



PROFESSIONAL MASTER'S THESIS

**Data Collection and Processing Pipeline  
for  
Cyber Vulnerability Intelligence**

**Igor Forain de Sá Freire**

**Brasília, November of 2022**

**UNIVERSITY OF BRASILIA**

Faculty of Technology  
DEPARTMENT OF ELECTRICAL ENGINEERING

UNIVERSITY OF BRASILIA  
Faculty of Technology

PROFESSIONAL MASTER'S THESIS  
**Data Collection and Processing Pipeline  
for  
Cyber Vulnerability Intelligence**

**Igor Forain de Sá Freire**

**Advisor: Prof. Rafael T. de Sousa Jr., Ph.D**

**Co-Advisor: Prof. Robson de Oliveira Albuquerque, Ph.D**

*Professional Master's Thesis submitted to the Department of Electrical  
Engineering as a partial requirement to obtain  
the degree of Master in Electrical Engineering*

**Brasília, November of 2022**

PUBLICATION: PPEE.MP.024

UNIVERSITY OF BRASILIA  
Faculty of Technology

PROFESSIONAL MASTER'S THESIS  
**Data Collection and Processing Pipeline  
for  
Cyber Vulnerability Intelligence**

**Igor Forain de Sá Freire**

*Professional Master's Thesis submitted to the Department of Electrical  
Engineering as a partial requirement to obtain  
the degree of Master in Electrical Engineering*

Examination Board

Prof. Rafael T. de Sousa Jr., Ph.D, FT/UnB  
*Advisor*

\_\_\_\_\_

Prof. Rafael Rabelo Nunes, Ph.D, FT/UnB  
*Internal Examiner*

\_\_\_\_\_

Prof. Mario A. R. Dantas, Ph.D, DCC/UFJF  
*External Examiner*

\_\_\_\_\_

Prof. Georges Daniel Amvame Nze, Ph.D, FT/UnB  
*Substitute Examiner*

\_\_\_\_\_

## CATALOG SHEET

FORAIN, IGOR

Data Collection and Processing Pipeline for Cyber Vulnerability Intelligence [Federal District] 2022.  
xvi, 82 p., 210 x 297 mm (ENE/FT/UnB, Master, Electrical Engineering, 2022).

Professional Master's Thesis - University of Brasilia, Faculty of Technology.

Department of Electrical Engineering

- |                       |                           |
|-----------------------|---------------------------|
| 1. Cyber Intelligence | 2. Vulnerability Database |
| 3. Exploit Database   | 4. Vulnerability Ranking  |
| I. ENE/FT/UnB         | II. Título (série)        |

## BIBLIOGRAPHIC REFERENCE

FORAIN, I. (2022). *Data Collection and Processing Pipeline for Cyber Vulnerability Intelligence*.

Professional Master's Thesis, Department of Electrical Engineering, University of Brasília, Brasília, DF, 82 p.

## ASSIGNMENT OF RIGHTS

AUTHOR: Igor Forain de Sá Freire

TITLE: Data Collection and Processing Pipeline for Cyber Vulnerability Intelligence.

GRADE: Master in Electrical Engineering      YEAR: 2022

Permission is granted to the University of Brasilia to reproduce copies of The Professional Master's Thesis and loan or sell such copies for academic and scientific purposes only. The authors reserve other publication rights and no part of The Professional Master's Thesis may be reproduced without the written permission of the authors.

---

Igor Forain de Sá Freire

Dept. of Electrical Engineering (ENE) - FT

University of Brasilia (UnB)

Darcy Ribeiro Campus

CEP 70919-970 - Brasília - DF - Brasil

## ACKNOWLEDGEMENTS

The author acknowledges the support of ABIN grant 08/2019.

Also, I would like to thank Prof. Rafael T. de Sousa Jr. for being my advisor and guiding me to achieve the goals of this dissertation. Beamer was an excellent tip.

I am incredibly grateful to Prof. Robson de Oliveira Albuquerque, a friend and co-supervisor. He had an essential role in this work and supported me in publishing two conference papers.

Lastly, I am grateful for the technical and computational support of the Decision-Making Technologies Laboratory - LATITUDE, of the University of Brasilia, which is supported by CNPq - Brazilian National Research Council (Grant 312180/2019-5 PQ-2, and 465741/2014-2 INCT on Cybersecurity), by CAPES - Brazilian Higher Education Personnel Improvement Coordination (Grant 88887.144009/2017-00 PROBRAL), by FAP-DF - Brazilian Federal District Research Support Foundation (Grant 0193.001366/2016 UIoT, and Grant 0193.001365/2016 SSDDC), by the Brazilian Ministry of the Economy (Grant 005/2016 DIPLA, and Grant 083/2016 ENAP), by the Institutional Security Office of the Presidency of Brazil (Grant ABIN 002/2017), by the Administrative Council for Economic Defense (Grant CADE 08700.000047/2019-14), by the General Attorney of the Union (Grant AGU 697.935/2019), by the National Auditing Department of the Brazilian Health System SUS (Grant DENASUS 23106.118410/2020-85), by the General Attorney's Office for the National Treasure (Grant PGFN 23106.148934/2019-67), and by the University of Brasilia (Grant UnB COPEI 7129).

---

## ABSTRACT

Cyber attacks are a ubiquitous reality nowadays, affecting organizations and countries worldwide. In 2021, information security incidents resulted in billions of dollars in losses. Most of those events evolved from known vulnerabilities in information technology assets. However, several heterogeneous databases and sources host information about those flaws, turning the risk assessment difficult. Despite massive vulnerability databases supported by the USA and China governments, they differ in operation and coverage, which hinders and turns uncertain risk assessment processes. This work creates a Data Collection and Processing Pipeline for Cyber Vulnerability Intelligence to compare the USA National Vulnerability Database (NVD), the China National Vulnerability Database (CNVD), the China National Vulnerability Database of Information Security (CNNVD), and the Exploit Database (EDB). The results reveal that the CNNVD has 1,661 vulnerabilities entries more than the NVD and at least 40 more entries regarding Chinese vendors. Besides, they show a temporal correlation of 0.917560 with 70% of text similarity between the NVD and CNNVD, indicating that despite the latter tracking the former, it is not an automatic translation of the NVD, requiring the work of cyber specialists. Moreover, the pipeline includes a Recommender Exploitation-Vulnerability System (REVS) with the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) using entropy-based weighting to rank vulnerability-exploit. REVS works as a network scanning and recommender system, leveraging a mix of national vulnerability and exploit databases to enhance penetration testing and System Security Assessment. Experiments evaluated in the GNS3 emulator show that this work approach identifies nine more vulnerabilities than the commercial tool Vulners and that the exploit features are more important criteria than the Common Vulnerability Scoring System (CVSS) parameters to rank vulnerabilities. To the best of the authors' knowledge, this work is the first to normalize and compare the NVD, CNVD, CNNVD, and EDB, showing that the Chinese national vulnerability databases are leveraging exploit data to infer reserved status CVEs.

**Keywords:** Cyber Intelligence, Vulnerability Database, Exploit Database, Vulnerability Ranking.

---

## RESUMO

Os ataques cibernéticos são uma realidade onipresente hoje em dia, afetando organizações e países em todo o mundo. Em 2021, os incidentes de segurança da informação resultaram em bilhões de dólares em perdas. A maioria desses eventos evoluiu a partir de vulnerabilidades conhecidas em ativos de tecnologia da informação. Entretanto, vários bancos de dados e fontes heterogêneos hospedam informações sobre essas falhas, tornando a avaliação de risco difícil. Apesar dos enormes bancos de dados de vulnerabilidade apoiados pelos governos dos EUA e da China, eles diferem em operação e cobertura, o que dificulta e torna os processos de avaliação de risco incertos. Este trabalho cria um Pipeline de Coleta e Processamento de Dados para Inteligência de Vulnerabilidade Cibernética a fim de comparar o National Vulnerability Databas dos EUA (NVD), o China National Vulnerability Database (CNVD), o China National Vulnerability Database of Information Security (CNNVD), e o Exploit Database (EDB). Os resultados revelam que o CNNVD tem mais 1.661 entradas de vulnerabilidades do que o NVD e pelo menos mais 40 entradas relativas a fornecedores chineses. Além disso, eles mostram uma correlação temporal de 0,917560 com 70% de similaridade de texto entre o NVD e o CNNVD, indicando que apesar de o último rastrear o primeiro, não é uma tradução automática dele, exigindo o trabalho de ciber especialistas. Além disso, o pipeline inclui um Recommender Exploitation-Vulnerability System (REVS) com a Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) usando ponderação baseada em entropia para classificar os pares de vulnerabilidade-exploits. O REVS funciona como um sistema de varredura e recomendação de rede, empregando um mix de bancos de dados de vulnerabilidades e e exploits para melhorar os testes de penetração e a avaliação de segurança do sistemas. Os experimentos realizados no emulador GNS3 mostram que esta abordagem de trabalho identifica mais nove vulnerabilidades do que a ferramenta comercial Vulners e que os critérios relativos aos exploits tem mais do que os parâmetros do Common Vulnerability Scoring System (CVSS) para classificar as vulnerabilidades. De acordo com o referencial bibliográfico utilizado, este trabalho é o primeiro a normalizar e comparar o NVD, CNVD, CNNVD e EDB, mostrando que os bancos de dados de vulnerabilidade nacionais chineses estão aproveitando os dados de exploração para inferir os CVEs de status reservado.

**Palavras-chave:** Inteligência Cibernética, Banco de Dados de Vulnerabilidade, Banco de Dados de Exploit, Priorização de Vulnerabilidade.

# CONTENTS

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	MOTIVATION	2
1.2	GOALS	3
1.2.1	SPECIFIC GOALS	3
1.3	CONTRIBUTION	3
1.4	DISSERTATION OVERVIEW	3
1.5	SUMMARY	4
<b>2</b>	<b>BACKGROUND</b>	<b>5</b>
2.1	VULNERABILITY ASSESSMENT	5
2.2	THE CVE FRAMEWORK	7
2.3	THE CPE FRAMEWORK	8
2.4	THE CWE FRAMEWORK	10
2.5	THE CVSS FRAMEWORK	11
2.6	THE CAPEC FRAMEWORK	14
2.7	VULNERABILITY DATABASES	15
2.7.1	NVD	15
2.7.2	CNVD	16
2.7.3	CNNVD	16
2.7.4	JNVD	17
2.8	EXPLOIT DATABASES	19
2.8.1	METASPLOIT	19
2.8.2	EXPLOIT-DB	19
2.9	VULNERS	20
2.10	SUMMARY	21
<b>3</b>	<b>RELATED WORK</b>	<b>22</b>
3.1	VULNERABILITY ASSESSMENT	22
3.2	VULNERABILITY AND EXPLOIT DATABASES	23
3.3	RECOMMENDER AND RANKING SYSTEMS	27
3.4	SUMMARY	29
<b>4</b>	<b>DATA COLLECTION AND PROCESSING PIPELINE</b>	<b>30</b>
4.1	VULNERABILITY DATA	31
4.1.1	DATA HARVESTING	31
4.1.1.1	NVD	31
4.1.1.2	CNVD	32
4.1.1.3	CNNVD	33
4.1.2	DATA STRUCTURING	35



4.1.3	DATA ANALYSIS .....	39
4.1.4	MULTILINGUAL NLP .....	39
4.2	EXPLOIT DATA.....	43
4.2.1	DATA HARVESTING.....	44
4.2.1.1	METASPLOIT.....	44
4.2.1.2	THE EXPLOIT DATABASE (EDB) .....	45
4.2.2	DATA STRUCTURING.....	46
4.2.3	DATA ANALYSIS AND VISUALIZATION .....	46
4.3	SCANNER .....	47
4.4	MATCHER .....	48
4.5	RECOMMENDER.....	49
4.6	SUMMARY .....	51
<b>5</b>	<b>EXPERIMENTAL RESULTS .....</b>	<b>52</b>
5.1	TEST ENVIRONMENT .....	52
5.2	VULNERABILITY DATABASE RESULTS .....	53
5.2.1	PERFORMANCE RESULTS .....	54
5.2.2	COMPARISON RESULTS .....	55
5.2.3	COMPARING THE NVD AND CNNVD .....	56
5.2.4	MULTILINGUAL NLP .....	58
5.3	EXPLOIT DATABASE RESULTS .....	62
5.3.1	EDB ANALYSIS .....	62
5.4	SCANNER RESULTS .....	69
5.5	MATCHER RESULTS .....	70
5.6	RECOMMENDER RESULTS .....	72
5.7	SUMMARY .....	73
<b>6</b>	<b>CONCLUSIONS AND FUTURE WORKS.....</b>	<b>74</b>
6.1	FUTURE WORKS .....	75
	<b>REFERENCES.....</b>	<b>76</b>

# LIST OF FIGURES

2.1	Nmap Basic Scan. ....	6
2.2	Entities relationship. ....	6
2.3	Parsing Nmap CPE list. ....	6
2.4	Nmap returning vulnerability data. ....	7
2.5	The CVE Program Structure (from [1]). ....	8
2.6	Example of direct and indirect vulnerability (from [2]). ....	9
2.7	Vulnerability tracking in Maven (from [2]). ....	10
2.8	CWE Example (from [3]). ....	11
2.9	CVSS v3.1 Metric Groups (from [4]). ....	12
2.10	Relation between CVE, CWE and CAPEC. ....	14
2.11	CAPEC Example: SQL Injection (from [5]). ....	14
2.12	Basic NVD Data for CVE-2021-29425 (from [6]). ....	15
2.13	CPE and CWE Data for CVE-2021-29425 (from [6]). ....	16
2.14	The CNVD Web Page (from [7]). ....	17
2.15	The CNNVD Web Page (from [8]). ....	18
2.16	Last ten vulnerability records in the JVNDB (from [9]). ....	18
2.17	Example of an exploit list available on the Exploit-DB (from [10]). ....	20
2.18	Vulners Dashboard (from [11]). ....	21
4.1	Data Collection and Processing Pipeline ....	30
4.2	Vulnerability Database Flow. ....	31
4.3	NVD Download Page. ....	32
4.4	CNVD download without logon. ....	32
4.5	CNNVD Download Site. ....	34
4.6	CNNVD Error Message. ....	34
4.7	CNNVD download without logon. ....	34
4.8	Vulnerability Entry in JSON File from the NVD. ....	35
4.9	The CPE element in JSON File from the NVD. ....	35
4.10	The CVSS element in JSON File from the NVD. ....	36
4.11	Vulnerability entry in XML File from the CNVD. ....	36
4.12	Vulnerability entry in XML File from the CNNVD. ....	37
4.13	Data Models. ....	38
4.14	ETL Pipeline. ....	38
4.15	Text Comparison Scheme. ....	40
4.16	Model Test Scheme. ....	41
4.17	Ranking Models Scheme. ....	42
4.18	GPU implementation. ....	43
4.19	The Exploit Database Summary Flow. ....	44
4.20	Example of Exploit Data from EDB. ....	45

4.21	Data Model for Exploit-DB.....	46
5.1	Emulated SOC with QEMU/KVN.....	52
5.2	CNVD Files Sample.....	54
5.3	Issues Summary.....	56
5.4	Vendors Comparison.....	57
5.5	Monthly Time Series.....	57
5.6	Word Count Histogram for the NVD.....	58
5.7	Word Count Histogram for the CNNVD.....	58
5.8	Simple Average Text Similarity.....	60
5.9	Weighted Average Text Similarity.....	60
5.10	Text Similarity between the NVD and CNNVD.....	61
5.11	EDB Inconsistency.....	62
5.12	Rendered Web Page from EDB Inconsistency.....	62
5.13	EDB Wrong Search.....	63
5.14	The CNVD Web Search.....	65
5.15	Search results for CVE-2021-3560 in the CNVD Web.....	65
5.16	The CNNVD Web Search.....	66
5.17	Heatmap: Authors vs. Programming Language.....	68

# LIST OF TABLES

2.1	CVSS Rating .....	13
3.1	Data sources cited in [12] .....	24
3.2	Data sources used in [13] .....	25
3.3	Comparison with Related Works I .....	26
3.4	Comparison with Related Works II .....	28
4.1	Vulnerability Database Attributes .....	39
4.2	Exploit Database Attributes .....	47
5.1	Environment hardware and VMs .....	53
5.2	Data and text processing hardware .....	53
5.3	Batch Processing .....	54
5.4	Data Summary .....	55
5.5	Models Processing Performance .....	59
5.6	Models Processing Performance with GPU .....	61
5.7	CVEs in the EDB without being in the NVD .....	63
5.8	CVEs mentioned in EDB and listed only in CNVD or CNNVD .....	64
5.9	CNNVD Vulnerabilities and Their Exploits in EDB .....	66
5.10	Exploit Classification .....	67
5.11	Top 20 Exploit Authors .....	67
5.12	Top 20 Exploit Authors since 2020 .....	69
5.13	Detected Services .....	69
5.14	Vulnerability Summary .....	70
5.15	Unidentified Vulnerabilities .....	70
5.16	List of Exploits .....	71
5.17	Criteria Weight Values .....	72
5.18	TOPSIS Vulnerability Rank .....	73

## LIST OF SOURCE CODES

4.1	HTTP GET Configuration.....	33
4.2	MSF Parser Code Snippet.....	44
4.3	Python Wrapper to Nmap.....	47
4.4	Python Class Snippet to Wrap Vulners Webservice.....	48
4.5	Nmap vulns script.....	49

# LIST OF ABBREVIATIONS

AI	Artificial intelligence
APCERT	Asia Pacific Computer Emergency Response Team
CAPEC	Common Attack Pattern Enumeration and Classification
CAVSS	Context-Aware Vulnerability Scoring System
CNA	CVE Numbering Authority
CNCERT/CC	National Computer Network Emergency Response Technical Team/Coordination Center of China
CNITSEC	China Information Technology Security Evaluation Center
CNVD	China National Vulnerability Database
CNNVD	China National Vulnerability Database of Information Security
CPE	Common Platform Enumeration
CVE	Common Vulnerability and Exposure
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
CWRAF	Common Weakness Risk Analysis Framework
DHS	Department of Homeland Security
EDB	Exploit Database
ETL	Extract Transform and Load
FIRST	Forum of Incident Response and Security Teams
GHDB	Google Hacking Database
GNS3	Graphical Network Simulator-3
IPA	Information-Technology Promotion Agency
JPCERT	Japan Cyber Emergency Response Team
JVN	Japan Vulnerability Note
JVNDB	Japan Vulnerability Notes iPedia
KVM	Kernel-Based Virtual Machine
MCDM	Multi-Criteria Decision-Making
MSF	Metasploit Framework
NASK	National Research Institute
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
NVD	National Vulnerability Database
OVM	Ontology for Vulnerability Management
OWASP	Open Web Application Security Project
PLOVER	Preliminary List Of Vulnerability Examples for Researchers
QEMU	Quick Emulator
REVS	Recommender Exploitation-Vulnerability System

TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
URI	Uniform Resource Identifier
VA	Vulnerability Assessment
WFN	Well-Formed Name
WIVSS	Weighted Impact Vulnerability Scoring System

# 1 INTRODUCTION

Nowadays, computer systems are ubiquitous in daily life. The COVID-19 epidemic forced a rush to provide online services [14]. However, it also exposed system vulnerabilities and increased the attack surface [15, 16]. Although there are several databases regarding system vulnerabilities, the most comprehensive are those sponsored by the USA and China [12].

Even with a plethora of information about system security, several heterogeneous databases and sources host those data, making the risk assessment difficult [17]. The National Vulnerability Database (NVD) from the National Institute of Standards and Technology (NIST) of the USA is the authoritative source of systems vulnerability information [18, 19]. The Department of Homeland Security (DHS) sponsors the NVD, which provides extensive data regarding system vulnerabilities using Common Vulnerability Exposure (CVE), Common Weakness Enumeration (CWE), Common Platform Enumeration (CPE), and Common Vulnerability Scoring System (CVSS) [20, 21]. The MITRE Corporation established those frameworks for data classification and enumeration regarding software vulnerabilities [22]. Afterward, it transferred these standards to NIST, which nowadays is responsible for the NVD [6].

Despite the best effort from NIST to gather vulnerabilities, there might be a delay between NVD and other open sources [23]. Moreover, several new software and hardware suppliers from China, like Xiaomi and Huawei, might carry new vulnerabilities not covered by the NVD [24]. For this reason, systems security improvement requires tracking bug reports or new vulnerability sources [25]. New official databases from China and commercial ones like the Exploit Database (EDB), the Metasploit Framework (MSF), and Vulners [11], when orchestrated, can help achieve better situational awareness of enterprise risks.

The China National Vulnerability Database (CNVD) and the China National Vulnerability Database of Information Security (CNNVD) are a new option for better vulnerability assessment and mitigation [26, 13]. The CNVD [7] and the CNNVD [8] are state-sponsored systems like the NVD. The National Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC) supports the first. In contrast, the second works as a project from the China Information Technology Security Evaluation Center (CNITSEC). They present several obstacles to their foreign users despite claiming public access. Both provide a web interface in Mandarin for vulnerability search but do not offer an API for automatic download [12]. Moreover, despite an average disclosure time lower than the NVD, there is evidence of vulnerability hiding [27].

The joining and orchestration of heterogeneous databases and sources like the NVD, CNVD, CNNVD, and EDB is still an open question [28, 29]. Most works about mining vulnerability databases covered only the NVD [18, 28, 30, 31, 32, 23, 33]. Besides, some approaches leveraged the NVD, CNVD, and CNNVD together to tackle the Internet of Things (IoT) platform [12, 13] or used only CNNVD to seek vulnerabilities [34].

The Vulnerability Assessment (VA) and Penetration Testing (PT) are essential steps to assess and rank vulnerabilities [35, 36, 37, 38]. However, choosing the most critical and threatening outcomes of those steps are a decision process under several constraints. There are several approaches to solving issues in the



decision process, ranging from using the Markov chain to Deep Learning algorithms [39]. However, several attributes from the Common Vulnerability Scoring System (CVSS) bounds the VA [40]. So, it becomes a decision problem that can leverage recommender system algorithms [41, 42, 43] — usually not suitable for the ranking process — or the Multi-Criteria Decision-Making (MCDM) [44]. The Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) is an algorithm of the MCDM family used for the cybersecurity of 5g networks [45], power control systems [46] and Intrusion Detection System (IDS) [47].

## 1.1 MOTIVATION

The NVD has grown into the world's de facto vulnerability database nowadays. NIST collects and records most known vulnerability entries and encourages the community to inform it about the new ones [48]. It is a natural consequence of the central role of American organizations in the hardware and software industry since the seventies. Even well-known and established commercial tools and data feeds that support security professionals and pentesters worldwide rely on the NVD data, one way or another [49]. Nevertheless, relying on a unique data source regarding system vulnerabilities is not advisable; it may become a single point of failure. Even for the NVD, it is hard to cover the whole possible sources of vulnerability information. Moreover, some vendors may not be interested in disclosing system flaws to the NVD before releasing a patch to their customers. Indeed, part of those vulnerabilities had been published in the Darkweb and on Social Media before their registering in the NVD [50].

