44 | 189 | 189 | 6 | 6 |

**Figure C.1** User-Item-Rating Dataset 1 (Vendor | Affected Software - CWE ID - CVSS Score)

| *User* | | *Item* | | *Rating* | |
|---|---|---|---|---|---|
| Vendor_Affected_Software_Edition_Version | Vendor_Affected_Software_Edition_Version.1 | CWE_ID | CWE_ID.1 | CVSS_Score | CVSS_Score.1 |
| microsoft \| windows_7 \| -:x64 \| - | 2013 | 189 | 189 | 9 | 9 |
| microsoft \| windows_7 \| N\A \| N\A | 2015 | 399 | 399 | 7 | 7 |
| microsoft \| internet_explorer \| N\A \| 9 | 1736 | 399 | 399 | 9 | 9 |
| microsoft \| internet_explorer \| N\A \| 9 | 1736 | 119 | 119 | 9 | 9 |
| mozilla \| firefox \| N\A \| 3.0.4 prev=1 | 2257 | 399 | 399 | 10 | 10 |
| mozilla \| firefox \| N\A \| 3.0.4 prev=1 | 2257 | 200 | 200 | 6 | 6 |
| google \| chrome \| N\A \| 8.0.552.214 prev=1 | 1256 | 399 | 399 | 4 | 4 |
| google \| chrome \| N\A \| 8.0.552.223 prev=1 | 1257 | 189 | 189 | 7 | 7 |
| apple \| mac_os_x \| N\A \| 10.4.11 | 168 | 200 | 200 | 4 | 4 |
| apple \| mac_os_x \| N\A \| 10.4.11 | 168 | 189 | 189 | 6 | 6 |

**Figure C.2** User-Item-Rating Dataset 2 (Vendor | Affected Software | Edition | Version - CWE ID - CVSS Score)

| *User* | | *Item* | | *Rating* | |
|---|---|---|---|---|---|
| Affected_Software_Edition_Version | Affected_Software_Edition_Version.1 | CWE_ID | CWE_ID.1 | CVSS_Score | CVSS_Score.1 |
| windows_7 \| -:x64 \| - | 2483 | 189 | 189 | 9 | 9 |
| windows_7 \| N\A \| N\A | 2485 | 399 | 399 | 7 | 7 |
| internet_explorer \| N\A \| 9 | 822 | 399 | 399 | 9 | 9 |
| internet_explorer \| N\A \| 9 | 822 | 119 | 119 | 9 | 9 |
| firefox \| N\A \| 3.0.4 prev=1 | 661 | 399 | 399 | 10 | 10 |
| firefox \| N\A \| 3.0.4 prev=1 | 661 | 200 | 200 | 6 | 6 |
| chrome \| N\A \| 8.0.552.214 prev=1 | 391 | 399 | 399 | 4 | 4 |
| chrome \| N\A \| 8.0.552.223 prev=1 | 392 | 189 | 189 | 7 | 7 |
| mac_os_x \| N\A \| 10.4.11 | 1485 | 200 | 200 | 4 | 4 |
| mac_os_x \| N\A \| 10.4.11 | 1485 | 189 | 189 | 6 | 6 |