There has been an increased market share of Chinese organizations in the software and hardware industries since 2010 [51]. Vendors like Huawei, ZTE, and Xiaomi became mainstream in sectors like mobile communications and 5G networks. Their products might have been a new source of system vulnerabilities unknown to the NVD and related commercial tools. In this scenario, the CNVD and CNNVD are turning into a relevant source of hardware and software vulnerabilities [52]. However, they are hard to access data feeds and may present the same issues as the NVD: missing and undisclosed data. Moreover, the CNVD and CNNVD use Mandarin, and their features have not been comprehensively compared to those from the NVD [12].

Also, these concurrent vulnerability databases complicate the vulnerability assessment and pentest processes. With the increase of hosts and IoT devices, even more vulnerable applications, services, and operating systems are running in networked systems [13]. Usually, a simple network scan in a medium enterprise environment returns hundreds of possible vulnerabilities, making it impossible to fix them at the same time [53]. The traditional approaches that only use the CVSS base score to rank the CVEs are not reliable [54, 55]. They do not consider time-changing factors like increased exploit capabilities, threat actors, and the level of importance of information assets in the organization. New approaches are necessary to prioritize the riskiest ones to patch. Several constraints and network conditions guide this prioritization process. Therefore, this master thesis addresses those issues with a new pipeline for vulnerability ranking with multiple national vulnerabilities and exploit databases to support multicriteria decision-making, leveraging them to provide cyber intelligence awareness.

## 1.2 GOALS

The main objective of this work is to create a data collection and processing pipeline for Cyber Vulnerability Intelligence to uncover the features of the USA and China National Vulnerability Databases and rank vulnerabilities according to their CVSS attributes and exploit maturity. The pipeline includes a Multilingual NLP function to compare the similarity of vulnerability descriptions in English and Chinese.

### 1.2.1 Specific Goals

- Extract Transform and Load (ETL) Chinese vulnerability databases;
- Join and orchestrate heterogeneous vulnerability and exploit databases;
- Develop a Recommender Exploitation-Vulnerability System (REVS).

## 1.3 CONTRIBUTION

This work reached the following contributions:

- Demonstration that the China vulnerability database is not a mere translation of the NVD, even for the registers that they have in common;
- Revelation that the China vulnerability databases are ahead of the USA and leveraging exploit data to infer vulnerabilities;
- Outperforming Vulners in vulnerability detection, enhancing the information security assessment and penetration testing;
- Improvement of vulnerability ranking through exploit criteria.

Moreover, it is worth mentioning that this work had part of its results published in the proceedings of two international conferences:

- The short paper “*REVS: A Vulnerability Ranking Tool for Enterprise Security*” in the 24<sup>th</sup> International Conference on Enterprise Information Systems (ICEIS 2022) [56].
- The full paper *Towards System Security: What a Comparison of National Vulnerability Databases Reveals* in the 17<sup>th</sup> Iberian Conference on Information Systems and Technologies (CISTI 2022) [57].

## 1.4 DISSERTATION OVERVIEW

The rest of this work is organized as follows:

- Chapter 2 provides the background about system vulnerability and exploit;
- Chapter 3 reviews the related works about information risk assessment, vulnerability databases, exploit data sources, and recommender/prioritization algorithms that motivated and supported this dissertation;
- Chapter 4 explains the Data Collection and Processing Pipeline for Cyber Vulnerability Intelligence, which includes a decision layer with TOPSIS to rank vulnerabilities;
- Chapter 5 exposes the test environment and discusses the experimental results achieved with the pipeline;
- Chapter 6 concludes this dissertation and suggests some future works.

## **1.5 SUMMARY**

This chapter introduced the dissertation, presenting the research context that motivated this work. It also listed the goals, contributions, and text structure. The next chapter presents the background contents of this work.

## 2 BACKGROUND

This chapter describes the basic concepts of vulnerabilities, exploits and their databases and frameworks. Section 2.1 presents the vulnerability assessment concepts and tools. Section 2.2 describes the CVE framework. Section 2.3 present the CPE naming system. Section 2.4 describes the CWE categorization scheme. Section 2.5 presents the CVSS Framework. Section 2.6 describes the CAPEC classification for cyberattack. Section 2.7 lists the main vulnerability databases. Section 2.8 lists two exploit databases. Section 2.9 presents Vulners, a new InfoSec data feed. Section 2.10 summarizes this chapter

### 2.1 VULNERABILITY ASSESSMENT

The Vulnerability Assessment, formally known by the acronym VA, is the workflow to identify, qualify, and quantify the flaws of a system, allowing prioritize handling it. The VA is a process that can be leveraged in any system, even if it is not based on computer hardware. Since several steps comprise the VA of network systems, this work focuses on the following ones: network scanning, CPE identification, CVE identification, exploit identification, and CVE prioritization.

There are some well-known tools to scan computer systems to search for available services. Nessus [58, 59], OpenVas [60], Qualys, Nmap [61] Nmap [62, 63] and Metasploit [64] are examples of those tools. Most are more than network scanners but frameworks and automation suites for network and systems security assessment. Moreover, as automation packages, several rely on Nmap as the main module for network scanning. Some of them are free software with established community support, as the example of Nmap that this work leverages as the core network scanner.

Nmap (Network Mapper) was first released in 1997 as a simple C language-based port scanner for Linux. Since then, it has been evolving with an OS reconnaissance module, version detection capacity, and a scripting engine [61]. It is free software based on the GNU GPLv2 license, forcing the software that leverages Nmap to use a compatible license type. The objective is to keep Nmap free and open, prohibiting its redistribution within proprietary products. If there is an intention to embed Nmap in a proprietary project, it provides a special Nmap OEM license.

Nmap became the de facto network scanner worldwide, providing several options from port mapping to version detection using TCP, ICMP, or UDP packets. Also, it is possible to calibrate the intensity of the reconnaissance step to avoid detection by IDS or IPS. One of the objectives of this work is vulnerability prioritization, first using services detection with Nmap without worries about bypassing defense mechanisms, which is out of the scope of this work. Figure 2.1 shows the basic service and its version detection by Nmap. This work with a Python interface will wrap this Nmap function.

Since the outcome of a network scanning is a list of system services listening to possible connections on computer ports, there is a need to establish a standard to name those services. It generated the CPE naming convention that this chapter describes in the subsection 2.3. After identifying the available CPEs, the next

```

igor@igor-Nitro:~/Projects$ nmap -sV 192.168.122.225 -oX ret_cpe.xml
Starting Nmap 7.80 ( https://nmap.org ) at 2022-04-17 19:55 -03
Nmap scan report for 192.168.122.225
Host is up (0.00044s latency).
Not shown: 991 filtered ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.7 ((Ubuntu))
445/tcp    open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp    open  ipp          CUPS 1.7
3000/tcp   closed ppp
3306/tcp   open  mysql        MySQL (unauthorized)
8080/tcp   open  http         Jetty 8.1.7.v20120910
8181/tcp   closed intermapper
Service Info: Host: UBUNTU; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.46 seconds

```

Figure 2.1: Nmap Basic Scan.

step is the search for possible vulnerabilities in them, formally known as CVEs. There is an intrinsic link between the CPE and CVE. The relation between them may be  $n$  to  $n$ . Figure 2.2 shows the relation among vendor, CPE, CVE, CWE, and exploit.

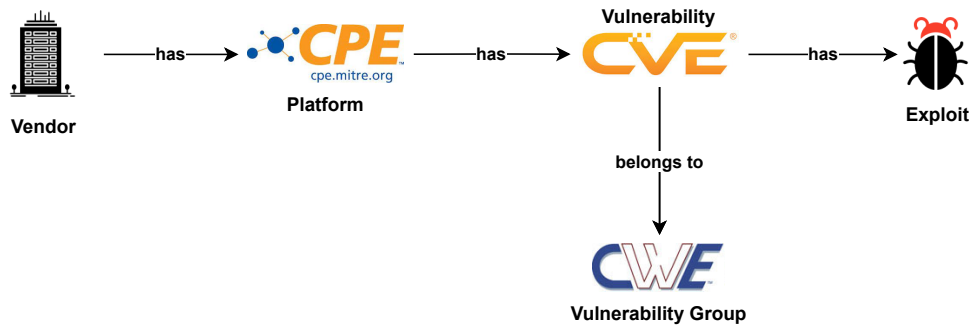


Figure 2.2: Entities relationship.

As can be seen in Figure 2.2, it is crucial to have the CPE name in order to look for possible CVEs. However, Figure 2.1 shows that `nmap -sV` command returns the list of services in the target machine to the standard output in a human-readable format. Fortunately, the Nmap enables the list of the same services in CPE format using the `-oX ret_cpe.xml` option. It saves several data about the target machine in XML format, including the corresponding CPEs of the services list. Figure 2.3 shows the CPEs list when parsing the `ret_cpe.xml` file.

```

igor@igor-Nitro:~/Projects$ xmllint --xpath '//cpe/text()' ret_cpe.xml
cpe:/a:proftpd:proftpd:1.3.5
cpe:/a:openbsd:openssh:6.6.1p1
cpe:/o:linux:linux_kernel
cpe:/a:apache:http_server:2.4.7
cpe:/a:samba:samba
cpe:/a:apple:cups:1.7
cpe:/a:mysql:mysql
cpe:/a:mortbay:jetty:8.1.7.v20120910

```

Figure 2.3: Parsing Nmap CPE list.

Now, with the service names in the CPE format, it is possible to look for entries in vulnerability databases. Nmap enables on-the-fly search for vulnerabilities regarding each detected CPE during network



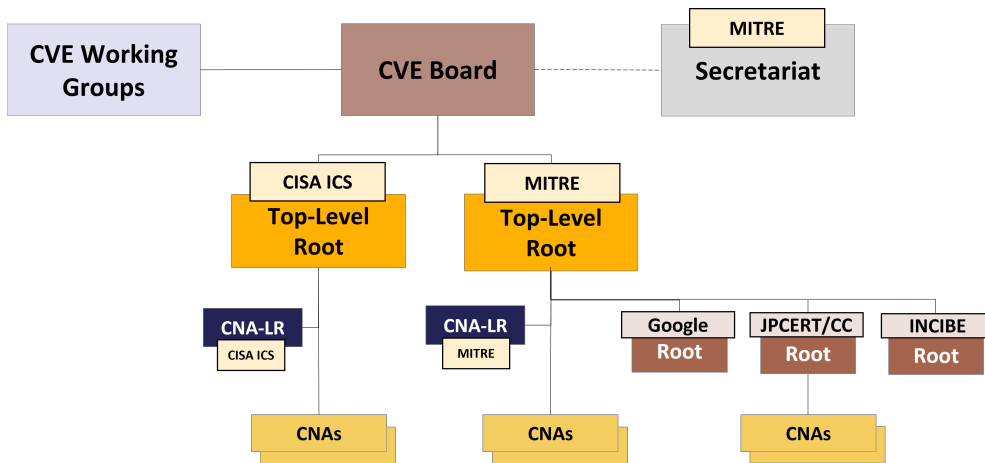


Figure 2.5: The CVE Program Structure (from [1])

for some time. They want to avoid reputation risks since the flaws might be exploited in case of sooner disclosure. Moreover, the vendor needs time to release patches for vulnerable platforms, highlighting that some huge companies like Microsoft and Google are CNA authorities.

## 2.3 THE CPE FRAMEWORK

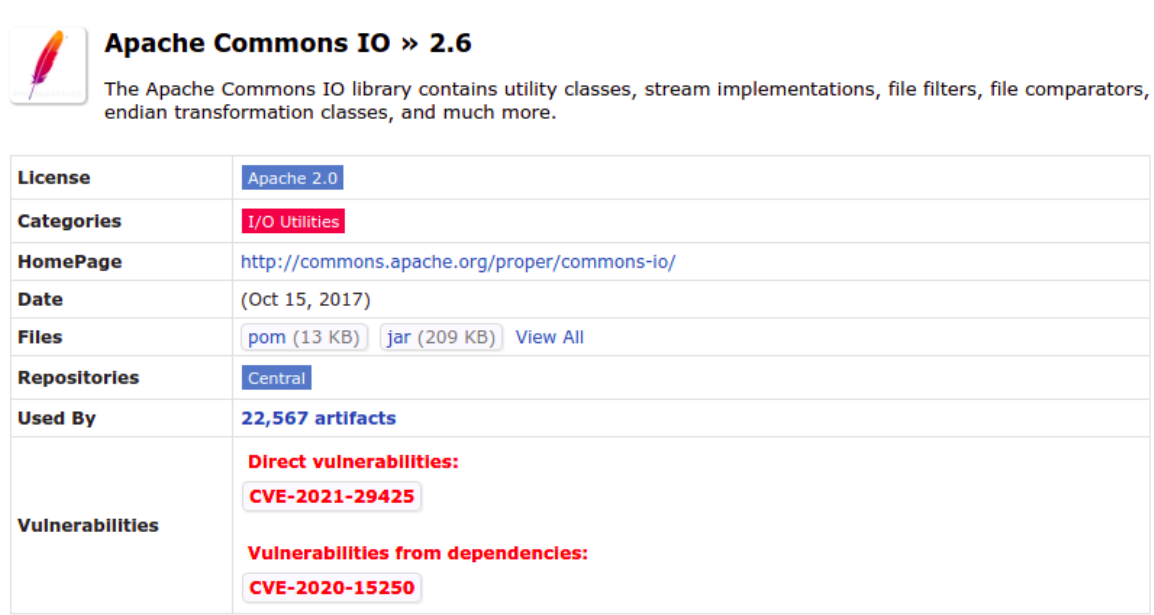
The Common Platform Enumeration (CPE) is a standardized naming scheme to identify hardware, applications, and operating systems groups. It is a scheme to identify classes of information technology assets, not a unique instance of those assets. For example, every Linux Ubuntu 20.04 has the same CPE identifier, despite being installed on different machines. A stack of specifications supports the CPE scheme, with the lower layers providing definitions for the upper ones: Applicability Language, Dictionary, Name Matching, and Naming. This last is the most important that supports the other ones. It specifies the Well-Formed Names (WFN) convention for CPE as well as procedures to bind those WFNs to machine-readable encoding using Uniform Resource Identifiers (URI) [67].

The CPE URI has the following pattern: *cpe:part:vendor:product:version:update:edition:sw\_edition:target\_sw:target\_hw:other*. The CPE 2.3 specification introduced the attributes *sw\_edition*, *target\_sw*, *target\_hw*, and *other*, which are called extended attributes to compatibility with prior specifications (2.1 and 2.2). The attributes of that URI are almost self-explanatory, but it is worth pointing out that the *part* attribute can accept the following values:

- /a: applications;
- /h: hardware;
- /o: operating systems.

It must be clear that the CPE is not the synonym for operating systems, hardware, application, or software library. It may be one of them or part of those systems. The software libraries are interesting cases

because some are not considered platforms but are supported by them. A library can depend on platforms, so its connection to the CVE is not direct because there is a CPE in the middle. Figure 2.6 illustrates this issue with the Apache Commons IO 2.6 <<https://mvnrepository.com/artifact/commons-io/commons-io/2.6>>.



**Apache Commons IO » 2.6**  
 The Apache Commons IO library contains utility classes, stream implementations, file filters, file comparators, endian transformation classes, and much more.

<b>License</b>	Apache 2.0
<b>Categories</b>	I/O Utilities
<b>HomePage</b>	<a href="http://commons.apache.org/proper/commons-io/">http://commons.apache.org/proper/commons-io/</a>
<b>Date</b>	(Oct 15, 2017)
<b>Files</b>	<a href="#">pom (13 KB)</a> <a href="#">jar (209 KB)</a> <a href="#">View All</a>
<b>Repositories</b>	Central
<b>Used By</b>	22,567 artifacts
<b>Vulnerabilities</b>	<p><b>Direct vulnerabilities:</b></p> <p><b>CVE-2021-29425</b></p> <p><b>Vulnerabilities from dependencies:</b></p> <p><b>CVE-2020-15250</b></p>


Figure 2.6: Example of direct and indirect vulnerability (from [2]).

Figure 2.6 shows an example in which a library has two vulnerabilities: a direct one, CVE-2021-29425, and an indirect one, CVE-2020-15250. The latter results from a vulnerability in JUnit4 (*cpe:/a:junit:junit4:*), a framework that Apache Commons IO 2.6 (*cpe:/a:apache:commons\_io:2.6:*) depends on. The Java ecosystem, for example, has thousands of libraries with several vulnerabilities that are not discovered directly by scanners. Either because they are not platforms or because they have only vulnerabilities from dependencies.

The mapping of vulnerable libraries without CPE is out of the scope of this work. However, it is worth mentioning that the same assessment method described here works for them. Regardless, network scanners like Nmap can detect the platforms the libraries depend on if they are running in the target system and have CPE id. For example, the Apache HTTP Server and the JBOSS Application Server.

There are some applications to list CVEs related to open source libraries. The Open Web Application Security Project (OWASP) Dependency-Check tools look for possible vulnerabilities regarding the chain of libraries. Those tools are essential during the development of secure software. They keep track of the vulnerability dependency tree using vulnerabilities repositories like Maven (<<https://mvnrepository.com/>>). Figure 2.7 shows an example of vulnerability tracking in Maven for several versions of the library Apache Commons IO.



 **Apache Commons IO**  
The Apache Commons IO library contains utility classes, stream implementations, file filters, file comparators, endian transformation classes, and much more.

License: [Apache 2.0](#)

Categories: [I/O Utilities](#)

Tags: [io](#)

Used By: 22,608 artifacts

---

Central (22) | [Atlassian 3rdParty \(1\)](#) | [Spring Plugins \(1\)](#) | [Redhat GA \(11\)](#) | [Redhat EA \(1\)](#) | [ICM \(3\)](#) | [Geomajas \(2\)](#) | [EBIPublic \(1\)](#) | [Mulesoft \(1\)](#)

	Version	Vulnerabilities	Repository
2.11.x	<a href="#">2.11.0.redhat-00001</a>		<a href="#">Redhat GA</a>
2.10.x	<a href="#">2.10.0.redhat-00001</a>		<a href="#">Redhat GA</a>
2.8.x	<a href="#">2.8.0.redhat-00001</a>		<a href="#">Redhat GA</a>
2.7.x	<a href="#">2.7.0.redhat-00003</a>	1 vulnerability	<a href="#">Redhat GA</a>
2.5.x	<a href="#">2.5.0.redhat-3</a>	1 vulnerability	<a href="#">Redhat GA</a>
	<a href="#">2.5.0.redhat-2</a>	1 vulnerability	<a href="#">Redhat GA</a>
2.4.x	<a href="#">2.4.0.redhat-1</a>	1 vulnerability	<a href="#">Redhat GA</a>
2.1.x	<a href="#">2.1.0.redhat-4</a>	1 vulnerability	<a href="#">Redhat GA</a>
	<a href="#">2.1.redhat-3</a>	1 vulnerability	<a href="#">Redhat GA</a>
	<a href="#">2.1.redhat-2</a>	1 vulnerability	<a href="#">Redhat GA</a>
	<a href="#">2.1.redhat-1</a>	1 vulnerability	<a href="#">Redhat GA</a>

Figure 2.7: Vulnerability tracking in Maven (from [2]).

## 2.4 THE CWE FRAMEWORK

Since the launching of the CVE framework by MITRE in 1999, a classification taxonomy for vulnerabilities became necessary. Until 2005, when the CVE was migrated to the NVD, MITRE evaluated by itself a categorization for software faults with implications for information security. In 2005, it released the Preliminary List Of Vulnerability Examples for Researchers (PLOVER). Since 2006, this fault classification has become a community-driven project under the Common Weakness Enumeration (CWE™) hood. With the surge of hardware faults affecting information security, the CWE added support for hardware weakness enumeration in 2020 [3]. The following list is examples of CWE:

- Software:
  - CWE-787 - Out-of-bounds Write;
  - CWE-79 - Improper Neutralization of Input During Web Page Generation (‘Cross-site Scripting’);
  - CWE-125 - Out-of-bounds Read.
- Hardware:
  - CWE-1189 - Improper Isolation of Shared Resources on System-on-a-Chip (SoC)
  - CWE-1191 - On-Chip Debug and Test Interface With Improper Access Control
  - CWE-1231 - Improper Prevention of Lock Bit Modification

Like the CVE, CWE has a scoring system for severity: Common Weakness Scoring System (CWSS™) and Common Weakness Risk Analysis Framework (CWRAF™). However, it has less direct security

implications than the CVSS for CVE. While the latter is specific for a vulnerability, the earlier is a broader score for a group of CVEs. Besides, the CWE has more educational than assessment purposes because it tries to avoid new vulnerabilities during software and hardware design. Moreover, the CWE is a hierarchical categorization of system flaws, as shown in Figure 2.8.

**CWE-276: Incorrect Default Permissions**

Weakness ID: 276  
Abstraction: Base  
Structure: Simple

Presentation Filter: Complete

**Description**  
During installation, installed file permissions are set to allow anyone to modify those files.

**Relationships**

**Relevant to the view "Research Concepts" (CWE-1000)**

Nature	Type	ID	Name
ChildOf		732	<a href="#">Incorrect Permission Assignment for Critical Resource</a>

**Relevant to the view "Software Development" (CWE-699)**

Nature	Type	ID	Name
MemberOf		275	<a href="#">Permission Issues</a>

**Relevant to the view "Hardware Design" (CWE-1194)**

Nature	Type	ID	Name
MemberOf		1198	<a href="#">Privilege Separation and Access Control Issues</a>

**Relevant to the view "Weaknesses for Simplified Mapping of Published Vulnerabilities" (CWE-1003)**

**Relevant to the view "Architectural Concepts" (CWE-1008)**

Figure 2.8: CWE Example (from [3]).

The CWE is a framework to group CVEs according to their type, e.g., web and buffer overflow. Lastly, the CPE is a method to logically describe the affected hardware and software, considering versions and conditions for the vulnerability [68]. For example, a web server software may be only vulnerable when running on a specific Operating System (OS) [69].

## 2.5 THE CVSS FRAMEWORK

The Forum of Incident Response and Security Teams (FIRST) was created in 1990 to improve the communication and coordination between different security incident response teams across the globe. As a non-profit organization from the USA, FIRST provides access to its services and products without costs.

The Common Vulnerability Scoring System (CVSS) is a framework for assessing the vulnerability attributes and calculating a score to its severity. It provides quantitative and qualitative outcomes to represent vulnerability features. Despite being one of the main components of any vulnerability database, the CVSS framework is provided by FIRST without costs as long as the user follows its guidelines. The CVSS Special Interest Group (SIG) from FIRST is responsible for keeping and improving the CVSS standard. Figure 2.9 presents the three sub-scores that comprise the CVSS [70].

The CVSS is the adopted scoring system by most vulnerability databases, including the NVD. Thus, it became the de facto standard to assess vulnerability risk worldwide. It has been evolving over the time: version 2.0 [71], version 3.0 [72] and version 3.1 [4]. Regardless of the version, the metric groups

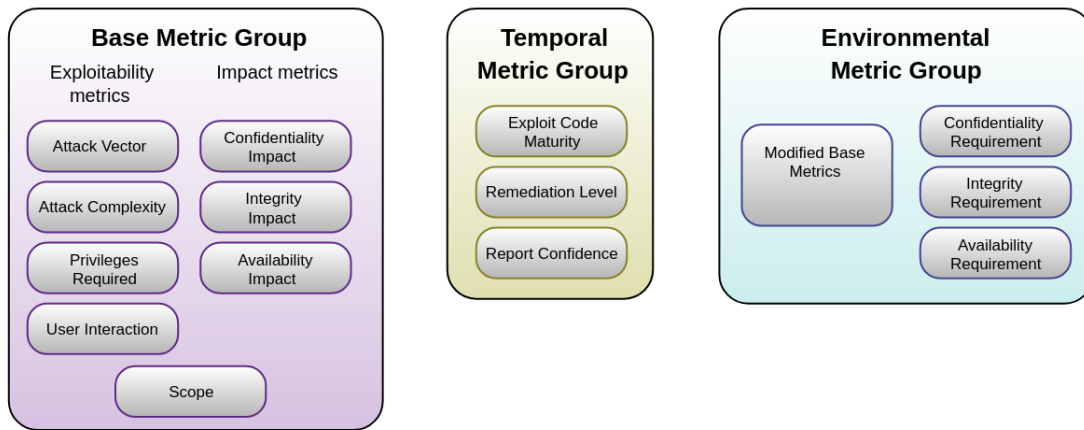


Figure 2.9: CVSS v3.1 Metric Groups (from [4]).

presented in Figure 2.9 are the building blocks of the framework. The following list – Base, Temporal and Environmental – describes those blocks:

- **Base** - the basic features of the vulnerability that almost do not change over time and are common to every instance;
- **Temporal** - the features that evolve and depend on external actors, e. g., the status of exploits and threat agents.;
- **Environmental** - the features that depend on the user requirements and system instances.

Most vulnerability databases and vendors use only the Base Score (BS). One premise of this score group is to be completely unaware of the external conditions: exploit status (Temporal Metric Group) and system requirements (Environmental Metric Group). The BS is a function of the metrics from the Base Metric Group in Figure 2.9. The features of this group are divided into three sub-groups:

- **Exploitability Sub-Score (ESS):** likelihood of being exploited.
  - *Attack Vector (AV)*: Network (N - 0.85), Adjacent (A - 0.62), Local (L - 0.55) or Physical (P - 0.2);
  - *Attack Complexity (AC)*: Low (L - 0.77) or High (H - 0.44);
  - *Privileges Required (PR)*: None (N - 0.85), Low (L - 0.62 or 0.68 if Scope is Changed (C)) or High (H - 0.27 (or 0.5 if Scope is Changed (C)));
  - *User Interaction (UI)*: None (N - 0.85) or Required (R - 0.62).
- **Impact Sub-Score (ISS):** impact of the exploitation
  - *Confidentiality (C)*: High (H - 0.56), Low (L - 0.22) or None (N - 0);
  - *Integrity (I)*: High (H - 0.56), Low (L - 0.22) or None (N - 0);
  - *Availability (A)*: High (H - 0.56), Low (L - 0.22) or None (N - 0);

- **Scope Sub-Score (SSS):** impacts resources beyond its own security authority.
  - *Scope (S):* Unchanged (U) or Changed (C)

The previous list has the variables to calculate the CVSS v3.x base score using the algorithm 1. The algorithm 1 describes the procedure to calculate the Base Score (BS) based on the CVSS v3.1 that the NVD uses. The BS from the CVSS v2.0 and v2.1 is slightly different. Since the CVSS v3.0 got in production only in 2015, every vulnerability published before that date has only the CVSS v2.x BS, while the entries after that date have the CVSS v2.x and CVSS v3.x BSs. The outcome from algorithm 1 is a number from 0 to 10, which can be converted to a qualitative view through table 2.1.

---

**Algorithm 1** CVSS v3.1 Base Score Calculation

---

```

ISS ← 1 - [(1 - C) × (1 - I) × (1 - A)]
ESS ← 8.22 × AV × AC × PR × UI
if S is 'U' then
  ISS ← 6.42 × ISS
else
  ISS ← 7.52 × (ISS - 0.029) - 3.25 × (ISS - 0.02)15
  N ← N - 1
end if
if ISS ≤ 0 then
  BS ← 0
else
  if S is 'U' then
    BS ← Roundup(Minimum[(ISS + ESS), 10])
  else
    BS ← Roundup(Minimum[1.08 × (ISS + ESS), 10])
  end if
end if

```

---

Table 2.1 reflects the evolution of the CVSS framework. The CVSS v3.x has greater granularity for rating than the CVSS x2.x. The CVSS v2.x also lacks several metrics to distinguish between different types of vulnerabilities regarding the new platforms, like IoT.

Table 2.1: CVSS Rating

<i>Rating</i>	<i>CVSS v3.x BS</i>	<i>CVSS v2.x BS</i>
None	0.0	***
Low	0.1 - 3.9	0.0 – 3.9
Medium	4.0 - 6.9	4.0 – 6.9
High	7.0 - 8.9	7.0 – 10.0
Critical	9.0 - 10.0	***

## 2.6 THE CAPEC FRAMEWORK

The Common Attack Pattern Enumeration and Classification (CAPEC™) is the counterpart of the CWE. The latter is a repository of weaknesses, while the earlier is a list of attack patterns to exploit those flaws. The CAPEC was also a project from MITRE that DHS incorporated in 2007. Figure 2.10 depicts the relation between the CVE, CWE, and CAPEC.



Figure 2.10: Relation between CVE, CWE and CAPEC.

Figure 2.10 shows that there is no direct relation between the CVE and CAPEC. However, there is an indirect relation between those two entities through the CWE. Like this last, the CAPEC also has hierarchical descriptions, but it has attack patterns descriptions instead of weaknesses. Figure 2.11 reproduces an example of CAPEC entry.

**CAPEC-66: SQL Injection**

Attack Pattern ID: 66 Status: Draft  
 Abstraction: Standard

Presentation Filter: Basic

**Description**

This attack exploits target software that constructs SQL statements based on user input. An attacker crafts input strings so that when the target software constructs SQL statements based on the input, the resulting SQL statement performs actions other than those the application intended. SQL Injection results from failure of the application to appropriately validate input.

**Extended Description**

When specially crafted user-controlled input consisting of SQL syntax is used without proper validation as part of SQL queries, it is possible to glean information from the database in ways not envisaged during application design. Depending upon the database and the design of the application, it may also be possible to leverage injection to have the database execute system-related commands of the attackers' choice. SQL Injection enables an attacker to interact directly to the database, thus bypassing the application completely. Successful injection can cause information disclosure as well as ability to add or modify data in the database.

**Relationships**

Nature	Type	ID	Name
ChildOf	✓	248	Command Injection
ParentOf	Ⓜ	7	Blind SQL Injection
ParentOf	Ⓜ	108	Command Line Execution through SQL Injection
ParentOf	Ⓜ	109	Object Relational Mapping Injection
ParentOf	Ⓜ	110	SQL Injection through SOAP Parameter Tampering
ParentOf	Ⓜ	470	Expanding Control over the Operating System from the Database

**View Name** **Top Level Categories**

Domains of Attack	Software
Mechanisms of Attack	Inject Unexpected Items

**Related Weaknesses**

CWE-ID	Weakness Name
89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
1286	Improper Validation of Syntactic Correctness of Input

Figure 2.11: CAPEC Example: SQL Injection (from [5]).

Figure 2.11 also shows that there is no direct relation between exploit and CAPEC. It is advisable not to misunderstand those entities. The latter works as categorizations and sequences of attack actions, while the earlier is an artifact or code snippet to target a specific software and hardware flaw. This vulnerability might have been listed as a CVE entry, or maybe it is unknown to the information security community. This last situation is critical because the attacker might keep exploiting that unknown vulnerability for a long time.

This work does not use the CAPEC for vulnerability ranking and prioritization because the features leveraged in the proposal depend mainly on the CVSS attributes and exploit status.

## 2.7 VULNERABILITY DATABASES

Vulnerability databases are repositories of software and hardware vulnerability entries organized so that information can be retrieved. Most of them use the Mitre frameworks, detailed in the last subsections, to create the data schema. The CVE id is usually the default primary key. Regarding the ownership, they can be private or public. However, as national security interests or affairs, the most extensive vulnerability databases worldwide are totally or partially supported by government institutions. This section addresses the following vulnerability databases: NVD, CNVD, CNNVD, and JNVD.

### 2.7.1 NVD

As explained in subsection 2.2, despite being independent programs, the CVE and NVD are backed up by the same USA State Authority, the DHS. Besides, there is a full synchronization between those two lists: a new entry in the Mitre CVE is instantly inserted or updated in the NVD. It provides data through three feeds: Web Page, Web Service based API, and JSON files feed [6]. Figure 2.12 shows an example of Web Page data provided by the NVD for the CVE-2021-29425. This CVE is the same mentioned in section 2.3.

**CVE-2021-29425 Detail**

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

**Current Description**

In Apache Commons IO before 2.7, When invoking the method `FileNameUtils.normalize` with an improper input string, like `"../foo"`, or `"\\.\foo"`, the result would be the same value, thus possibly providing access to files in the parent directory, but not further above (thus "limited" path traversal), if the calling code would use the result to construct a path value.

[Hide Analysis Description](#)

**Analysis Description**

In Apache Commons IO before 2.7, When invoking the method `FileNameUtils.normalize` with an improper input string, like `"../foo"`, or `"\\.\foo"`, the result would be the same value, thus possibly providing access to files in the parent directory, but not further above (thus "limited" path traversal), if the calling code would use the result to construct a path value.

**Severity** CVSS Version 3.x CVSS Version 2.0

CVSS 3.x Severity and Metrics:

**NIST:** NVD **Base Score:** 4.8 MEDIUM **Vector:** CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N

*NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.*

*Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.*

Figure 2.12: Basic NVD Data for CVE-2021-29425 (from [6]).

Figure 2.12 shows that the Web Page provides a more user-friendly interface than the API and JSON file feeds, but with almost the same data. The API returns data regarding one CVE ID at a time over JSON HTTP. On the other hand, the JSON files feed returns data regarding CVE IDs aggregated in one JSON file for each year. This last option is more suitable for batch processing than the other two. Figure 2.13 shows

that the Web Page also provides the CWE and CPE information.

### Weakness Enumeration

CWE-ID	CWE Name	Source
CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	NIST
CWE-20	Improper Input Validation	Apache Software Foundation

### Known Affected Software Configurations [Switch to CPE 2.2](#)

Configuration 1 ([hide](#))

✖ cpe:2.3:a:apache:commons_io:2.2:-:*:*:*:*
<a href="#">Show Matching CPE(s)</a>
✖ cpe:2.3:a:apache:commons_io:2.3:-:*:*:*:*
<a href="#">Show Matching CPE(s)</a>
✖ cpe:2.3:a:apache:commons_io:2.4:-:*:*:*:*
<a href="#">Show Matching CPE(s)</a>
✖ cpe:2.3:a:apache:commons_io:2.5:-:*:*:*:*
<a href="#">Show Matching CPE(s)</a>
✖ cpe:2.3:a:apache:commons_io:2.6:-:*:*:*:*
<a href="#">Show Matching CPE(s)</a>

Figure 2.13: CPE and CWE Data for CVE-2021-29425 (from [6]).

According to the NVD data feeds URL, this database offers the following vulnerability attributes:

- **Descriptions:** a brief analysis of the vulnerability record;
- **Severity:** the CVSS v2 and v3 – v3 only for entries after 2015 – as described in section 2.5;
- **References:** external URLs with extra information;
- **Weakness Enumeration:** the CWE ID of the weakness group to which the vulnerability record belongs;
- **Known Affected Software Configurations:** the vulnerable CPEs list.

## 2.7.2 CNVD

The National Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC) is responsible for hosting and supporting the China National Vulnerability Database (CNVD). That Center claims to be a non-profit organization with no subordination to the China government. As the main CERT of China, it is responsible for coordinating other cybersecurity regional centers in that country. It is also a member of the FIRST and one of the leaders of the Asia Pacific Computer Emergency Response Team (APCERT) [73].

Figure 2.14 is a print screen of the CNVD Web Page. It shows that one of the main drawbacks of the CNVD is the language support. There is no English language support, which requires this work to inspect that Web Site to get the vulnerability data.

## 2.7.3 CNNVD

The China Information Technology Security Evaluation Center (CNITSEC) is an organization tied to the Chinese Central Government and the Communist Party of China (CPC). It is the central organization in



Figure 2.14: The CNVD Web Page (from [7]).

China for vulnerability analysis and risk assessment regarding information security. It is also responsible for performing and funding cybersecurity research. The CNITSEC has also been responsible for the China National Vulnerability Database of Information Security (CNNVD) since 2009 [74]. It uses the following sources for data collection [8]:

- independent data mining;
- submission by users;
- collaborative sharing;
- network collection;
- technical testing.

The previous list shows that the CNNVD leverages an ecosystem of data sources to gather vulnerability records. Its Chinese presentation also lists the following partners to collect data: mainstream application software, operating systems, network equipment at home and abroad, government departments, industry, vendors, universities, and research institutions. After discovering system loopholes, the CNNVD has a dedicated team to perform analyses and vulnerability assessment [8].

Figure 2.15 is a print screen of the CNNVD Web Page. It also presents the identical drawback of the CNVD: there is only Chinese language support. Thus, this work had to reverse engineering part of the Web Page source code to grasp the CNVD services and functions.

## 2.7.4 JNVD

The Japan Vulnerability Notes iPedia (JVND) is a database of vulnerabilities regarding OSs, software, embedded systems, applications, and libraries used in Japan. The Japan Vulnerability Notes (JVN) is a web portal that hosts the JVND and provides analysis, solutions, and comments about vulnerabilities entries stored in that database. The JVN is jointly supported by the Japan Cyber Emergency Response





Figure 2.15: The CNNVD Web Page (from [8]).

Team (JPCERT) Coordination Center (CC) and the Information-technology Promotion Agency (IPA). They released the JVNDB in 2007. On that date, it had 325 vulnerabilities entries from 1998 to February 2007 [9]. Figure 2.16 shows the last ten vulnerability records in the JVNDB.

Last Updated	ID	Title	CVSSv3
2022/04/20 Update	<a href="#">JVND-2021-000097</a>	Multiple vulnerabilities in CLUSTERPRO X and EXPRESSCLUSTER X	9.8(Critical)
2022/04/15	<a href="#">JVND-2022-000026</a>	WordPress Plugin "MicroPayments - Paid Author Subscriptions, Content, Downloads, Membership" vulnerable to cross-site request forgery	4.3(Medium)
2022/04/07	<a href="#">JVND-2022-001526</a>	Trend Micro Antivirus for Mac vulnerable to privilege escalation	-
2022/03/31	<a href="#">JVND-2017-000174</a>	Self-Extracting Encrypted Files created by AttachCase may insecurely load Dynamic Link Libraries	7.8(High)
2022/03/31	<a href="#">JVND-2022-001494</a>	Trend Micro Apex Central and Trend Micro Apex Central as a Service vulnerable to improper check for file contents	8.6(High)
2022/03/30	<a href="#">JVND-2021-004912</a>	Multiple vulnerabilities in multiple ELECOM routers	8.8(High)
2022/03/30	<a href="#">JVND-2022-000024</a>	Zero-channel BBS Plus vulnerable to cross-site scripting	6.1(Medium)
2022/03/30	<a href="#">JVND-2022-000023</a>	WordPress Plugin "Advanced Custom Fields" vulnerable to missing authorization	6.5(Medium)
2022/03/30	<a href="#">JVND-2022-000022</a>	AttachCase may insecurely load Dynamic Link Libraries	7.8(High)
2022/03/29	<a href="#">JVND-2021-000108</a>	Multiple vulnerabilities in multiple ELECOM LAN routers	8.0(High)

Figure 2.16: Last ten vulnerability records in the JVNDB (from [9]).

This subsection cites the JVNDB just for awareness purposes because this work does not leverage that database. The JVN claims to use the following data sources to populate the database: the information published in the JVN portal, Japanese vendors, and the NVD. Moreover, figure 2.5 showed that the JPCERT/CC, which supports the JVN, is a high-level member of the CVE Program in the USA. Therefore, the JVNDB data should be promptly sent to the NVD, providing some synchronization between the JVNDB and NVD. By April, 20 2022, the JVNDB had 2416 entries [9].

This work does not mention the Bugtraq because it was discontinued by Accenture, the current owner of SecurityFocus, in 2021 [75]. The Bugtraq was a famous mailing list about information security supported by SecurityFocus since 1999.

## **2.8 EXPLOIT DATABASES**

Exploit databases are repositories of code snippets and Proof of Concepts (PoC) to exploit software and hardware vulnerabilities. Despite intrinsic linked to the vulnerability databases, while this last has the system flaws, that one provides offensive weapons. This section addresses the following exploit databases: Metasploit and Exploit-DB.

### **2.8.1 Metasploit**

The Metasploit was created in 2003 as a portable network tool base on the Perl language. After that, it was rebuilt in Ruby and acquired by the Rapid7 in 2009 [76]. This company boosted the Metasploit development that turned into a framework capable of performing all phases of penetration testing. It has two versions: Metasploit Pro and Metasploit Framework. The first is commercial one that offers several automation tools and user friendly graphical interface. Metasploit Framework is the original open source command-line tool on which the Pro version relies [77].

The architecture of Metasploit groups payloads, attack libraries and exploit in modules based on the Ruby language. Recently, Metasploit started to support modules in Python and Go languages also [77]. Usually, the main objective of those modules and exploits is getting actively or passively access to a command prompt in the target machine. No API or relational database is ready to download exploits or link them to CVEs in Metasploit. Rapid7 also offers a searching tool on the Web for vulnerabilities – most of them from the NVD – and exploit modules [76]. This work had to create a search mechanism to look for exploits in Metasploit's source code and relate them to CVEs.

### **2.8.2 Exploit-DB**

The Exploit Database, known as Exploit-DB, is a repository of exploits supported by the Offensive Security Organization [78]. It is a famous company for its information security certifications and PT consulting. The Exploit-DB provides the source code of the exploits and, sometimes, the vulnerable target applications. That repository is not a structured database but a mix of source code and text descriptions. The Exploit-DB also has the Google Hacking Database (GHDB) since 2010. In most cases, the GHDB is a list of google queries that return sensitive information that was not supposed to be public [10].

The Exploit-DB is a public database available at two URLs: <<https://www.exploit-db.com/>> and <<https://github.com/offensive-security/exploitdb>>. The first is a Web Page embedding metadata and information regarding the exploits and the target applications. The CVE reference might be available if it is the target of the exploit. The GitHub URL does not store an explicit reference to the CVE, which is only available in text

comments in the exploit’s source code. None of them offer an API to enable data access or download. So, this work had to develop a custom crawler to get the exploits and their metadata one at a time. Figure 2.17 is an example of an exploit list available on the Exploit-DB Web Page.

Date	D	A	V	Title	Type	Platform	Author
2022-04-19				EaseUS Data Recovery - 'ensserver.exe' Unquoted Service Path	Local	Windows	bios
2022-04-19				PTPublisher v2.3.4 - Unquoted Service Path	Local	Windows	bios
2022-04-19				Fuel CMS 1.5.0 - Cross-Site Request Forgery (CSRF)	WebApps	PHP	Ali J
2022-04-19				7-zip - Code Execution / Local Privilege Escalation	Local	Windows	Kağan Çapar
2022-04-19				WordPress Plugin Elementor 3.6.2 - Remote Code Execution (RCE) (Authenticated)	WebApps	PHP	AkuCyberSec
2022-04-19				PKP Open Journals System 3.3 - Cross-Site Scripting (XSS)	WebApps	PHP	Hemant Kashyap
2022-04-19				Delta Controls enteliTOUCH 3.40.3935 - Cookie User Password Disclosure	Remote	Hardware	LiquidWorm
2022-04-19				Delta Controls enteliTOUCH 3.40.3935 - Cross-Site Scripting (XSS)	Remote	Hardware	LiquidWorm

Figure 2.17: Example of an exploit list available on the Exploit-DB (from [10]).

## 2.9 VULNERS

Vulners is a data feed service that joins cybersecurity information from multiple public and private sources. Vulners is a relatively new organization created in 2015 by Russian information security professionals. It has been gaining market share as an information security data feed. There is little information on the Internet about Vulners as a company. It has two physical addresses for its headquarters: in Delaware, a tax haven in the USA, and another one in Moscow/Russia. Vulners claims to leverage more than 170 data sources to populate its data lake: different vendors, national vulnerability databases, infosec blogs, and zero-day reports. So, it returns data ranging from CVEs to exploits and infosec comments. Figure 2.18 is the main dashboard in Vulners’ Web Page [11].

Unlike most data sources presented in this chapter, Vulners is a high-priced service (€ 500 per month in April 2022). Its free option through API offers limited functionality, while the searches on the Web Page are free but challenging to integrate with external systems [11]. One of its clients is Nmap. Despite not having any commercial relations with Vulners, Nmap provides indirect data access to its data through the Nmap scripting module. This work uses Vulners as on the fly searching tool after reverse-engineering the Nmap Vulners script.