**Figure C.3** User-Item-Rating Dataset 3 (Affected Software | Edition | Version - CWE ID - CVSS Score)

# Appendix D R Data Pre-Processing Code

```
#NVD Dataset Import
nvdDataFrame <- read.csv(file = "nvd_2014.csv", header = TRUE, sep = ",")
colnames(nvdDataFrame) <-
    c("CVE_ID", "Publication_Date", "Modification_Date", "CVSS_Score", "CVSS_Impact",
        "CVSS_Exploitability", "CVSS_Access_Vector", "CVSS_Access_Complexity",
        "CVSS_Authentication", "CVSS_Confidentiality", "CVSS_Integrity", "CVSS_Availability",
        "Affected_Software", "Edition", "Last_Version", "Vendor", "Description", "Category")

#CVE Details (NVDPubDate1999-2014Altered.csv) CSV file import
cveDataFrame <- read.csv(file = "CVEPubDate1999-2014Altered.csv", header = TRUE, sep = ",")
colnames(cveDataFrame) <-
    c("Number", "CVE_ID", "CWE_ID", "Number_of_Exploits",
        "Vulnerability_Type(s)", "Publish_Date", "Update_Date",
        "Score", "Gained_Access_Level", "Access", "Complexity",
        "Authentication", "Conf", "Integ", "Avail", "Desc")

#NVD and CVE Details data frames merge
#Filtered by CVE_ID
cveDataFrame$Number <- NULL
nvdCVEDataFrameMerge <- merge(nvdDataFrame, cveDataFrame, by = "CVE_ID")

#Convert the nvdCVEDataFrameMerge Publication_Date Column to Date format
nvdCVEDataFrameMerge$Publication_Date <-
    as.Date(nvdCVEDataFrameMerge$Publication_Date, "%d/%m/%Y")

#NVD / CVE Details merged data frame - published date into 'Day', 'Month' and 'Year'
#Publication_Day
nvdCVEDataFrameMerge$Publication_Day <- nvdCVEDataFrameMerge$Publication_Date
nvdCVEDataFrameMerge$Publication_Day <-
    format(as.Date(nvdCVEDataFrameMerge$Publication_Day, format = "%Y-%m-%d"), "%d")
nvdCVEDataFrameMerge$Publication_Day <-
    as.factor(nvdCVEDataFrameMerge$Publication_Day)

#SIMILAR PROCESS FOR THE MONTH AND YEAR

#nvdCVEDataFrameMerge reorganise Publication Day, Month and Year
nvdCVEDataFrameMerge <-
    nvdCVEDataFrameMerge[c(1,2,33:35,3:32)]

#NVD / CVE Details Merge data frame - start date of 2005 onwards
nvdCVEDataFrame2005Onward <-
    nvdCVEDataFrameMerge[nvdCVEDataFrameMerge$Publication_Date >= '2005-01-01',]

#Linux
LinuxNVDCVEDF2005Onward <-
    nvdCVEDataFrame2005Onward[nvdCVEDataFrame2005Onward$Vendor == "linux",]

#SIMILAR PROCESS FOR MICROSOFT, MOZILLA, GOOGLE, APPLE, SUN, CISCO

#Vendor(s) Combined Data frame - Key
VendorsKeyNVDCVEDF2005Onward <-
    rbind(LinuxNVDCVEDF2005Onward, MicrosoftNVDCVEDF2005Onward, MozillaNVDCVEDF2005Onward,
                GoogleNVDCVEDF2005Onward, AppleNVDCVEDF2005Onward, SunNVDCVEDF2005Onward,
                CiscoNVDCVEDF2005Onward)
```

```
#Attributes: Vendor, Affected Software, CWE ID, CVSS Score
#User: Vendor (19), Affected Software (16)
#Item: CWE ID (22)
#Rating: CVSS Score (7)
View(VendorsKeyNVDCVEDF2005Onward)
venAffSoftCWEIDCVSSScoreNVDCVEDF2005Onward <-
    VendorsKeyNVDCVEDF2005Onward[c(19, 16, 22, 7)]

#Combining Vendor, Affected Software into one column
venAffSoftCWEIDCVSSScoreNVDCVEDF2005Onward$Vendor_Affected_Software <-
    paste(venAffSoftCWEIDCVSSScoreNVDCVEDF2005Onward$Vendor,"|",
                venAffSoftCWEIDCVSSScoreNVDCVEDF2005Onward$Affected_Software)
venAffSoftCWEIDCVSSScoreNVDCVEDF2005Onward$Vendor <- NULL
venAffSoftCWEIDCVSSScoreNVDCVEDF2005Onward$Affected_Software <- NULL
venAffSoftCWEIDCVSSScoreNVDCVEDF2005Onward <-
    venAffSoftCWEIDCVSSScoreNVDCVEDF2005Onward[c(3,1:2)]
venAffSoftCWEIDCVSSScoreNVDCVEDF2005Onward$Vendor_Affected_Software <-
    as.factor(venAffSoftCWEIDCVSSScoreNVDCVEDF2005Onward$Vendor_Affected_Software)

#Vendor, Affected Software column converted to integer format
venAffSoftIntNVDCVEDF2005Onward <-
    as.data.frame(sapply(venAffSoftCWEIDCVSSScoreNVDCVEDF2005Onward$Vendor_Affected_Software, as.integer))

#Vendor, Affected Software Integer column bound to venAffSoftCWEIDCVSSScoreNVDCVEDF2005Onward data frame
venAffSoftCWEIDCVSSScoreIntNVDCVEDF2005Onward <-
    cbind(venAffSoftCWEIDCVSSScoreNVDCVEDF2005Onward, venAffSoftIntNVDCVEDF2005Onward)
venAffSoftCWEIDCVSSScoreIntNVDCVEDF2005Onward$Vendor_Affected_Software <- NULL
venAffSoftCWEIDCVSSScoreIntNVDCVEDF2005Onward <-
    venAffSoftCWEIDCVSSScoreIntNVDCVEDF2005Onward[c(3,1:2)]
colnames(venAffSoftCWEIDCVSSScoreIntNVDCVEDF2005Onward) <-
    c("Vendor_Affected_Software", "CWE_ID", "CVSS_Score")

#Vendor, Affected Software, CWE ID, CVSS Score Text and Integer Data Frames Column Binding
venAffSoftCWEIDCVSSScoreTextIntNVDCVEDF2005Onward <-
    cbind(venAffSoftCWEIDCVSSScoreNVDCVEDF2005Onward,
                venAffSoftCWEIDCVSSScoreIntNVDCVEDF2005Onward)
venAffSoftCWEIDCVSSScoreTextIntNVDCVEDF2005Onward <-
    venAffSoftCWEIDCVSSScoreTextIntNVDCVEDF2005Onward[c(1,4,2,5,3,6)]

#Remove rows with no CWE ID
venAffSoftCWEIDNACVSSScoreTextIntNVDCVEDF2005Onward <-
    venAffSoftCWEIDCVSSScoreTextIntNVDCVEDF2005Onward[!is.na(venAffSoftCWEIDCVSSScoreTextIntNVDCVEDF2005Onward$CWE_ID),]

#Vendor, Affected Software, CWE ID, CVSS Score Integer Data Frame, CWEs with NA removed
venAffSoftCWEIDNACVSSScoreIntNVDCVEDF2005Onward <-
    venAffSoftCWEIDNACVSSScoreTextIntNVDCVEDF2005Onward[c(2, 4, 6)]
colnames(venAffSoftCWEIDNACVSSScoreIntNVDCVEDF2005Onward) <-
    c("Vendor_Affected_Software", "CWE_ID", "CVSS_Score")

#Write CSV File: venAffSoftCWEIDNACVSSScoreIntNVDCVEDF2005Onward
write.table(venAffSoftCWEIDNACVSSScoreIntNVDCVEDF2005Onward,
                    "venAffSoftCWEIDNACVSSScoreIntNVDCVEDF2005Onward.csv",
                    row.names = FALSE, col.names = FALSE, sep = ",")

#SIMILAR PROCESS FOR DATASET 2 AND DATASET 3
```