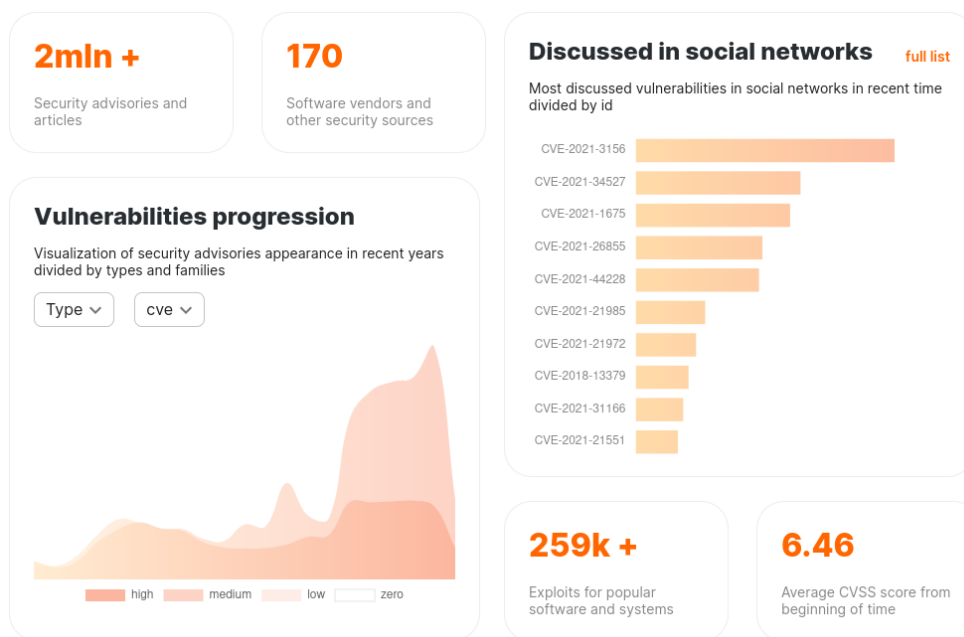


Figure 2.18: Vulners Dashboard (from [11]).

## 2.10 SUMMARY

This chapter introduced the main concepts regarding vulnerabilities and exploits. It also listed databases and sources regarding them. The following chapter reviews works about national vulnerability databases, VA methods, and vulnerability prioritization.

## 3 RELATED WORK

This chapter reviews related works and compare them with this. Section 3.1 reviews articles regarding vulnerability assessment and pentest approaches. Section 3.2 addresses works that extract and compare national vulnerability databases. Section 3.3 handles the use of recommender algorithms and MCDM approaches for vulnerability choosing or ranking. Section 3.4 summarizes this chapter.

### 3.1 VULNERABILITY ASSESSMENT

Valea and Oprisa proposed pentest automation using Nmap and Metasploit [79]. The Nmap suite provides the Nmap Scripting Engine (NSE) with Lua programming language to automate part of the scanning procedure. With NSE scripts, the pentester can get CVEs regarding vulnerable services in the target machine with some suggestions of available exploits. The work of Valea and Oprisa [79] focused only on vulnerabilities regarding a root shell with Meterpreter. So, it worked with a small set of vulnerabilities-exploits, which required algorithms like decision trees to avoid overfitting. Unlike this work, it leveraged only the NMAP and its script engine as the source of vulnerability information [79], while this work uses a mix of three national databases, Vulners and EDB. In the end, Valea and Oprisa did not realize that the Nmap vulners script gets all its data from Vulners service API.

Huo et al. used Deep Reinforcement Learning (DRL) algorithm to automate the penetration test operation [80, 81]. The method has three parts: training data, algorithm definition, model training, and evaluation. It used Shodan, CVE, Nmap, Metasploit, and Microsoft to get the training data. Regarding the model, it decided to use the Deep Q-Learning Network (DQN) method because of its plasticity to mimic human behavior. Besides, DQN is ideal for treating state transitions in a Markov Chain Model. The evaluation restricts the scope to web vulnerabilities. Also, it leverages the attack tree generation based on multi-host multi-stage vulnerability analysis (MulVAL) [82].

The DQN model aims to achieve the optimal path using the CVSS base score as a reward function. Although employing a deep learning model, the validation process was not straightforward. The training process uses Shodan data to emulate local networks. Their experiment evaluated a small topology without specification about the simulation environment and exploited only the CVE-2012-0053 [80, 81].

This section described works that showed the high computational cost of deep learning approaches for extensive networks and sometimes convergence issues. They leveraged Nmap as the main searching tool for vulnerabilities and exploits, but did not realize it was using data from Vulners. The Nmap scripts are an on the fly integration with Vulners API through the NSE. This work uses an operation research method and improves the NMAP integration with new data sources.

## 3.2 VULNERABILITY AND EXPLOIT DATABASES

Wang and Guo proposed an Ontology for Vulnerability Management (OVM) to achieve knowledge representation of the Common Vulnerabilities Exposures (CVE) of the NVD [20]. It was the first attempt to create a knowledge base (KB) of vulnerabilities, but it did not consider the exploits possibility, which is different from this work. GAO et al. created a taxonomy and ontology for network attack classification using description logic (DL) [21]. Those works guided the CVE data model that this work uses for vulnerability databases.

Kanakogi et al. [28, 30] used five approaches to map CVE to Common Attack Pattern Enumeration and Classification (CAPEC). The first is the conventional way with Common Weakness Enumeration (CWE) as a proxy. The other four approaches employed natural language processing (NLP). The former uses Term Frequency-Inverse Document Frequency (TF-IDF) to match a CVE to a CAPEC. The second one employs the encoder model USE developed by researchers at Google. The third uses the Sentence-BERT (SBERT). The fourth uses the doc2vec python library. Finally, they conclude that the TF-IDF approach outperforms the other three methods. They used only 58 CVEs from the NVD in the evaluation, not the entire set, which may bias the results [28, 30]. Those results indicated the need for new vulnerability data sources beyond the NVD.

Householder et al. [33] evaluated a systematic analysis of the relation between vulnerabilities and exploits. They proposed three research goals: percentage of CVEs exploited, speed of exploit creation, and features that influence exploit publication. By the end of 2019, only around 4.1% of the vulnerabilities exposed after 2013 had an exploit publication. The creation speed is about two days. Rodriguez et al. showed that vulnerability disclosure delay between the NVD and other open sources, e. g. in SecurityFocus, may reach 244 days [23].

Moreover, Householder et al. [33] used the following vulnerability database sources: the NVD, the CERT/CC Vulnerability Notes from Carnegie Mellon University, and TrendMicro's Zero Day Initiative (ZDI). For exploits, they used the EDB and Metasploit Framework. They found that vendor and CWE are more correlated with the probability of exploitation than CVSS, but a causal relationship was not clear [33]. Moreover, they did not comprehensively download the entire EDB to assess its exploits and metadata. They only checked for the existence of exploits regarding a set of CVEs from the NVD.

Despite several existing works regarding NVD mining, most of them leveraged some text mining (TM) approach to visualize trends and patterns [18, 32, 33], even chaotic patterns (CP) [31]. Other approaches used TM and machine learning (ML) to correlate the Common Attack Pattern Enumeration and Classification (CAPEC) with CVE [28, 30]. Lastly, there were works leveraging text mining to detect disclosure delays [23], NVD features estimations [69] and inconsistencies [66, 83]. This work learned from these articles that the CVSS and CPE features carry many inconsistencies, but none of them went beyond the NVD. Moreover, they did not leverage a data normalization (DN) approach.

A few works handled more than one national vulnerability database. Two of them used NVD, CNVD, CNNVD, and other databases to extract data, but only regarding the IoT devices [12, 13]. The National Research Institute (NASK) from Poland conducted the research Co-financed by the Connecting Europe Facility of the European Union, which resulted in those two journal articles. NASK is the central institution

for cybersecurity in Poland that works under the hood of the Chancellery of the Prime Minister. The first article tackles the lack of a vulnerability database dedicated to IoT platforms.

According to Rytel et. al [12], the information about IoT vulnerabilities does not satisfy the architecture layers' requirements: Perception, Network, and Application. Moreover, the smart device market boosts several new equipment manufacturers (OEMs), which release devices under multiple trade names, making the CPEs hard to identify. Based on this motivation, the authors proposed a survey about IoT vulnerability data feeds. They used the data sources from Table 3.1.

Table 3.1: Data sources cited in [12]

<i>Data Source</i>	<i>Description</i>	<i>Observation</i>
NVD	National Vulnerability Database from the USA (the de facto standard)	Not focused on IoT
CNVD	China National Vulnerability Database	Has some IoT amenities
CNNVD	Chinese National Vulnerability Database of Information Security	Not focused on IoT
JNVD	Japan Vulnerability Notes iPedia	Not focused on IoT
IVD	Industrial Control System (ICS) vulnerability database	IoT or ICS focused
ICS-CERT-CN	Chinese CERT branch dedicated to ICS vulnerability	IoT or ICS focused
US-CERT-ICS	US-CERT dedicated to ICS vulnerability	IoT or ICS focused
Other CERTs	CERTs from other countries besides China and USA	None
ZDI	Website of Zero Day Initiative	Not focused on IoT
Bugtraq	Forum maintained by SecurityFocus	Not focused on IoT
Vulners	Data Feed Aggregator	Not focused on IoT
Exploitee.rs	Wiki Site dedicated to IoT devices	IoT or ICS focused
Other	Blogs and Web Sites	None

Table 3.1 shows that the work [12] was a comprehensive survey about vulnerability databases but focused on IoT platforms. It is not a coincidence that [12] follows the list of data sources provided by FIRST in <https://www.first.org/global/sigs/vrdx/vdb-catalog>. With that work, this one grasped that despite some sources like the CERTs releasing new vulnerabilities sometimes sooner than the national databases, after some while, the latter end up getting that information.

In their second work [13], the authors proposed an unstructured database for IoT vulnerabilities with layers to clean data and avoid repeated entries from the raw data sources. They launched the Vulnerability and Attack Repository for IoT (VARIoT - <https://www.variot.eu/>) in 2021 due to their work. Despite the VARIoT [13] being an open database and forum regarding IoT, they do not provide access to the harvesting programs' source code. In contrast, this work provides access to its programming approaches and source

code in Python.

Rytel et al. [12] did not leverage the flaws in the CNNVD Web Page to access the XML files without using Web Parsing. Moreover, Rytel et al. focused on getting only IoT vulnerabilities and comparing the databases over the following features: CVE use, CVSS scoring, and product identification. Regarding this last feature, this work reached a different and more comprehensive conclusion about the CPE in the CNVVD because of the use of the entire set of XML files. This work also compared the CNNVD and CNVD regarding Chinese vendors, external references, and temporal correlation.

Regarding Vulners, this work and [12, 13] reached different conclusions. Despite the vastness of Vulners, when leveraging it for network scanning, it performed worst than the NVD and CNNVD. It is a consequence of the previous CPE conclusions. In common, Rytel and the authors of this work tried to contact the CNNVD support team to access the XML files through the official channels. None of us obtained any reply to our contacts. While Rytel et al. leveraged Web Crawlers to parse the HTML code, this work exploited a hidden address in the Web Page Source code to get the files.

Janiszewski et al. [13] used the survey [12] to chose the data sources for the IoT database construction. Table 3.2 lists the data sources that [13] leveraged.

Table 3.2: Data sources used in [13]

<i>Data Source</i>	<i>Type</i>	<i>Source Trust</i>	<i>Interface</i>	<i>This work</i>
NVD	Vulnerability	1.0	JSON/API/Web Page	Yes
CNVD	Vulnerability	0.6	XML/Web Page	Yes
CNNVD	Vulnerability	0.6	XML/Web Page	Yes
JNVD	Vulnerability	0.8	XML/API/Web Page	No
IVD	Vulnerability	0.2	Web Page	No
CERT/CC	Vulnerability	0.8	GitHub	No
ZDI	Vulnerability	0.7	Web Page	No
VUL-HUB	Vulnerability	0.1	Web Page	No
Vulmon	Vulnerability	0.1	Web Page	No
ZSL	Vulnerability	0.1	Web Page	No
Exploit-DB	Exploit	0.9	Web Page	Yes
Packet Storm Packet	Vulnerability/Exploit	0.1	Web Page	No
Bugtraq	Vulnerability/Exploit	0.3	Web Page	No

Table 3.2 shows that comparing [13] to [12], the main difference is the exclusion of Vulners and the addition of the Exploit-DB and Packet Storm. They also decided to use exploit databases in the second work to get more information about IoT devices. However, they did not know how to exploit the CNNVD website’s flaws to get the XML files. Thus, they kept using Web Scraping techniques to get the CNNVD data. The *Source Trust* attribute in Table 3.2, according to those authors, came from experts opinion.

The work described in [13], of which Rytel is one of the authors, is an extension of the vulnerability



data source survey detailed in [12]. Janiszewski et al. [13] develop a pipeline to build an IoT unstructured database with four steps: raw, low, medium, and high databases. Each one of the steps has the function of harvesting, standardization, aggregation, and enhancement, respectively. Unlike this work, Janiszewski et al. [13] leverage Elasticsearch and Kibana (ELK) to store and process the data over JSON, while this work used a normalization approach with PostgreSQL.

This work decided to use only the data sources with file feeds because it allows a fast database update. Web Page Scraping requires a long time of slow Web Access to get data without calling the attention of blocking mechanisms. The only exception was for Exploit-DB; this work also used a Web Scraping approach and Vulners. This last is an interesting case because they kept without realizing in both works [12, 13] that it is possible to use Vulners through reversing engineering Nmap vulners script in Lua programming language.

Lastly, Table 3.3 shows that other approaches used only CNNVD to evaluate data analysis through descriptive statistics (DS) [24], vulnerability classification with TM and ML [34] and severity prediction with deep learning (DL) [84] without comparing with the NVD. This work seeks a more extensive comparison between the NVD, CNVD, and CNNVD, leveraging them for vulnerability assessment and ranking.

Table 3.3: Comparison with Related Works I

<i>Work</i>	<i>Data Sources</i>	<i>Approach</i>	<i>Objective</i>
[18, 32, 33]	NVD	TM	Pattern Detection Data Visualization
[31]	NVD	CP	Pattern Detection
[28, 30]	NVD	TM	Link CAPEC to CPE
[23]	NVD Web Data	TM	Disclosure Delay
[69]	NVD	ML	CPE discovering
[66, 83]	NVD	TM ML	Data inconsistencies
[12, 13]	NVD CNVD, CNNVD, others Exploit-DB, Packet Storm Bugtraq	TM	Vulnerability DB for IoT
[24]	CNNVD	DS	Data Analysis
[34]	CNNVD	TM ML	Vulnerability Classification
[84]	CNNVD	DL	Severity Prediction
This work	NVD CNVD, CNNVD Exploit-DB, Vulners	TM DN	Database Comparison Vulnerability Assessment and Ranking Multilingual NLP

### 3.3 RECOMMENDER AND RANKING SYSTEMS

Pawlicki et al. wrote a comprehensive survey about the application of recommender systems for cybersecurity [85]. The authors, through a long bibliography review, proposed the following classification for the systems: Collaborative Filtering (CF), Content-Based (CB), and Knowledge-Based (KB). Besides, some approaches can combine them to form various hybrid systems. Those three approaches have advantages and disadvantages. CF relies on data from similar products, so it suffers from the cold start problem when there is no initial data. CB does not rely on data from similar cases but depends on data about the users. Both may suffer from sparsity issues. KB depends on ontology creation which might be complex and usually requires user preferences representation. That survey [85] indicates Polatidis et al. [41, 42] works as the main reference for recommender system for cybersecurity.

Polatidis et al. [41, 86] proposed an attack path prediction algorithm to achieve an information risk assessment. Their work used CWE and CVE from Mitre and IT maritime infrastructure to generate the attack graphs. After this, a recommendation system used a multi-level collaborative filtering method [42] to choose the riskiest path. Polatidis et al. focused on attack path prediction in Microsoft Windows, whose objective was gaining access to machines through the easiest attack path.

Polatidis et al. [41, 86] leveraged several qualitative criteria that depend on experts' opinions to feed and calibrate the recommender algorithm: Attack Path Analysis, Vulnerability Chain Analysis, Integration of Open Source Information, Integration of Crowd Sourcing Information, Collaboration Capabilities, Support tool, Tool availability, Pruning of paths, Propagation length, Attacker location, Attacker capability, Entry points, Target Points, Satisfaction of EU policies, risk assessment functionality, Vulnerability types, and Clarity and replication. Lima et al. [43] proposed a tridimensional matrix context-aware recommendation system to model the benign behavior of mobile device users. The CB filtering uses a cluster algorithm approach to indicate whether it is the same user, a different profile, or an abnormal one. In that case, the recommender system aimed to determine malicious behavior and not rank vulnerability issues about the system. Different from them, this work uses quantitative features from the CVSS to access the vulnerable target, not the path to it. Besides, this work leverages TOPSIS targeting any platform, not only maritime, based on Microsoft Windows.

Bączkiewicz and Wątróbski propose Python software packages to facilitate the Multi-criteria Decision Analysis (MCDA) implementation: Crispyn [87] and pyrepo-mcda [88, 89]. The first is a library with the following methods for the criteria weighting calculation: equal weighting, Standard Deviation, Statistical Variance, MEthod based on the Removal Effects of Criteria (MEREC), Coefficient of Variation (CV) [13,19], Angle, Criteria Importance Through Inter-criteria Correlation (CRITIC), Integrated Determination of Objective CRIteria Weights (IDOCRIW), Criteria Impact Loss (CILOS), Gini coefficient-based and entropy. The last method uses Shannon's theory [90] to evaluate the information entropy from the criteria data and assess the importance of each criterion.

Whereas the Crispyn [87] has weighting methods, the pyrepo-mcda [88, 89] provides the following MCDA algorithms implementation: VlseKriterijumska Optimizacija I Kompromisno Resenje (VIKOR), Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), COmbinative Distance-based ASsessment (CODAS), Weighted Aggregated Sum Product Assessment (WASPAS) Multiattribute

Boundary Approximation Area Comparison (MABAC), MULTIMOORA, Multi-Objective Optimization on the basis of Ratio Analysis (MOORA), Stable Preference Ordering Towards Ideal Solution (SPOTIS), Evaluation based on Distance from Average Solution (EDAS), Preference Ranking Organization Method for Enrichment Evaluation II (PROMETHEE II), Complex Proportional Assessment (COPRAS), Additive Ratio Assessment (ARAS), PROMETHEE for Sustainability Assessment-Criteria (PROSA-C), Compromise Ranking of Alternatives from Distance to Ideal Solution (CRADIS), Simple Additive Weighting (SAW), Measurement of Alternatives and Ranking according to Compromise Solution (MARCOS) and Additive Ratio Assessment (ARAS).

The MCDA analysis had already been used to prioritize software vulnerability types, but not the CVEs themselves [91]. Moreover, there is an example of work comparing the VIKOR and TOPSIS methods to rank vulnerability types, concluding that they have almost the same performance [92]. On the other hand, this work ranks a list of CVEs based on their features. An attempt to use a hybrid approach of subjective and objective weighting calculation with entropy has already been described for ranking material supplier [93]. As far as the author knows, no work has yet to try to rank CVEs using the CVSS attributes instead of the CVSS aggregated value. Table 3.4 presents a summary comparison between the works cited in the subsection and this work. It shows that it leverages more vulnerability database sources to a broader network scope.

Table 3.4: Comparison with Related Works II

<i>Work</i>	<i>Data Sources</i>	<i>Algorithm</i>	<i>Scope</i>
[85]	Doesn't apply	Doesn't apply	Survey
[41]	Mitre CVE	CF	MS Windows
[42]	Mitre CVE	CF	MS Windows
[43]	User profiles	CB	Mobile Devices
[45]	NVD, Metasploit	TOPSIS	5G networks
[47]	Doesn't apply	TOPSIS	IDS attributes
[46]	Private	TOPSIS	Power Systems
[87]	✗	Weighting Calculation	Software Package
[88]	✗	MCDA	Software Package
[89]	✗	Weighting Calculation and MCDA	Software Package
[91]	Mitre CVE	TOPSIS	Vulnerability Type and Ranking
[92]	Mitre CVE	VIKOR and TOPSIS	Vulnerability Type and Ranking
[93]	✗	TOPSIS	Material Supplier Type and Ranking
This work	NVD, CNVD, CNNVD, Metasploit, Exploit-DB, Vulners	TOPSIS	Database Comparison, Linux Servers, Multilingual NLP, and Exploit Correlation

Table 3.4 indicates that three works use the CF and CB recommendation algorithms: the first two [41, 42] to recommend attack paths and the last [43] to model benign behavior. None of them target vulnerability ranking. Table 3.4 also shows that three works treated the recommendation problem with an MCDM approach [45, 46, 47]. It relies on minimizing or maximizing the geometric distance from an ideal solution like the classical recommendations systems. Most of them leveraged the TOPSIS for security assessment of mobile networks on a reduced scope of vulnerabilities without considering the maturity of the exploit artifacts and for Intrusion Detection System (IDS), without leveraging multiple national vulnerability databases [45, 46, 47]. Besides, Table 3.4 lists three works that do not handle system vulnerability ranking, but they provide Python-based libraries with MCDA methods, including algorithms to objective weighting calculation [87, 88, 89]. Furthermore, it shows works regarding the application of the TOPSIS and VIKOR for ranking vulnerability types using only the Mitre CVE database and without considering the CVSS features [91, 92]. Lastly, Table 3.4 presents an article about the use of TOPSIS with entropy-based weighting for building material supplier selection [93].

### **3.4 SUMMARY**

This chapter discussed the main related works in three areas: information security assessment, vulnerability/exploit data sources, recommender, and ranking systems. The next chapter proposes an ETL data pipeline to assess, compare and link the NVD, CNVD, CNNVD, and EDB. It also proposes a recommender system to rank the vulnerabilities.

## 4 DATA COLLECTION AND PROCESSING PIPELINE

This chapter describes the architecture of the Data Collection and Processing Pipeline for Cyber Vulnerability Intelligence. Figure 4.1 presents the two steps that comprise the vulnerability and exploit databases building. Moreover, it also displays the three steps to scan, match and rank the vulnerabilities.