# Appendix E Recommender System Java Code

```
public class RecSysConfigMavenDriver
{
    /*
     * AbstractRecommender
     * Collaborative Filtering
     * UserKNNRecommender
     * ItemKNNRecommender
     */
    public static String CONFIGURATION_FILE = "conf/UserKNN-CF.properties";
    //public static String CONFIGURATION_FILE = "conf/ItemKNN-CF.properties";

    /*
     * AbstractRecommender
     * Collaborative Filtering - Ranking
     * SLIMRecommender
     */
    //public static String CONFIGURATION_FILE = "conf/SLIM-CF-Ranking.properties";

    /*
     * MatrixFactorizationRecommender
     * Collaborative Filtering - Ranking
     * FISMaucRecommender
     * FISMrmseRecommender
     */
    //public static String CONFIGURATION_FILE = "conf/FISMauc-CF-Ranking.properties";
    //public static String CONFIGURATION_FILE = "conf/FISMrmse-CF-Ranking.properties";

    public static void main(String[] args) throws Exception
    {
        System.out.println("Configuration File Path: " + CONFIGURATION_FILE.toString() + "\n");

        Configuration configuration = new Configuration();
        String configurationFilePath = CONFIGURATION_FILE;
        Properties properties = new Properties();
        properties.load(new FileInputStream(configurationFilePath));
        for(String name: properties.stringPropertyNames())
        {
            configuration.set(name, properties.getProperty(name));
        }

        if(configurationFilePath.equals("conf/UserKNN-CF.properties"))
        {
            System.out.println("User KNN Recommender\n");
        }
        else if(configurationFilePath.equals("conf/ItemKNN-CF.properties"))
        {
            System.out.println("Item KNN Recommender\n");
        }
        else if(configurationFilePath.equals("conf/SLIM-CF-Ranking.properties"))
        {
            System.out.println("SLIM Recommender\n");
        }
        else if(configurationFilePath.equals("conf/FISMauc-CF-Ranking.properties"))
        {
            System.out.println("FISMauc Recommender\n");
        }
        else if(configurationFilePath.equals("conf/FISMrmse-CF-Ranking.properties"))
        {
            System.out.println("FISMrmse Recommender\n");
        }

        RecommenderJob job = new RecommenderJob(configuration);
        job.runJob();
        System.out.println("\nFinished Recommendation Process\n");
    }
}
```