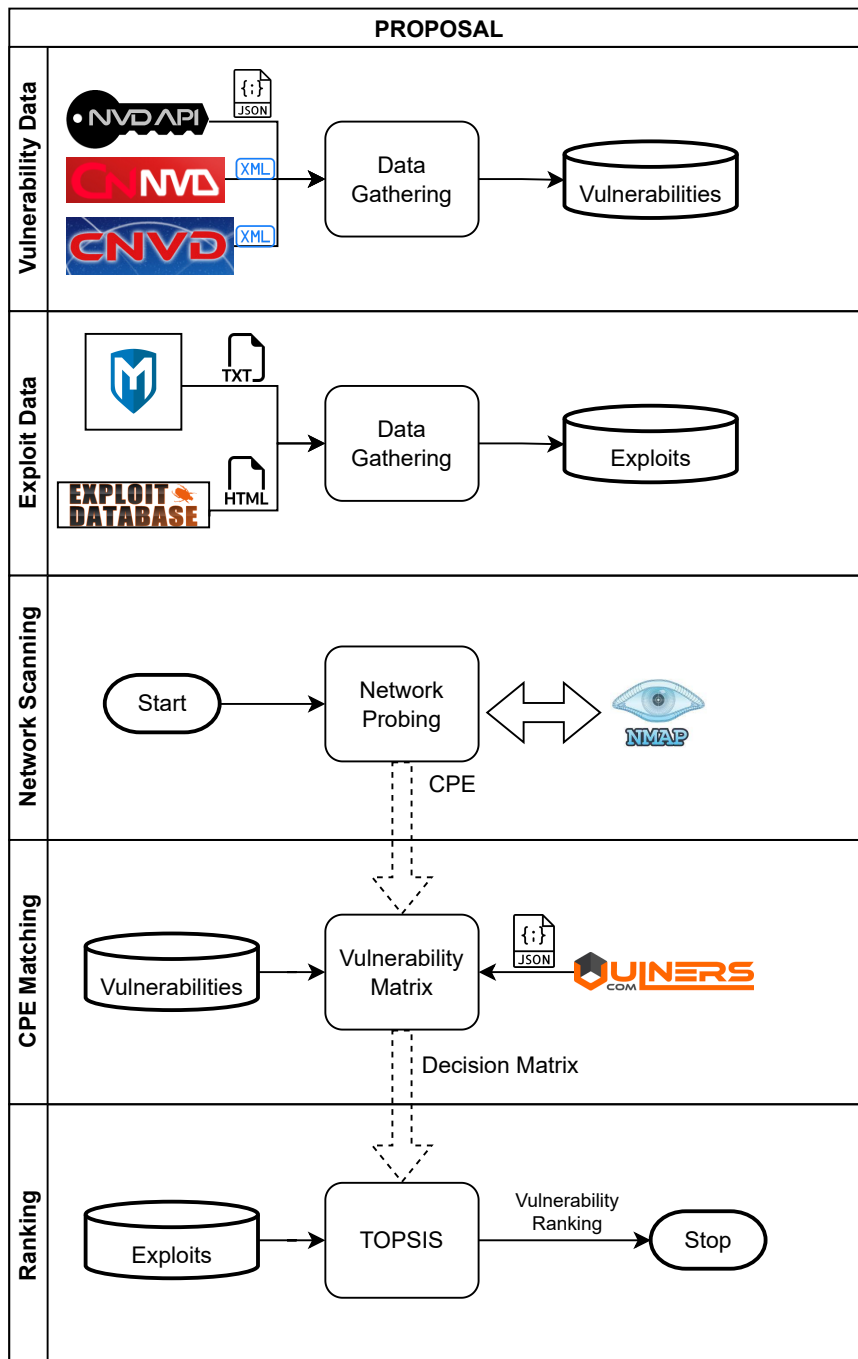


Figure 4.1: Data Collection and Processing Pipeline

Section 4.1 describes the vulnerability database building process. Section 4.2 shows the exploit database

building process. Section 4.3 presents the Scanner process. Section 4.4 describes the process to search for the vulnerability-exploit tuples. Section 4.5 shows the step for vulnerability-exploit ranking. Section 4.6 concludes this chapter.

## 4.1 VULNERABILITY DATA

This section describes the approach to getting the vulnerability data. This work normalizes and links them to evaluate the data analysis and support the vulnerability ranking and prioritization process. Figure 4.2 presents the summary of the vulnerability database building and analysis flow, which has three steps.

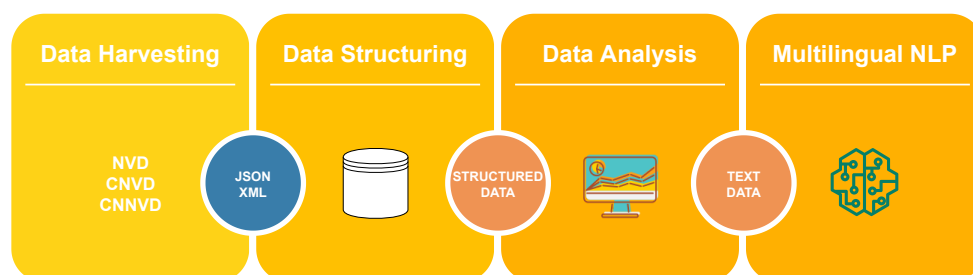


Figure 4.2: Vulnerability Database Flow.

Subsection 4.1.1 describes the data harvesting step. Subsection 4.1.2 shows the data normalization approach to create the relational database. Subsection 4.1.3 describes the database features.

### 4.1.1 Data Harvesting

The Data Harvesting step comprises the data collection of the following national vulnerability databases described in section 2.7: the NVD, CNVD, and CNNVD. They are vulnerability databases supported by state organizations. Moreover, this work starts downloading the NVD, CNVD, and CNNVD simultaneously to achieve a fair comparison.

#### 4.1.1.1 NVD

The NVD is available for download through compressed data files. They are in JSON format, archived by year since 2002. It does not mean that there are only vulnerabilities since 2002. The 2002 archive contains that year and previous vulnerabilities. Besides, there are two extra files: recent and modified. The first is the archive of new vulnerabilities in the last seven days. The latter contains the modified entries in the last month. NIST updates the year archives daily, while the two extra files are updated every two hours.

NIST also offers an API to get data receiving one CVE ID as a parameter. Figure 4.3 presents the web page with the JSON archive files. This work uses the OpenCVE [94] program to download the files and host them in a local PostgreSQL database. It provides a JSON data type that this work leverages to store the entire dataset in one batch before the data normalization process.

Feed	Updated	Download	Size (MB)
CVE-Modified	04/08/2022; 8:00:01 AM -0400	META	
		GZ	0.50 MB
		ZIP	0.50 MB
CVE-Recent	04/08/2022; 8:00:00 AM -0400	META	
		GZ	0.08 MB
		ZIP	0.08 MB
CVE-2022	04/08/2022; 3:00:02 AM -0400	META	
		GZ	0.80 MB
		ZIP	0.80 MB
CVE-2021	04/08/2022; 3:00:16 AM -0400	META	
		GZ	5.19 MB
		ZIP	5.19 MB
CVE-2020	04/08/2022; 3:00:37 AM -0400	META	
		GZ	5.19 MB
		ZIP	5.19 MB
CVE-2019	04/08/2022; 3:00:54 AM -0400	META	
		GZ	4.39 MB
		ZIP	4.39 MB
CVE-2018	04/08/2022; 3:01:10 AM -0400	META	
		GZ	3.91 MB
		ZIP	3.91 MB

Figure 4.3: NVD Download Page.

#### 4.1.1.2 CNVD

The CNVD does not offer an interface or documentation regarding data download. Moreover, the website that hosts the database in Mandarin requires user authentication and blocks access when there are many connection attempts from the same IP address. However, Figure 4.4 shows that it is possible to exploit a system flaw to download the files through a direct link.



Figure 4.4: CNVD download without logon.

The CNVD Web Page hides the base URL to download the XML data files in its source code. There are direct links to those files concatenating the base URL <https://www.cnvd.org.cn/shareData/download/> with an integer number from 1 to 1000. Some URLs do not correspond to an XML file, returning a 404 HTTP error. This work created a Python program to exploit that system flaw and download the entire XML dataset. Furthermore, the Python web scraper must set up the HTTP header with customized “User-Agent” tag and cookie (`__jsluid` and `__jsl_clearance_s`). The Web Browser sets up those parameter values during the HTTP GET Request to the CNVD Site. Listing 4.1 shows an example of a customized HTTP Tag to bypass the CNVD blocking system.

```
default_headers = {
    'Accept-Language' : 'en-US,en;q=0.5',
    'Accept-Encoding' : "gzip, deflate, br",
    "Sec-Fetch-User" : "?1",
    "User-Agent": "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:95.0) Gecko/20100101\
Firefox/95.0",
    "Accept" : "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,\
image/webp,*/*;q=0.8",
    "Accept-Encoding" : "gzip, deflate, br",
    "Connection" : "keep-alive",
    "Host" : "www.cnvd.org.cn",
    "Sec-Fetch-Dest" : "document",
    "Sec-Fetch-Mode" : "navigate",
    "Sec-Fetch-Site" : "none",
    "Sec-Fetch-User" : "?1",
    "Upgrade-Insecure-Requests" : "1",
    "Host": "www.cnvd.org.cn",
    "Cookie": "__jsluid_s=3f0307d50bb07b51034147cc24d25413;\
__jsl_clearance_s=l641480308.561|0|dkCFvrbJVih2aIEVVilqxemRkoM%3D",
}
```

Listing 4.1: HTTP GET Configuration

#### 4.1.1.3 CNNVD

The CNNVD site is also in Mandarin like the CNVD and claims to be an open database. After profiling the site, this work located the download section for the CNNVD dataset in the URL <http://www.cnnvd.org.cn/web/xxk/xmlDown.tag> as shown in Figure 4.5. It shows that the CNNVD download page looks like the NVD page: one file for each year from 1999 to 2022. It also provides two more files: daily updates in the first row and monthly updates in the second. However, the download attempts result in the error message shown in Figure 4.6.

The message box in Figure 4.6 returns the following message translated to English: “Not logged in, please log in first!”. It indicates that CNNVD requires user authentication, but no signup options exist. After profiling the Web Page, it suggested contacting the system administrator through the email `cnnvd@itsec.gov.cn`, which has not answered this work request for access.



年份	更新日期	下载	大小 (MB)
最新	2022-02-09	XML	1.71
当月	2022-02-09	XML	3.76
2022	2022-02-09	XML	1.73
2021	2022-01-06	XML	25.43
2020	2022-01-06	XML	22.64
2019	2022-01-06	XML	24.98
2018	2022-01-06	XML	13.72
2017	2022-01-06	XML	8.91
2016	2022-01-06	XML	3.92
2015	2022-01-06	XML	2.78
2014	2022-01-06	XML	3.46
2013	2022-01-06	XML	3.61
2012	2022-01-06	XML	4.24
2011	2022-01-06	XML	5.03
2010	2022-01-06	XML	2.79
2009	2022-01-06	XML	2.54
2008	2022-01-06	XML	2.32
2007	2022-01-06	XML	2.66
2006	2022-01-06	XML	2.49
2005	2022-01-06	XML	1.85
2004	2022-01-06	XML	1.32
2003	2022-01-06	XML	0.88
2002	2022-01-06	XML	1.30
2001	2022-01-06	XML	0.64
2000	2022-01-06	XML	0.36
1999及之前	2022-01-06	XML	1.31

Figure 4.5: CNNVD Download Site.

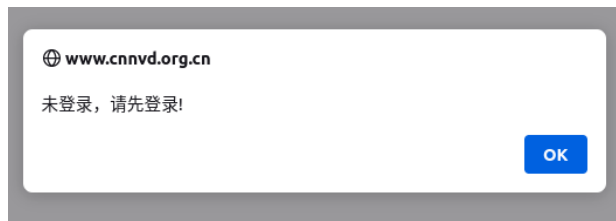


Figure 4.6: CNNVD Error Message.

Regardless, the page source code hides the download URL to each file: the string pattern `<http://www.cnnvd.org.cn/>` concatenated with the file name available in the source code of the screen presented in Figure 4.5. Figure 4.7 shows that, like the CNVD system, it is possible to download CNNVD XML files without signing in. It is also worth mentioning that the CNNVD uses the HTTP protocol instead of HTTPS.



Figure 4.7: CNNVD download without logon.

## 4.1.2 Data Structuring

The JSON and XML data types are not the best option for data and text processing with Python in PostgreSQL. So, this work evaluates a database normalization of the NVD. It also uses the same approach for the CNVD and CNNVD, with the difference that the NVD has deeply nested JSON. Besides, the CPE in the NVD includes logical conditions that do not exist in the CNVD and CNNVD. Moreover, the NVD has CVSS information unavailable in the other two.

An overview of entries in the JSON and XML files can help to grasp the differences between those three data feeds. Figure 4.8 shows the entry in the NVD for the CVE-2020-15009.

```
▼ cve:
  data_type: "CVE"
  ▼ references:
    ▼ reference_data:
      ▼ 0:
        url: "https://drive.google.com/file/d/1ClzHH5Jw3PgZw74RvKrEP8xU0Tuc5Ta0/view?usp=sharing"
        name: "https://drive.google.com/file/d/1ClzHH5Jw3PgZw74RvKrEP8xU0Tuc5Ta0/view?usp=sharing"
        tags:
          0: "Third Party Advisory"
        refsource: "MISC"
      ▼ 1:
        url: "https://www.asus.com/support/FAQ/1043674"
        name: "https://www.asus.com/support/FAQ/1043674"
        tags:
          0: "Vendor Advisory"
        refsource: "CONFIRM"
      ▼ 2:
        url: "https://www.asus.com/Static_WebPage/ASUS-Product-Security-Advisory/"
        name: "https://www.asus.com/Static_WebPage/ASUS-Product-Security-Advisory/"
        tags:
          0: "Vendor Advisory"
        refsource: "CONFIRM"
    data_format: "MITRE"
  ▼ description:
    ▼ description_data:
      ▼ 0:
        lang: "en"
        value: "AsusScreenXpertService.exe and ScreenXpertUpgradeServiceManager.exe in ScreenPad2_U additional restrictions when a user puts an application at a particular path with a p"
  ▼ problemtype:
    ▼ problemtype_data:
      ▼ 0:
        description:
          ▼ 0:
            lang: "en"
            value: "CWE-426"
    data_version: "4.0"
  ▼ CVE_data_meta:
    ID: "CVE-2020-15009"
    ASSIGNER: "cve@mitre.org"
```

Figure 4.8: Vulnerability Entry in JSON File from the NVD.

Besides the CVE element with data regarding the external references, description and the CWE in the Figure 4.8, the JSON file has one separated element for the CPE description and other one for the CVSS data. Figures 4.9 and 4.10 present those elements.

```
▼ configurations:
  ▼ nodes:
    ▼ 0:
      children: []
      operator: "OR"
      ▼ cpe_match:
        ▼ 0:
          cpe23Uri: "cpe:2.3:a:asus:screenpad2_upgrade_tool:1.0.3:*:*:*:*:*"
          cpe_name: []
          vulnerable: true
      CVE_data_version: "4.0"
```

Figure 4.9: The CPE element in JSON File from the NVD.

```

  impact:
    baseMetricV2:
      cvssV2:
        version: "2.0"
        baseScore: 4.4
        accessVector: "LOCAL"
        vectorString: "AV:L/AC:M/Au:N/C:P/I:P/A:P"
        authentication: "NONE"
        integrityImpact: "PARTIAL"
        accessComplexity: "MEDIUM"
        availabilityImpact: "PARTIAL"
        confidentialityImpact: "PARTIAL"
        severity: "MEDIUM"
        acInsufInfo: false
        impactScore: 6.4
        obtainAllPrivilege: false
        exploitabilityScore: 3.4
        obtainUserPrivilege: false
        obtainOtherPrivilege: false
        userInteractionRequired: true
      baseMetricV3:
        cvssV3:
          scope: "UNCHANGED"
          version: "3.1"
          baseScore: 7.8
          attackVector: "LOCAL"
          baseSeverity: "HIGH"
          vectorString: "CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H"
          integrityImpact: "HIGH"
          userInteraction: "REQUIRED"
          attackComplexity: "LOW"
          availabilityImpact: "HIGH"
          privilegesRequired: "NONE"
          confidentialityImpact: "HIGH"
          impactScore: 5.9
          exploitabilityScore: 1.8
          publishedDate: "2020-07-20T13:15Z"

```

Figure 4.10: The CVSS element in JSON File from the NVD.

In order to compare the data feeds and grasp a data model, Figures 4.11 and 4.12 show the entries in the CNVD and CNNVD for the same CVE picked up in the NVD, CVE-2020-15009.

```

<vulnerability>
  <number>CNVD-2022-04723</number>
  <cves>
    <cve>
      <cveNumber>CVE-2020-15009</cveNumber>
      <cveUrl>https://nvd.nist.gov/vuln/detail/CVE-2020-15009</cveUrl>
    </cve>
  </cves>
  <title>ASUS ScreenPad2 Upgrade Tool代码问题漏洞</title>
  <severity>中</severity>
  <products>
    <product>ASUS ASUS ScreenPad2 Upgrade Tool 1.0.3</product>
  </products>
  <isEvent>通用软硬件漏洞</isEvent>
  <submitTime>2020-07-21</submitTime>
  <openTime>2022-01-18</openTime>
  <referenceLink>
    https://drive.google.com/file/d/1ClzHH5Jw3PgZw74RvKrEP8xU0TUc5Ta0/view?usp=sharing
  </referenceLink>
  <formalWay>
    厂商已发布了漏洞修复程序，请及时关注更新： https://www.asus.com/Static_WebPage/ASUS-Product-Security-Advisory/
  </formalWay>
  <description>
    ASUS ScreenPad2 Upgrade Tool是中国台湾华硕（ASUS）公司的一款用于ASUS ScreenPad2触控板的更新工具。ASUS ScreenXpertUpgradeServiceManager.exe文件存在代码问题漏洞。攻击者可利用该漏洞执行代码。
  </description>
  <patchName>ASUS ScreenPad2 Upgrade Tool代码问题漏洞的补丁</patchName>
  <patchDescription>
    ASUS ScreenPad2 Upgrade Tool是中国台湾华硕（ASUS）公司的一款用于ASUS ScreenPad2触控板的更新工具。ASUS ScreenXpertUpgradeServiceManager.exe文件存在代码问题漏洞。攻击者可利用该漏洞执行代码。目前，供应商发布了安全公告
  </patchDescription>
</vulnerability>

```

Figure 4.11: Vulnerability entry in XML File from the CNVD.

```

-<entry>
  <name>ASUS ScreenPad2 Upgrade Tool 代码问题漏洞</name>
  <vuln-id>CNNVD-202007-1304</vuln-id>
  <published>2020-07-20</published>
  <modified>2020-07-31</modified>
  <source/>
  <severity>高危</severity>
  <vuln-type>代码问题</vuln-type>
  <thrtype>本地</thrtype>
  <vuln-descript>
    ASUS ScreenPad2 Upgrade Tool是中国台湾华硕（ASUS）公司的一款用于ASUS ScreenPad2触控板的更新工具。ASUS ScreenPa
    存在代码问题漏洞。攻击者可利用该漏洞执行代码。
  </vuln-descript>
  <vulnerable-configuration/>
  <vuln-software-list/>
  <other-id>
  <cve-id>CVE-2020-15009</cve-id>
  <bugtraq-id/>
  </other-id>
  <refs>
  <ref>
    <ref-source>MISC</ref-source>
    <ref-name/>
    <ref-url>
      https://drive.google.com/file/d/1ClzHH5Jw3PgZw74RvKrEP8xU0TUc5Ta0/view?usp=sharing
    </ref-url>
  </ref>
  <ref>
    <ref-source>CONFIRM</ref-source>
    <ref-name/>
    <ref-url>
      https://www.asus.com/Static_WebPage/ASUS-Product-Security-Advisory/
    </ref-url>
  </ref>
  <ref>
    <ref-source>CONFIRM</ref-source>
    <ref-name/>
    <ref-url>https://www.asus.com/support/FAQ/1043674</ref-url>
  </ref>
  <ref>
    <ref-source>nvd.nist.gov</ref-source>
    <ref-name/>
    <ref-url>https://nvd.nist.gov/vuln/detail/CVE-2020-15009</ref-url>
  </ref>
  </refs>
  <vuln-solution>
    目前厂商已发布升级补丁以修复漏洞，补丁获取链接： https://www.asus.com/Static_WebPage/ASUS-Product-Security-Advisory/
  </vuln-solution>
</entry>

```

Figure 4.12: Vulnerability entry in XML File from the CNNVD.

Despite not providing an XML Schema Definition (XSD), it is possible to infer a data model to the CNVD and CNNVD. Besides the XML tags describing the vulnerability, they have a tag list equivalent to the CPE list in the NVD: `<vuln-software-list>` in the CNNVD and `<products>` in the CNVD. Although not the CVSS data are not available, they provide a severity classification tag: `<serverity>` in the CNVD and `<severity>` in the CNNVD. By the way, the CNVD and CNNVD misspell some XML tags in English.

From the JSON and XMLs data feeds overview, this work created a relational data model to store the data from those files in a PostgreSQL database. Figure 4.13 depicts the data model with the main tables. It shows that the central entity is the vulnerability table. Besides, there are other entities for the CVE metrics: CVSS, description, CWE, CPE, and references. These last are data like URLs for external sites or exploits.

The CNVD and CNNVD use a database model very similar to the NVD. The main difference is that they do not carry information about complex CPEs. This work stores the complex and basic CPEs in different tables to enhance the data analysis. The CNVD and CNNVD only list the vulnerable platforms without considering logical conditions. Furthermore, there are no data about CWE in these databases. This work leverages the Pandas library and list compression technique with the Python Multiprocessing library to accelerate the file reading, processing, and data normalization. Moreover, the data processing includes a data cleaning to remove the `<\r>`, `<\n>`, `<\t>`, and `<\\>` characters.

This work uses the Python Multiprocessing library for fast creation and loading of the data from JSON and XML files to PostgreSQL. It allows throwing  $n$  parallel processes, leveraging a multi-core CPU

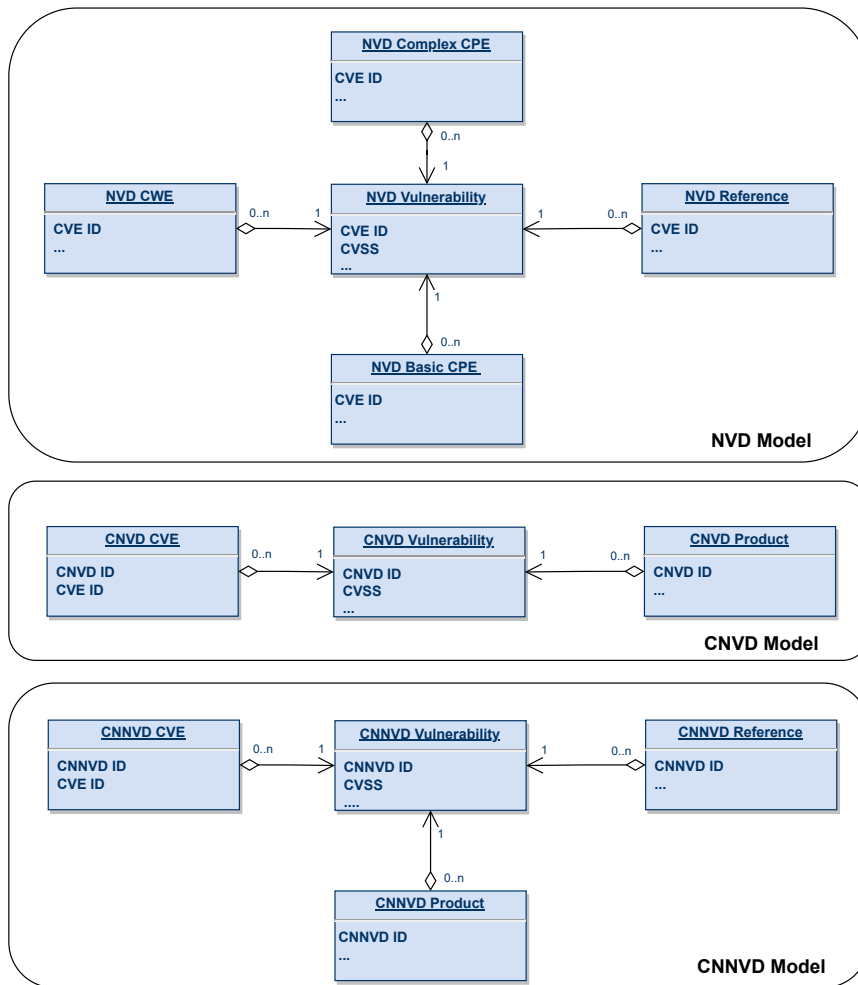


Figure 4.13: Data Models.

architecture, which is ubiquitous nowadays. Each parallel process is responsible for reading, processing, and loading one file into the relational database. Figure 4.14 depicts the Extract Transact and Load (ETL) data pipeline built with Python and PostgreSQL.

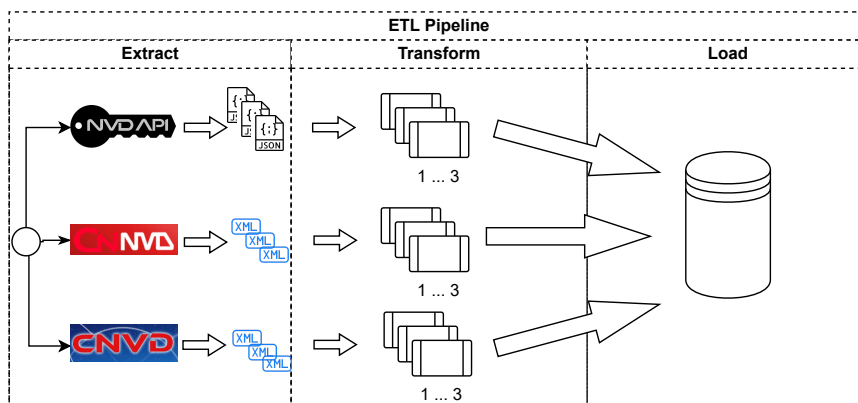


Figure 4.14: ETL Pipeline.

Figure 4.14 shows the same basic structure for loading JSON and XML files. The main difference between them is the step for file reading. The NVD files are deeply nested JSON elements, which requires

a normalization approach with the method *json\_normalize* from the *Pandas* library. This work uses the *xml.etree.ElementTree* API to handle the XML files from the CNVD and CNNVD. For the database schema creation, this work leverages the method *to\_sql* from the *Pandas* library with the *SQLAlchemy* engine.

### 4.1.3 Data Analysis

This step depends on the relational database created in PostgreSQL. It seeks data descriptions and inconsistencies in those three national vulnerability databases. So, it requires the calculation of numeric features of each database regarding the attributes presented in Table 4.1.

Table 4.1: Vulnerability Database Attributes

<i>Feature</i>	<i>Target Database</i>
Vulnerability Count	NVD, CNVD, CNNVD
Missing CPE	NVD, CNVD, CNNVD
Missing CVSS	NVD, CNVD, CNNVD
Missing Reference	NVD, CNVD, CNNVD
Missing CVE	CNVD, CNNVD
Missing Data	NVD, CNVD, CNNVD
Duplicated CVE	CNVD, CNNVD
CVSS version	NVD, CNVD, CNNVD
Missing CWE	NVD, CNVD, CNNVD

The attributes of Table 4.1 allow a database summary and tracking of each database’s evolution since the historical series’s beginning. After grasping the numeric features of the NVD, CNVD, and CNNVD, it is possible to compare them and look for data inconsistencies. This work will also seek a possible numeric correlation between the historical series of those three national vulnerability databases.

Lastly, this work leverages text mining techniques in PostgreSQL to seek uncommon vulnerability descriptions, vendors, and platforms. This part pays more attention to comparing Chinese software and hardware industry occurrences, e. g. Huawei, Xiaomi, and ZTE, in each one of the databases.

### 4.1.4 Multilingual NLP

More than checking the data features of the national vulnerability databases, this work proceeds a text comparison between them. In order to check if the Chinese databases are only copying the NVD, this work proceeds with a text comparison between the NVD and CNNVD entries that share the same CVE id. Nevertheless, the NVD text is in English, while the CNNVD is in Chinese, making the comparison difficult. Initially, it would be necessary at least to translate the Chinese databases to English before comparing them with the NVD. Evaluating a comparison model or manual checking would be necessary even after translation. This last option is almost impossible because there are more than one hundred thousand CVEs

in the CNNVD. Besides, high-quality translation services would be necessary, which are not free of charge nowadays. Therefore, this work decided to use a semantic text similarity approach.

This work uses a transfer model to create a word embedding for each vulnerability description to avoid the translation step. The word embedding is a numeric text vector representing each vulnerability description. Theoretically, the vectors representing identical texts have a high cosine similarity regardless of the language. Thus, after generating the word embedding for the vulnerability descriptions set, this work calculates the cosine distance between them to assess the similarity between the entries with the same CVE id. Figure 4.15 describes this work comparison approach.

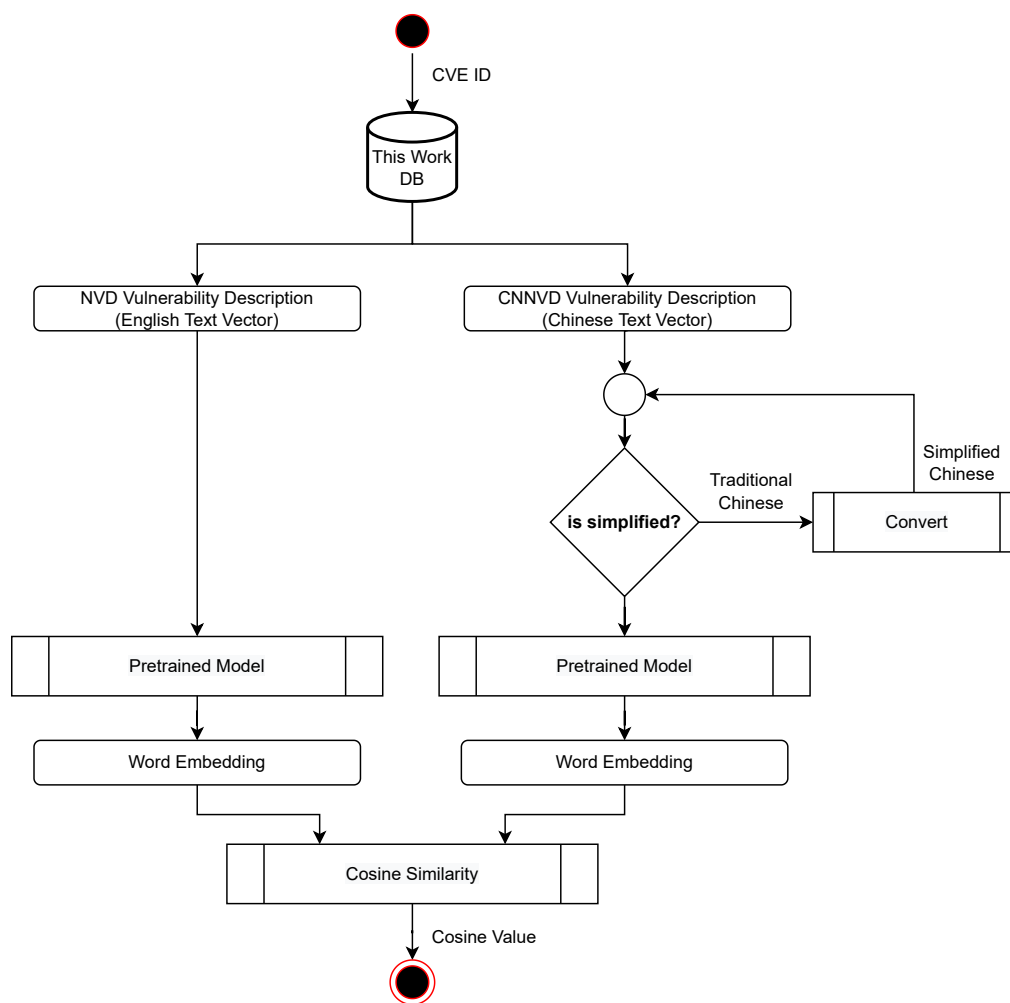


Figure 4.15: Text Comparison Scheme.

Figure 4.15 describes a scheme that requires a multilingual model to handle texts in English and Chinese. The last type of text also may be in traditional or simplified Chinese. However, despite China’s use of simplified Chinese as a standard, too much legacy content is written in the traditional version. Since the pre-trained models can handle only simplified Chinese, the scheme displayed in Figure 4.15 requires a Chinese version checking and conversion step. This work uses the Hanzi Identifier Python-based library [95] for the checking and the Hanzi Converter Python-based library [96] for the conversion. The following diagrams omit the Chinese language checking and conversion block for simplification purposes.

This work applies pre-trained transfer models on a representative sample set of vulnerabilities from

the NVD and CNNVD. Considering a population of 168,398 entries referencing the same CVE id in the NVD and CNNVD, a representative sample set with 95% of confidence level and 5% of maximum deviation must have at least 384 samples. After getting the vulnerability samples randomly from those databases, the Chinese text of each entry is manually translated to English using two different services: Google translate and DeepL libraries.

After the translation process, this work uses the following five pre-trained models from Sentence Transformers [97]: distiluse-base-multilingual-cased-v1, distiluse-base-multilingual-cased-v2, paraphrase-multilingual-MiniLM-L12-v2, LaBSE and paraphrase-multilingual-mpnet-base-v2 [98, 99, 100]. This work chose those five models because they provide support for multilingual NLP, including English and Chinese languages, according to [98, 99, 100].

Before evaluating those five multilingual pre-trained models, this work uses a histogram of the word count of the vulnerability descriptions in the NVD and CNNVD. Through the analysis of the histogram models, it is possible to reach a *max\_word\_length* parameter value for those models to fit most of the texts without truncation. Figure 4.16 shows the sampling approach that uses the preceding comparison scheme.

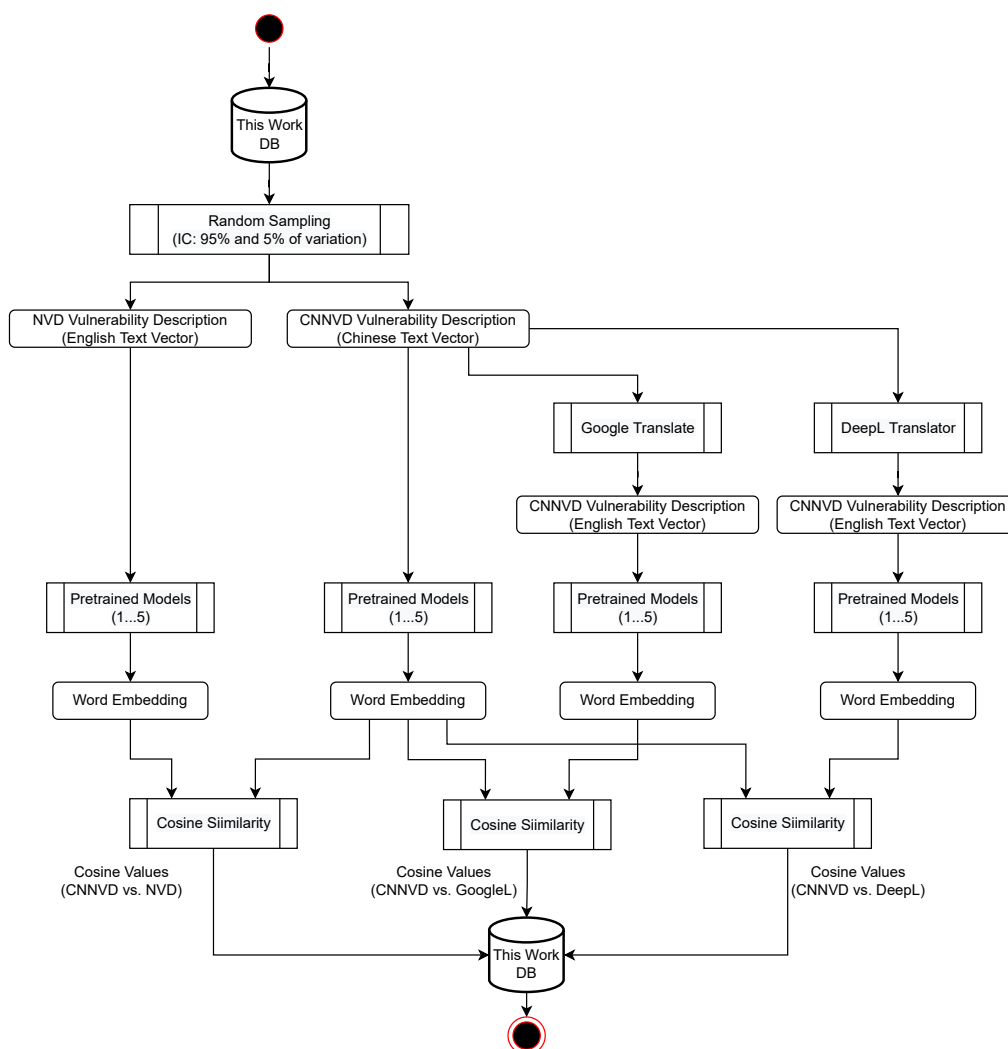


Figure 4.16: Model Test Scheme.



Figure 4.16 shows the process flow to compare each Chinese vulnerability description against three texts: the Google-translated version, the Deepl-translated version, and the NVD description. Thus, it calculates the cosine similarity for each text comparison. After evaluating the calculation for the sample set using those five transfer models, this work computes the standard and weighted average of the cosine similarities to choose the model with the best performance. Beyond calculating the similarities, this work also gauges the execution performance of the models. Figure 4.17 shows the testing scheme.

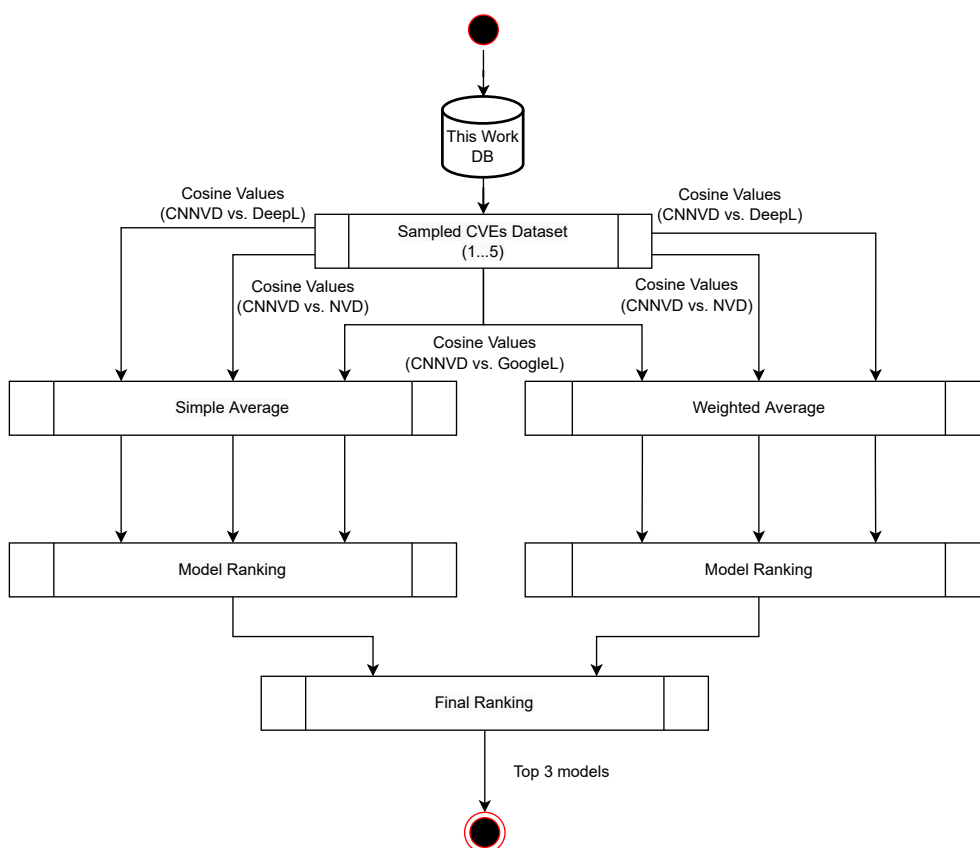


Figure 4.17: Ranking Models Scheme.

Figure 4.17 shows that the models with the results closest to one (100%) present the best performance. With it, this work creates a scheme to evaluate the comparison of the NVD and CNNVD entries with the same CVE reference. Considering the amount of text, it may take too much time for CPU processing, even with multi-core processing. Moreover, the pre-trained neural network models depend on intensive matrix calculations, slowing the overall process even further. In that case, a GPU hardware with hundreds of unit cores tuned to perform the same multiplication operation in parallel over different data can perform much better than a CPU — usually with less than ten general purpose cores — in model training and evaluation [101].

Thereby, this work decided to leverage GPU processing with PyTorch and Compute Unified Device Architecture (CUDA) to speed up the encoding of the text vectors into numerical vectors (word embedding) and the similarity calculation. Since the data volume may be larger than the Dedicated Video RAM (VRAM), it may require splitting them into data chunks to fit into the GPU memory for each calculation batch. Figure 4.18 describe the GPU approach.

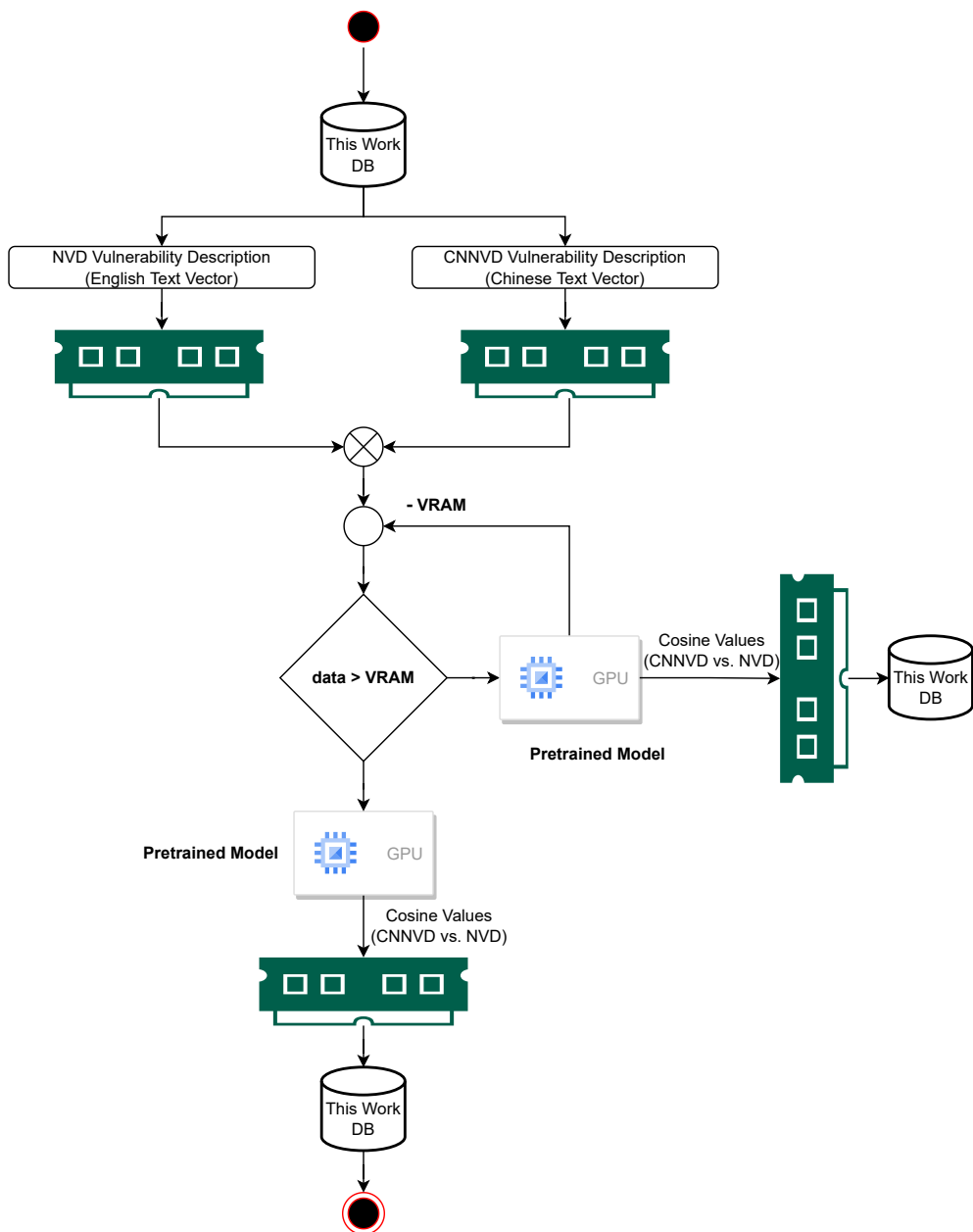


Figure 4.18: GPU implementation.

## 4.2 EXPLOIT DATA

This section describes the procedure to get the exploits data. Unlike the vulnerabilities data that state organizations usually provide, the community or private organization gets used to supporting the exploit repositories. Sometimes, the national vulnerability databases mentioned before getting data from exploit archives to enrich their knowledge base, as this work will demonstrate. This work normalizes the unstructured data from two exploit repositories and links them to the previously collected vulnerability databases. Figure 4.19 presents the summary of the database methodology flow, which has three steps.

Subsection 4.2.1 describes the data harvesting step. Subsection 4.2.2 shows the data normalization approach to create the relational database. Subsection 4.2.3 describes the database features to link and

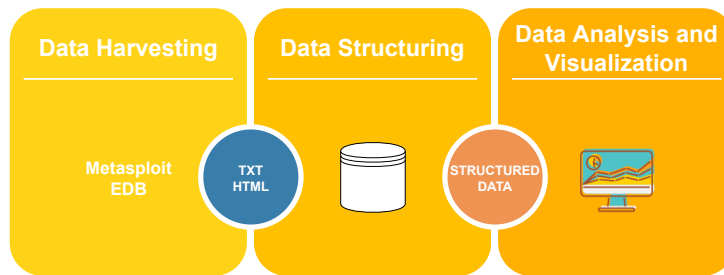


Figure 4.19: The Exploit Database Summary Flow.

compare with the national vulnerability databases.