# Appendix F Recommender System Properties Code

```
# Data Directory
dfs.data.dir=data

# Result Directory
dfs.result.dir=result

# Log Directory
dfs.log.dir=log

# Data Input Path
# User: Vendor|Affected Software; Item: CWE ID, Rating: CVSS Score
data.input.path=nvd/csv/venAffSoftCWEIDNACVSSScoreIntNVDCVEDF2005Onward.csv
# User: Vendor|Affected Software|Edition|Version; Item: CWE ID, Rating: CVSS Score
# data.input.path=nvd/csv/venAffSoftEditVerCWEIDNACVSSScoreIntNVDCVEDF2005Onward.csv
# User: Affected Software|Edition|Version; Item: CWE ID, Rating: CVSS Score
# data.input.path=nvd/csv/affSoftEditVerCWEIDNACVSSScoreIntNVDCVEDF2005Onward.csv

# Data Model Convertor
data.model.format=text
data.column.format=UIR
# binarize threshold mainly used in ranking
# -1.0 - maxRate, binarize rate into -1.0 and 1.0
# binThold = -1.0, do nothing
# binThold = value, rating > value is changed to 1.0 other is 0.0, mainly used in ranking
# for PGM 0.0 maybe a better choose
data.convert.binarize.threshold=-1.0

# Data Model Splitter (kcv)
data.model.splitter=kcv
data.splitter.cv.number=5

# Evaluator (ranking)
rec.recommender.isranking=true
rec.eval.enable=true
rec.recommender.ranking.topn=10
rec.eval.classes=auc,ap,ndcg

# Set the random seed for reproducing the results (split data, init parameters and other methods using random)
# Default is set 11
# If do not set ,just use System.currentTimeMillis() as the seed and could not reproduce the results.
rec.random.seed=2018
```

```
# User KNN Recommender
rec.recommender.class=userknn
rec.recommender.similarities=user
rec.similarity.class=pcc
# rec.similarity.class=cos
rec.neighbors.knn.number=50
rec.similarity.shrinkage=-1
rec.filter.class=generic

# Item KNN Recommender
rec.recommender.class=itemknn
rec.recommender.similarities=item
rec.similarity.class=pcc
#rec.similarity.class=cos
rec.neighbors.knn.number=20
rec.similarity.shrinkage=-1

# SLIM Recommender
rec.recommender.class=slim
rec.similarity.class=cos
rec.recommender.similarities=item
rec.iterator.maximum=40
rec.similarity.shrinkage=-1
rec.recommender.earlystop=true
rec.slim.regularization.l1=2
rec.slim.regularization.l2=0.1

# FISMrmse Recommender
rec.recommender.class=fismrmse
rec.iteration.learnrate=0.001
rec.iterator.maximum=14
rec.user.regularization=0.0008
rec.item.regularization=0.0008
rec.bias.regularization=0.0008
rec.factor.number=64
rec.fismrmse.rho=1
rec.fismrmse.alpha=0.8

# FISMauc Recommender
rec.recommender.class=fismauc
rec.iteration.learnrate=0.0001
rec.iterator.maximum=5
rec.user.regularization=0.00001
rec.item.regularization=0.00001
rec.bias.regularization=0.00001
rec.factor.number=192
rec.fismauc.rho=0.5
rec.fismauc.alpha=0.9
rec.fismauc.gamma=0.1
```