## 4.2.1 Data Harvesting

The Data Harvesting step comprises the data collection of the following exploit databases: the Metasploit Framework (MSF) and Exploit Database (EDB). They are archives of exploits provided by private organizations.

### 4.2.1.1 Metasploit

More than a system, the MSF is a framework integrating reconnaissance tools, exploits, and payloads in the same environment. The EDB is supposed to list every exploit from the MSF. Moreover, the MSF implements only a subset of the exploits from the EDB, most of them based on Ruby programming language. As the exploits in it leverage interpreted languages as described in subsection 2.8, this work uses a custom python program to parse MSF's modules and search for CVEs references. Listing 4.2 reproduces the python code snippet that executes that CVE pattern search.

```
import shutil, sys
import subprocess
import pandas as pd

cmd = ['grep', '--recursive', '--line-number', '--extended-regexp', '--only-matching', \
      '/opt/metasploit-framework/embedded/framework/modules', \
      "--regex=[\\'\\[\\ \\ ]{1,3}CVE[\\'\\ \\ ]{1,4}[0-9]{4}\\-[0-9]{1,5}'? ?]?"]

a = subprocess.Popen(cmd, stdout=subprocess.PIPE)

if sys.version_info[0] < 3:
    from StringIO import StringIO
else:
    from io import StringIO

buff = StringIO(a.communicate()[0].decode('utf-8'))

header_list = ["module", "line", "cve"]
df = pd.read_csv(buff, sep=":", names = header_list)

df['cve'] = df['cve'].str.replace('[\\' \\[\\ ]*', '', regex=True).str.replace(',', '-')

print(df)
}
```

Listing 4.2: MSF Parser Code Snippet

Listing 4.2 shows that this work searches for the pattern <CVE-YYYY-DDDD> in Metasploit's modules directory. After getting the CVEs list, this work stores it in the PostgreSQL table for joining search in the national vulnerability databases and EDB.

#### 4.2.1.2 The Exploit Database (EDB)

The EDB is an exploit archive supported by the Offensive Security Organization [78] as explained in section 2.8. It is a repository of unstructured data because it neither provides an API for input calls nor offers semistructured data like JSON and XML files. The EDB is a collection of exploit source codes and their metadata embedded in the HTML and Javascript of the Web Page.

Different from the three vulnerability databases, the EDB requires a custom crawler to profile its Web Page Source code to get the exploit source codes and their metadata one by one. The EDB Web page hosts each exploit in a separate address with the following URL pattern: <https://www.exploit-db.com/exploits/> concatenated with an integer number which is the exploit identification. Figure 4.20 is an example provided by EDP for the exploit <https://www.exploit-db.com/exploits/900>.

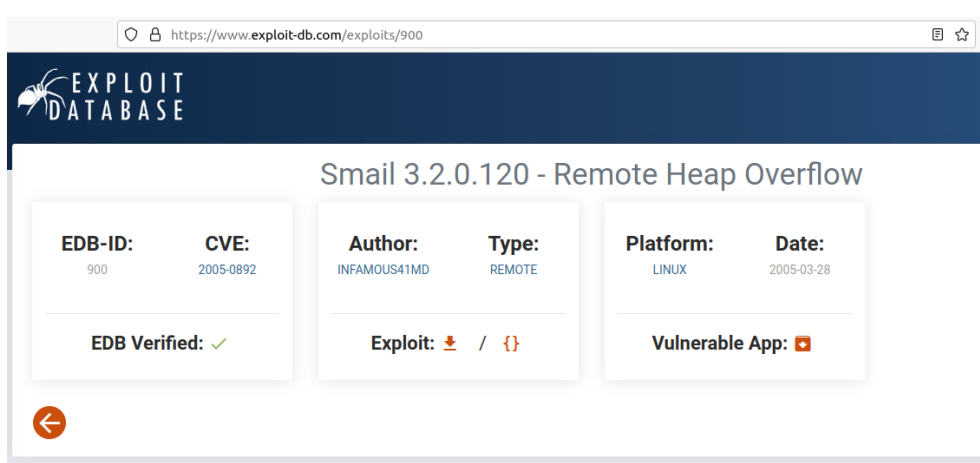


Figure 4.20: Example of Exploit Data from EDB.

Figure 4.20 also shows that exploit id is the same integer number that comprises the URL. It also shows from the rendered HTML page that it is possible to get the following attributes:

- **EDB-ID:** exploit id;
- **CVE:** exploited CVE id;
- **Author:** author's name of the exploit;
- **Type:** exploit classification;
- **Platform:** classification of the platform that hosts the application target;
- **Date:** publish date;
- **EDB Verified:** already tested;

- **Exploit:** source code;
- **Vulnerable App:** binary of the vulnerable application.

## 4.2.2 Data Structuring

On the other hand, crawling the EDB Web Page requires getting the site entries one by one, making this task IO and not CPU bound. Then, the multiprocessing approach used to handle the vulnerability data feeds does not work. In that case, it is advisable to use a multithreading technique. Nevertheless, even being an IO-bound task requires a balance between the multithreading approach to speed up the download versus the possibility of being IP blocked by an IDS in the EDB site.

The *Thread* class from the *threading* library enables multithreading in the Python programming language. This work decided not to stress the EDB Web page and did not leverage a multithreading approach to speed up the database download. After getting the data through sequential calls using the *Requests* and *BeautifulSoup* libraries, this work handled them in the same way as the vulnerability databases with *Pandas* and *SqlAlchemy* libraries. Figure 4.21 represents the Exploit database model.

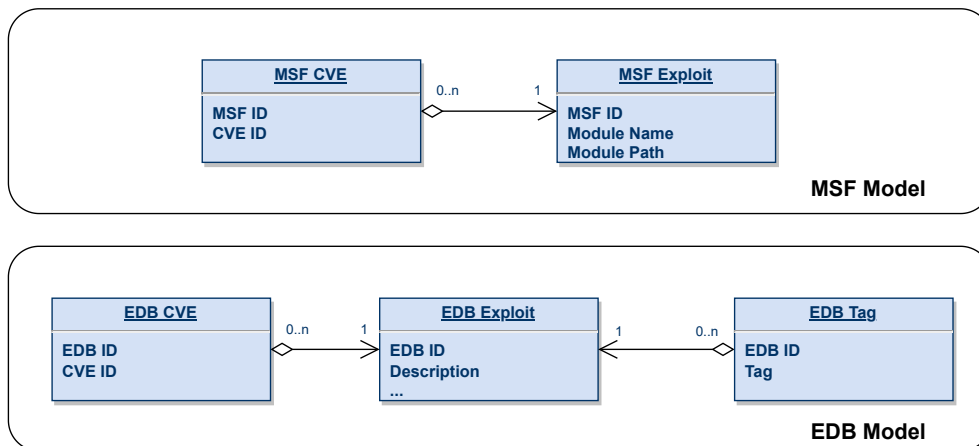


Figure 4.21: Data Model for Exploit-DB.

## 4.2.3 Data Analysis and Visualization

This step depends on the relational databases created for the national vulnerabilities and exploit data. First, it calculates aggregated values regarding exploit count and missing references to CVEs. Second, this work groups the exploits according to the type and author.

After those two steps, it evaluates a cross-reference between the EDB and the NVD, CNVD, and CNNVD. The objective is to realize which are more up-to-date to exploit maturity and if the Chinese databases are using the EDB to grasp new vulnerabilities before the NVD. Thus, it requires the calculation of numeric features of each database regarding the attributes presented in Table 4.2.

Table 4.2: Exploit Database Attributes

<i>Feature</i>	<i>Target Database</i>
Exploit Count	MSF and EDB
Missing CVE	EDB
Exploit Groups	EDB
Exploit Authors	EDB
Cross-Reference	EDB, NVD, CNVD, and CNNVD

### 4.3 SCANNER

The work implements two scanning methods: network and host. The former returns the list of running nodes in a target network. The later returns the list of CPEs for running services in a target node. The former uses the NMAP function to detect the network hosts' situation (up or down). The latter also leverages NMAP to identify CPEs of the opening services in each up machine. While the CPE is the key to searching for entries in the created vulnerability database, the CVE is the key to linking them to entries in the exploit database.

This work built a new Python class to wrap the NMAP scanning functions, as shown in Listing 4.3. It shows that the class implements two functions: *scanNetworkStatus* to check for machines availability in a network and *scanFullTarget* to list exposed services/CPEs in a target machine. The more detailed the CPE information, the more likely the chance to precisely match exploitable vulnerabilities for the target in the Matcher module.

```
class NmapScanner:

    def __init__(self):
        self.scanner = nmap.PortScanner()
        print("Starting REVS...")
        print("Starting Nmap...")

    def scanFullTarget(self, ip):

        nm = self.scanner
        nm.scan(hosts=ip)
        serv_list = list()

        for proto in nm[ip].all_protocols():
            lport = nm[ip][proto].keys()
            for port in lport:
                serv_list.append({"port":port, "state":nm[ip][proto][port]['state'],\
                                "protocol":proto, "service_name":nm[ip][proto][port]['product'],\
                                "cpe":nm[ip][proto][port]['cpe']})
        return serv_list

    def scanNetworkStatus(self, network):
        nm = self.scanner
        nm.scan(hosts=network, arguments='-n -sP -PE -PA21,23,80,3389')
        hosts_list = [(x, nm[x]['status']['state']) for x in nm.all_hosts()]

        return hosts_list
```

Listing 4.3: Python Wrapper to Nmap

## 4.4 MATCHER

This step uses the results from the prior one to search for security data in the following databases: Vulnerabilities, Exploits, and Vulners API. The last is a web service provided by Vulners [11], an Information Security Company from Russia as described in section 2.9. The other two are the PostgreSQL databases described in sections 4.1 and 4.2, respectively.

The Vulners data provides a cross-validation layer to the vulnerability and exploit databases that this work created before. An interface to Vulners database is embedded in NMAP and coded in Lua programming language: *vulns script*. The work of [79] leveraged it to carry out the MSF automation. However, this work requires more vulnerability features to create the decision matrix for the TOPSIS algorithm. So, this work created a new wrapper to Vulners API (<<https://vulners.com/api/v3/burp/software?>>) through a Python class. Listing 4.4 shows a code snippet of that Python wrapper.

```
class VulnersScraper:
    def __init__(self):
        self.vulners_url = 'https://vulners.com/api/v3/burp/software?'
        print("Gathering Vulnerabilities and Exploits...")
    def getVulnersCPE(self, vuln_software, vuln_version, vuln_type):
        vulners_lst = []
        vulners_dict = {}
        session = HTMLSession()
        url = self.vulners_url + "software=" + vuln_software + '&version=' + \
            vuln_version + "&type=" + vuln_type
        response = session.get(url=url)
        if response.json()['result'] == 'warning':
            return response.json()['result'], response.json()['data']['warning']
        resp = response.json()['data']['search']
        for line in resp:
            line_dict = {}
            line_dict['nmap_cpe'] = vuln_software + ':' + vuln_version
            line_dict['id'] = line['_source']['id']
            line_dict['type'] = line['_source']['type']
            line_dict['bulletin_family'] = line['_source']['bulletinFamily']
            line_dict['title'] = line['_source']['title']
            line_dict['description'] = line['_source']['description']
            line_dict['cvss_score'] = line['_source']['cvss']['score']
            line_dict['cvss_vector'] = line['_source']['cvss']['vector']
```

Listing 4.4: Python Class Snippet to Wrap Vulners Webservice

Nmap Scripting Engine (NSE) uses an embedded Lua interpreter to enable the extension of Nmap functions. Moreover, a library connects Lua scripts with the Nmap API, allowing the parallel call of available functions. It is so consolidated in the Nmap environment that most users think the *-script vulns* option is a native function from it. However, a Lua script stored in the *script* folder under the NMAP root directory evaluates the search for vulnerabilities. Listing 4.5 is a code snippet of that Lua program. It inspired this work for building the Python class presented in Listing 4.4. It wraps the call to the Vulners web service to return more CVSS attributes than the Nmap Lua Script.

```

function get_results(what, vers, type)
  local api_endpoint = "https://vulners.com/api/v3/burp/software/"
  local vulns
  local option={
    header={
      ['User-Agent'] = string.format('Vulners NMAP Plugin %s', api_version)
    },
    any_af = true,
  }

  local response = http.get_url(('s?software=%s&version=%s&type=%s'):
format(api_endpoint, what, vers, type), option)

  local status = response.status
  if status == nil then
    -- Something went really wrong out there
    -- According to the NSE way we will die silently rather
    -- than spam user with error messages
    return
  elseif status ~= 200 then
    -- Again just die silently
    return
  end

  status, vulns = json.parse(response.body)

  if status == true then
    if vulns.result == "OK" then
      return make_links(vulns)
    end
  end
end
}

```

Listing 4.5: Nmap vulns script

## 4.5 RECOMMENDER

This work approach chooses the riskiest vulnerability using a decision-making method. In this case, it uses the TOPSIS algorithm based on the optimization technique described in [102, 103, 104]. The attack is a one-layer problem of picking up the riskiest vulnerability, i.e., the lowest cost and highest impact. So, the decision matrix has  $m$  vulnerabilities (rows) compared to  $n$  features (columns).

$$\mathbf{A} = (a_{ij}) \quad i = 1, 2, \dots, m; j = 1, 2, \dots, n \quad (4.1)$$

Equation 4.1 shows that the  $a_{ij}$  is the  $j$ th feature value of the  $i$ th vulnerability. After that, the  $\mathbf{A}$  is normalized to  $\mathbf{Y}$  and weighted by columns to its  $\mathbf{X}$  form:

$$\mathbf{Y} = (y_{ij}) \quad y_{ij} = a_{ij} / \sqrt{\sum_{i=1}^m a_{ij}^2} \quad (4.2)$$



$$\mathbf{X} = (x_{ij}) \quad x_{ij} = \lambda_j * y_{ij} \quad (4.3)$$

The technique requires choosing the best and worst options from the attacker role.

$$\mathbf{Z}^+ = (z_1^+, z_2^+, z_3^+, \dots, z_n^+) \quad (4.4)$$

$$\mathbf{Z}^- = (z_1^-, z_2^-, z_3^-, \dots, z_n^-) \quad (4.5)$$

Understanding if the column feature is a benefit or a cost is essential. If it is a benefit, the  $z_j$  value for the best option is the maximum column value. Otherwise, it must be the minimum value. Equations 4.6 e 4.7 indicate the calculus for benefit and cost features, respectively.

$$z_j^+ = \max(x_{ij}) \quad z_j^- = \min(x_{ij}) \quad (4.6)$$

$$z_j^+ = \min(x_{ij}) \quad z_j^- = \max(x_{ij}) \quad (4.7)$$

There are  $m$  vectors with dimension  $n$ , which are the rows of the matrix  $\mathbf{X}$ . The ideal solutions are two new vectors with sizes  $n$ ,  $Z^+$ , and  $Z^-$ , which TOPSIS uses to calculate the Euclidean distance with each of the  $m$  vectors.

$$d_i^+ = \sqrt{(Z_i - X_+)^2} \quad (4.8)$$

$$d_i^- = \sqrt{(Z_i - X_-)^2} \quad (4.9)$$

Finally, it calculates the performance ratio to rank each one of the vulnerabilities, from highest to lowest.

$$p_i = d_i^+ / (d_i^- + d_i^+), \quad i = 1, 2, 3, \dots, m \quad (4.10)$$

This work uses nine features as criteria in the decision matrix:

- **Seven Vulnerability Features:** described in Subsection 2.5 — access\_vector, access\_complexity, authentication, confidentiality\_impact, integrity\_impact, availability\_impact —, and the time gap between the vulnerability publishing date and the evaluation date. This last is a cost feature that TOPSIS must minimize;
- **Two Exploit Features:** the number of correspondent exploits for each vulnerability and how many verified exploits for each vulnerability.

The Multi-criteria Decision Analysis (MCDA) proposed in TOPSIS allows the ranking and assessing of

alternatives guided by criteria that may be conflicting. The weights that are used by each criterion have a strong influence over the MCDA results. Thus, besides the method itself, the weighting calculation algorithm influences the ranking procedure. There are three types of weighting methods: subjective, objective, and mixed. The first relies on expert opinions, which are subjected to human interference and error. The second depends solely on the algorithm and data themselves. The third one is a combination of the previous two. This work uses an objective method based on Entropy due to the lack of consensus from experts about the level of importance of each CVSS attribute over the other.

The weights vector  $\mathbf{W}$  ( $\lambda_j$ ) can be calculated from the normalized matrix  $\mathbf{Y}$  ( $y_{ij}$ ), using the following equations:

$$\mathbf{Z} = (z_{ij}) \quad z_{ij} = y_{ij} / \sum_{i=1}^m y_{ij} \quad (4.11)$$

$$\mathbf{E} = (e_j) \quad e_j = -(1/\ln m) \sum_{i=1}^m z_{ij} \ln z_{ij} \quad (4.12)$$

$$\mathbf{W} = (\lambda_j) \quad \lambda_j = (1 - e_j) / (n - \sum_{j=1}^n e_j) \quad (4.13)$$

The entropy value  $e_j$  from equation 4.12 is in the interval  $[0, 1]$ . Moreover, if  $z_{ij} = 0$  then  $z_{ij} \ln z_{ij} = 0$ . A smaller entropy value indicates a greater dispersion of the feature values, embedding more information and resulting in a greater weight value. On the other side, a greater entropy value shows less information, resulting in a decrease in the weight value. If the criterion values  $z_j$  are equal, the entropy value  $e_j$  will be one, indicating no information and resulting in a null weight value.

## 4.6 SUMMARY

This chapter describes the new Data Collection and Processing Pipeline to produce cyber intelligence and awareness. It generated a new vulnerability database merging the NVD, CNVD, and CNNVD. Furthermore, it also handles an exploit database using MSF and EDB, with the latter containing the former. The pipeline includes a layer to compare the vulnerability descriptions between the USA and China national databases using multilingual pre-trained models, being able to check the similarity between Chinese and English texts. Moreover, there is also a layer to rank scanned vulnerabilities using an entropy-based multi-criteria decision method with TOPSIS algorithm.

# 5 EXPERIMENTAL RESULTS

This chapter presents the experimental results of this work. Section 5.1 describes the software and hardware in the test environment. Section 5.2 shows the performance results for the vulnerability database building step and comparison results between the NVD, CNVD, and CNNVD. Section 5.3 explains the results regarding the EDB, which includes the MSF, and its relation with those vulnerability databases. Section 5.4 shows the Scanner results. Section 5.5 explains the Matcher results leveraging Nmap and the created databases. Section 5.6 analyses the Recommender results using TOPSIS algorithm. Section 5.7 concludes this chapter.

## 5.1 TEST ENVIRONMENT

This work emulates a medium enterprise Security Operation Center (SOC) using Graphical Network Simulator-3 (GNS3). It uses Quick Emulator (QEMU) on kernel-based virtual machines (KVM) in Linux Ubuntu 20.04. Furthermore, KVM in Linux performs better than type 2 hypervisors like VirtualBox and VMware because of hardware acceleration and kernel-embedded commands. Figure 5.1 presents the test environment.

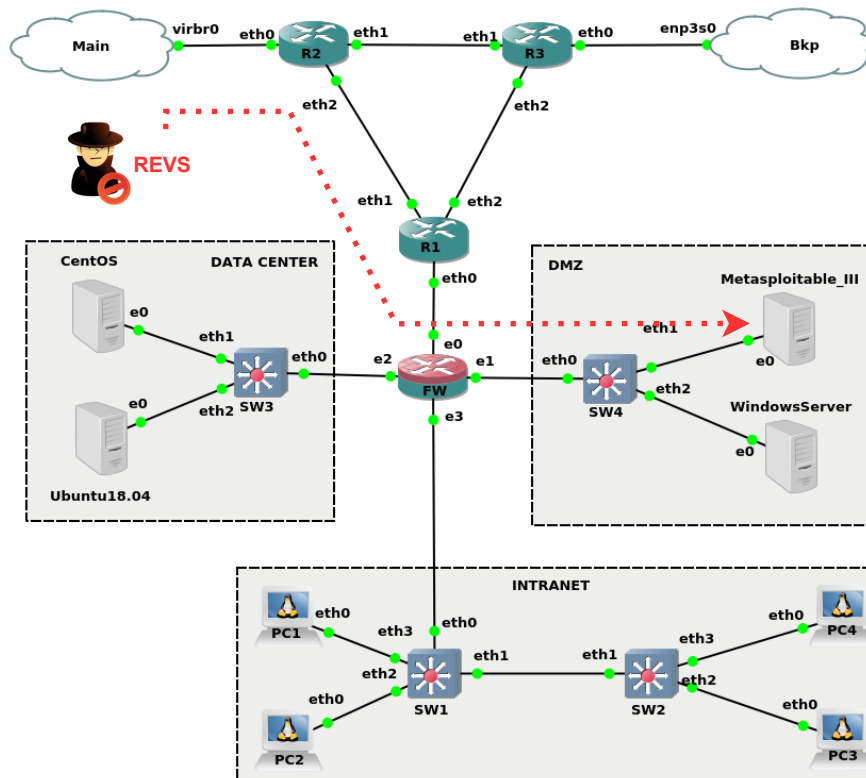


Figure 5.1: Emulated SOC with QEMU/KVN.

The GNS3 makes it possible to build an environment with different Operating Systems (OSs): Kali Linux, Ubuntu 18.04, CentOS 8, Windows Server, and Metasploitable III [105] machine. This last is a vulnerable public VM based on Ubuntu 14.04. Table 5.1 lists the environment hardware and VMs.

Table 5.1: Environment hardware and VMs

<i>Machine</i>	<i>Description</i>
Host	Ryzen 7 4800h / 16gb RAM
Emulator	GNS3 2.2.28
Metasploitable III	Ubuntu 14.04
Firewall	pfSense 2.5.2
Routers	VyOS 1.1.8
Switches	Open vSwitch 2.4.0

For the data scraping and processing, relational database creation, and NLP of the vulnerability description text in the NVD and CNNVD, this work uses two computers running OS Ubuntu 20.04, each with 16GB RAM. The NLP task requires more computer processing, which this work achieves using CPU processing for the sampled texts and GPU processing for the entire dataset, as described in subsection 4.1.4. Table 5.2 lists each hardware item and its respective purpose.

Table 5.2: Data and text processing hardware

<i>Item</i>	<i>Function</i>
Ryzen 7 4800h	Data scraping and database
NVIDIA GeForce GTX 1650 Ti	NLP
Intel(R) Core(TM) i5-10310U	NLP

## 5.2 VULNERABILITY DATABASE RESULTS

This work implements three data scrapers to collect vulnerability data from the NVD, CNVD, and CNNVD. The first uses the JSON data feed supported by NIST, which provides vulnerability registers with unique CVE id and CVSS v2 and v3 scores. Besides, some of these registers also contain CPEs and CWEs related to the vulnerability. This work uses the OPENCVE tool [94] to retrieve data from the REST API and store it locally in a PostgreSQL relational database.

The other two scrapers require implementation from scratch. The CNVD does not provide any API to return vulnerability data. Otherwise, it provides a set of XML files lacking schema definition. Moreover, CNCERT/CC generates a new XML file every Monday at 18h:00 (CST) with vulnerability registers from the past week. Different from the work of [12], REVS used the python requests library with custom

“User-Agent” tag and cookie parameters (`__jsluid` and `__jsl_clearance_s`) to bypass the CNVD blocking system.

Figure 5.2 displays a screenshot of the Linux terminal, showing that the CNVD provides one file each week. Besides, this work revealed that the CNVD XML data only have vulnerability records since 2015, despite the CNVD Web site showing registers before that date.

```

2020-09-28_2020-10-04.xml 2021-10-04_2021-10-10.xml
2020-10-05_2020-10-11.xml 2021-10-11_2021-10-17.xml
2020-10-19_2020-10-25.xml 2021-10-18_2021-10-24.xml
2020-10-26_2020-11-01.xml 2021-10-25_2021-10-31.xml
2020-11-02_2020-11-08.xml 2021-11-01_2021-11-07.xml
2020-11-09_2020-11-15.xml 2021-11-08_2021-11-14.xml
2020-11-16_2020-11-22.xml 2021-11-15_2021-11-21.xml
2020-11-23_2020-11-29.xml 2021-11-22_2021-11-28.xml
2020-11-30_2020-12-06.xml 2021-11-29_2021-12-05.xml
2020-12-07_2020-12-13.xml 2021-12-06_2021-12-12.xml
2020-12-14_2020-12-20.xml 2021-12-13_2021-12-19.xml
2020-12-21_2020-12-27.xml 2021-12-20_2021-12-26.xml
2020-12-28_2021-01-03.xml 2021-12-27_2022-01-02.xml
2021-01-04_2021-01-10.xml 2022-01-03_2022-01-09.xml
2021-01-11_2021-01-17.xml 2022-01-10_2022-01-16.xml
2021-01-18_2021-01-24.xml 2022-01-17_2022-01-23.xml

```

Figure 5.2: CNVD Files Sample.

Afer downloading the XML files from the CNVD dataset, this work detect six missing files (weeks): *2019-02-11\_2019-02-17.xml*, *2019-05-20\_2019-05-26.xml*, *2019-09-23\_2019-09-29.xml*, *2020-03-02\_2020-03-08.xml*, *2020-10-12\_2020-10-18.xml* and *2015-02-16\_2015-02-22.xml*.

This section discusses the performance results during the Structuring step. Lastly, it explains the features and differences between the national vulnerability databases.

### 5.2.1 Performance Results

The performance results of the Table 5.3 show the execution time to transform the unstructured XML and JSON into structured data into PostgreSQL.

Table 5.3: Batch Processing

<i>Database</i>	<i>Format</i>	<i>Files</i>	<i>Total Size</i>	<i>Process Time</i>
NVD	JSON	23	1.5 GB	266.74s
CNVD	XML	364	146.9 MB	11.33s
CNNVD	XML	25	2.4 GB	48.88s

The Data Structuring step uses a pool of five workers. They can process in parallel five blocks of data leveraging multicore processors. This approach assigns each data file to a worker for data normalization. Although there is more data in the CNNVD than in NVD, this takes more processing time than that because of nested JSON data.

## 5.2.2 Comparison Results

Table 5.4 shows an outline of the three national vulnerability databases.

Table 5.4: Data Summary

<i>Feature</i>	<i>NVD</i>	<i>CNVD</i>	<i>CNNVD</i>
Vulnerability	178,906	99,261	180,567
Missing Weeks	0	6	0
Missing CVE	0	23,281	9,963
Repeated ID	0	88	0
Repeated CVE	0	108	15
Wrong CVE	0	193	0
Missing CVSS	10,933	326	8,466
Missing CPE	11,143	113	25,694
Missing CWE	10,928	99,261	180,567
Missing Reference	24,035	19,012	5,786

Table 5.4 shows that the CNNVD has more entries than the NVD, providing a more comprehensive dataset regarding system vulnerabilities. The CNNVD also enables mapping to the NVD using the CVE id to link the two databases. There are only 9,963 entries without CVE id mapping in the CNNVD. Despite fewer entries than the CNNVD, the CNVD has many more vulnerabilities without CVE mapping. Other than that, the previous files from the CNVD are not updated.

At first sight, the CVSS metric seems to be more available in the Chinese databases than in the NVD. Nevertheless, after profiling the 10,933 entries without CVSS score in the NVD, 10,531 has a “REJECT” status. CNAs required those 10,531 CVE ids without assigning any vulnerability. Only 721 vulnerabilities of these 10,531 are stored in the CNNVD, showing that this Chinese database evaluates filtering mechanisms.

The CVSS in the CNVD and CNNVD is a one-column data to the severity metric representing the CVSS base score. There is no score calculation but the equivalent text description. Moreover, the CNVD uses CVSS version 2 (low, medium, and high) and the CNNVD uses versions 2 and 3 (low, medium, high and critical), trying to follow the FIRST standard. The CNVD and CNNVD neither provide the CVSS base and temporal attributes nor the CVSS vector string. Section 2.5 presented the background of the CVSS scoring system.

Figure 5.3 shows that 23.45% of the vulnerability entries in the CNVD have no CVE feature, preventing a link to the NVD. Despite much more entries, the CNNVD has 5.52% without CVE. The NVD has no CVE issues because the CVE code is its primary key. Despite the main responsible for CVE Mitre framework implementation, the NVD has 6.11% of the entries without any CVSS metric, while the CNNVD has 4.69%. Otherwise, the CNNVD has 14.23% of the entries without CPE, preventing a comprehensive identification of vulnerable systems in those cases. On the other hand, the CNNVD carries more information regarding

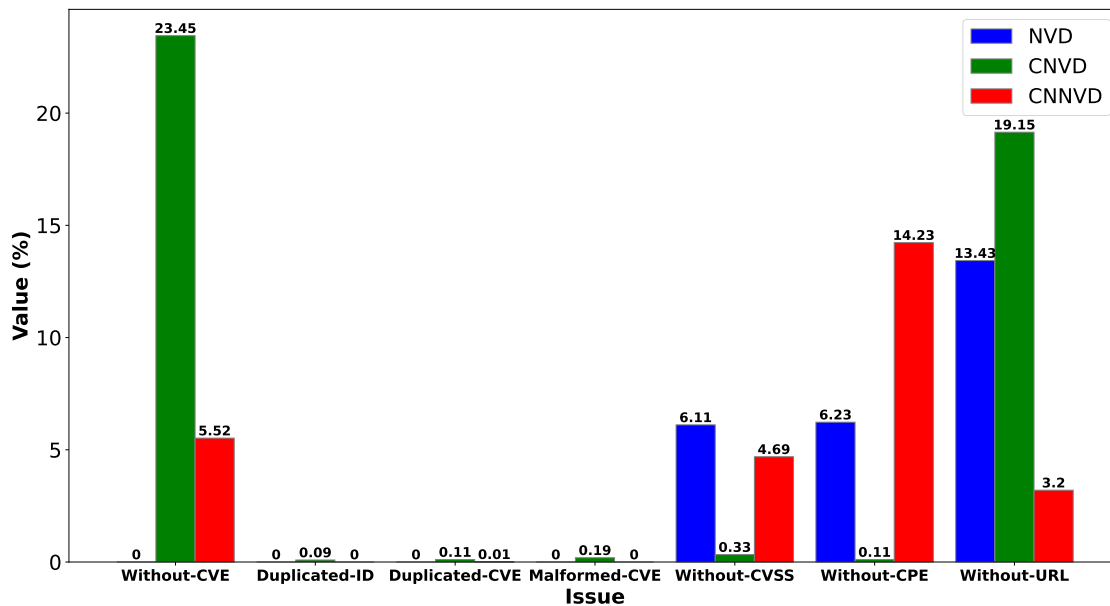


Figure 5.3: Issues Summary.

external URL references, with only 3.2% of the entries without this information.

Also, it is worth mentioning that the Chinese databases do not present any CWE information, while the NVD has 6.11% of the entries without this information. The CWE, critical information to group the hardware and software vulnerabilities, is not embedded in the Chinese databases. Nevertheless, they offer a text defining the type of vulnerability that looks like the CWEs description from the MITRE framework.

The CPE information of the NVD is more comprehensive than that of the CNNVD and CNVD. The earlier covers three configurations: Basic, Running On/With, and Advanced. The Chinese databases do not include the CPE information with this granularity. The CNNVD lists every software version affected by the vulnerability, while the NVD describes logical conditions covering the possibilities of vulnerability occurrence. Furthermore, the CNNVD provides data about the possible solutions with external URLs. The NVD does not offer this solution information in a separate column.

### 5.2.3 Comparing the NVD and CNNVD

After realizing that the CNVD XML files have vulnerability records only after 2015, this work decided to keep assessing the NVD and CNNVD. This work evaluated three tests with them: seek an explanation for unmatched CVEs, compare Chinese vendors in that database with the NVD, and temporal analysis. For the first issue, a complete translation of the English language is a necessary step that this work did not evaluate. Despite that, this work found 25 vulnerabilities entries of the type “information leakage”, which are not available to the public. They have been published since 2019 and received an id, but they are classified information. Figure 5.4 shows vulnerability entries for three famous Chinese vendors in the CNNVD and NVD: Huawei, ZTE, and Xiaomi.

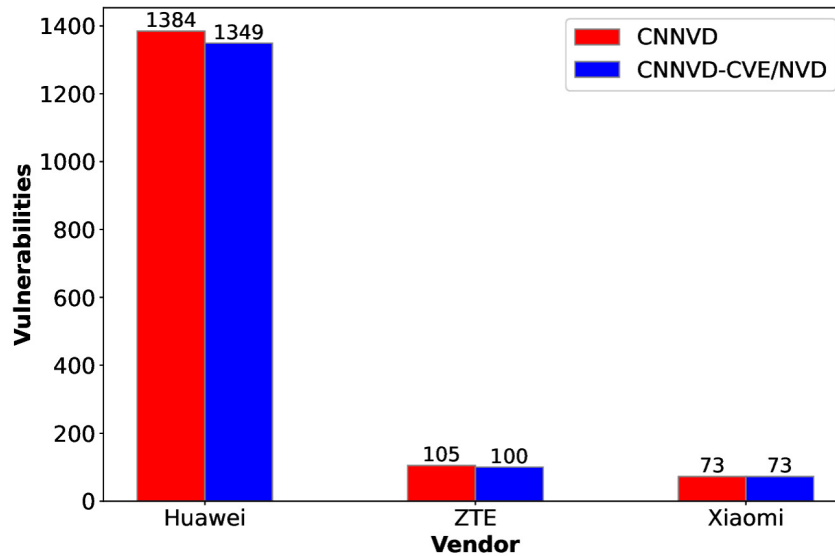


Figure 5.4: Vendors Comparison.

Figure 5.4 indicates that the CNNVD has more vulnerability entries regarding Huawei than the NVD, while there is no difference for Xiaomi. By the way, the number of Figure 5.4 comes from counting the vulnerabilities with the string “Huawei” in the description. There may be more entries from these vendors in mandarin. Lastly, Figure 5.5 shows the monthly time series for the NVD and CNNVD.

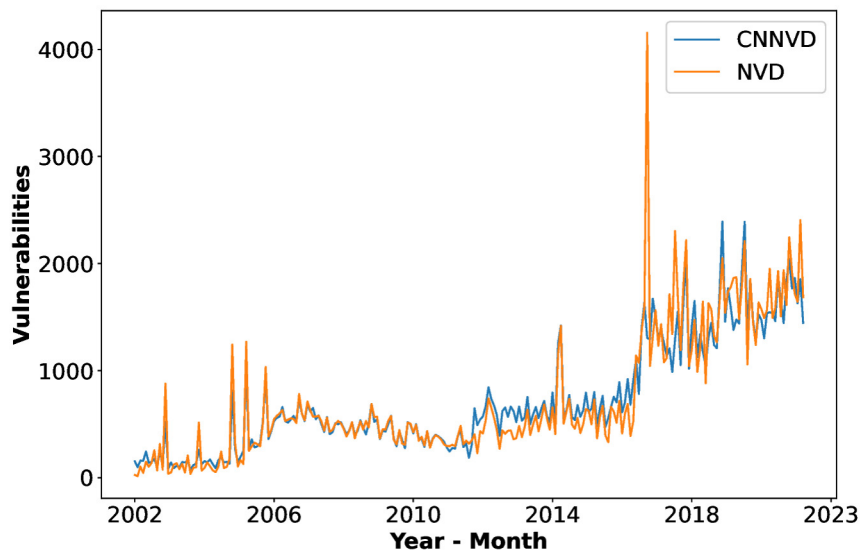


Figure 5.5: Monthly Time Series.

Figure 5.5 reveals that the time series are very similar, with a Pearson correlation of 0.917560. It indicates that they may have been using the same information sources, or maybe they are tracking each other.



## 5.2.4 Multilingual NLP

Despite showing 170,604 registers with mapped CVE — that is the reason for 9,963 missing CVE entries in Table 5.4 — 168,398 of them point to CVEs that both the NVD and CNNVD have. Thus, that set of 168,398 vulnerabilities is the sample space to support the choice of the best performance multilingual transfer model with 95% of confidence. So, this work randomly selects 384 registers from that population with the test scheme proposed in subsection 4.1.4 to assess those five models regarding execution times and text similarity results. Figures 5.6 and 5.7 present the histograms for the NVD and CNNVD, respectively, embedding the values for the simple average, 50th, 95th, and 99th quantiles.

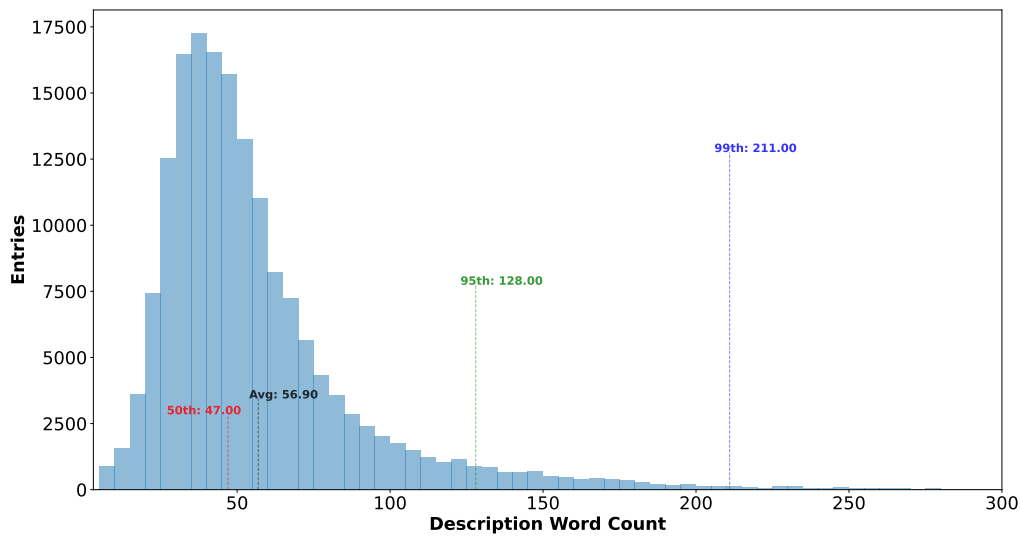


Figure 5.6: Word Count Histogram for the NVD.

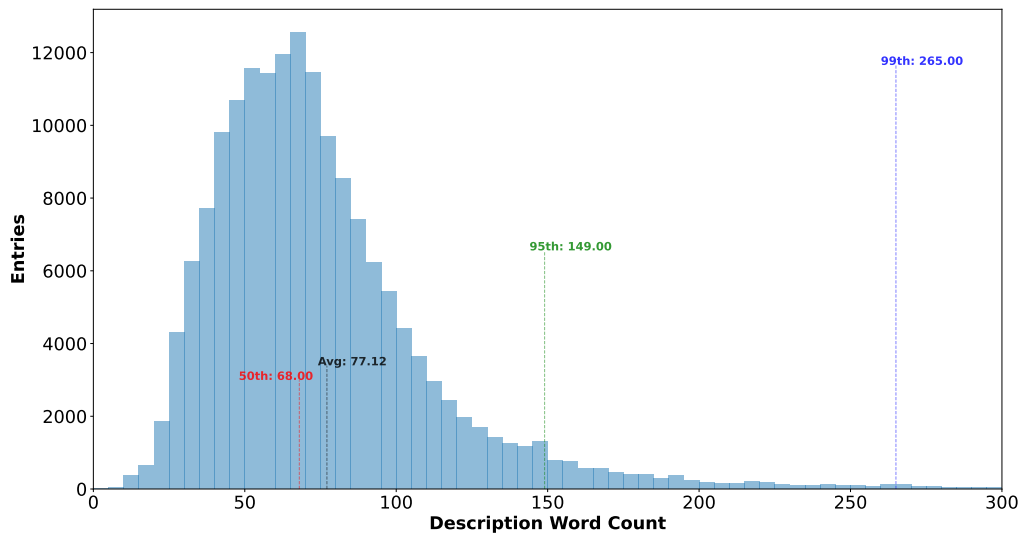


Figure 5.7: Word Count Histogram for the CNNVD.

Figures 5.6 and 5.7 show that the CNNVD has simple average, 50th, 95th and 99th quantiles about 25% greater than the NVD. However, both have more than 99% of vulnerability text descriptions with less than 300 words. Therefore, this work uses 300 as the *max\_word\_length* parameter for those five pre-trained models, avoiding text truncation for most vulnerability descriptions.

After setting 300 as the *max\_word\_length* parameter, this work runs each model against the same sampled set with 384 vulnerabilities. Because of the small sample set, this work uses a CPU parallel processing approach with PyTorch to evaluate and compare those five pre-trained models regarding the processing time and similarity calculation performances. Since the CORE i5 vPRO 10th generation CPU has four physical cores, this work uses three parallel processes for model processing, leaving one core free for the OS. Therefore, this procedure is repeated five times since this work tackles five models, enabling a fair comparison of the processing time. Table 5.5 shows how much time takes for each model to process the sampled set.

Table 5.5: Models Processing Performance

<i>Model</i>	<i>Processing Time</i>
distiluse-base-multilingual-cased-v1 (distiluseV1) [98, 99]	159.30s
distiluse-base-multilingual-cased-v2 (distiluseV2) [98, 99]	171.75s
paraphrase-multilingual-MiniLM-L12-v2 (MiniLM) [98, 99]	97.87s
LaBSE (LaBSE) [100]	328.31s
paraphrase-multilingual-mpnet-base-v2 (mpnet) [98, 99]	290.71s

Table 5.5 shows that the two versions of the distiluse model took almost the same time to process the dataset, about 120s. The MiniLM is the fastest, taking 74.60s, while the LaBSE lasted 322.25s, showing it is the slowest. After that, this work compares the processing results regarding the similarity between the Chinese text description of those 384 vulnerabilities with their translated versions by the Google Translator Service(<<https://translate.google.com/>>) and DeepL Translator (<<https://www.deepl.com/translator>>). Figure 5.8 shows the achieved results using the simple average of the text cosine similarities.

Figure 5.8 shows that the LaBSE model reaches higher cosine similarity (closer to 100%) for the translated versions — 95% with the Google and 94.83% with DeepL — and also a higher value with the NVD, 72.94%. Despite not being the objective of this experiment, all models indicate that the Google-translated text is slightly more similar to the Chinese one than that translated by DeepL. It is worth mentioning that the MiniLM achieved results close to the LabBSE and mpnet, despite being 3,35x faster than the first and 2,97x faster than the second. It indicates that the MiniLM is more suited when the time processing constraints are more important than the similarity requirements. This work also evaluated the average of the similarities weighted by the text size to check if it influences the results. Figure 5.9 shows the same bar graph style of the Figure 5.8, but now using the weighted cosine similarities.

Figures 5.8 and 5.9 display that the following three models achieved better comparison performance in crescent order: MiniLM, Mpnet and Labse. Therefore, this work runs those three models to compare the

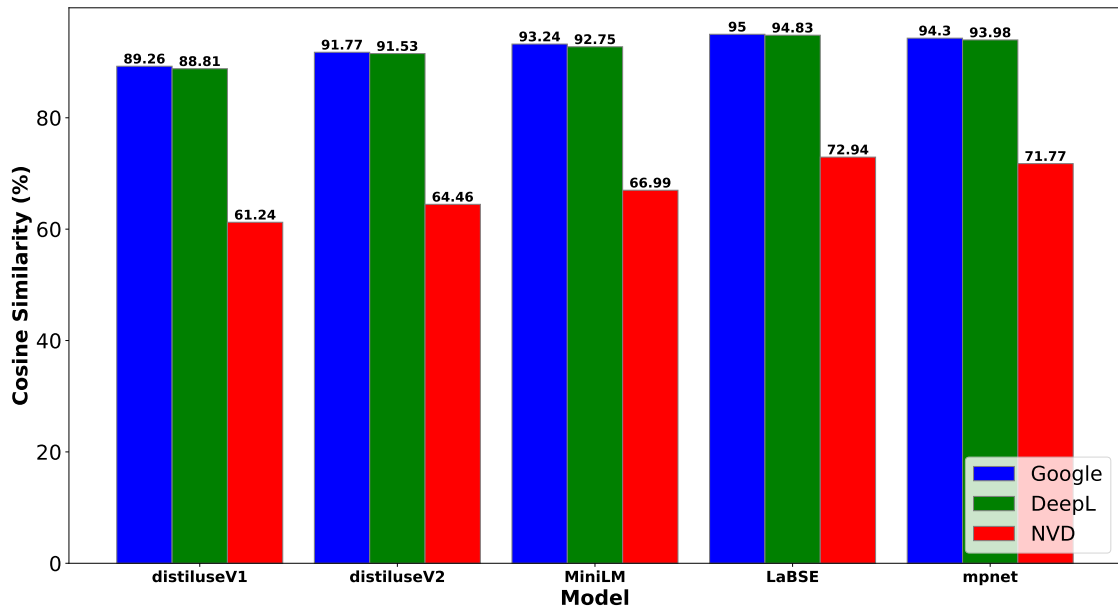


Figure 5.8: Simple Average Text Similarity

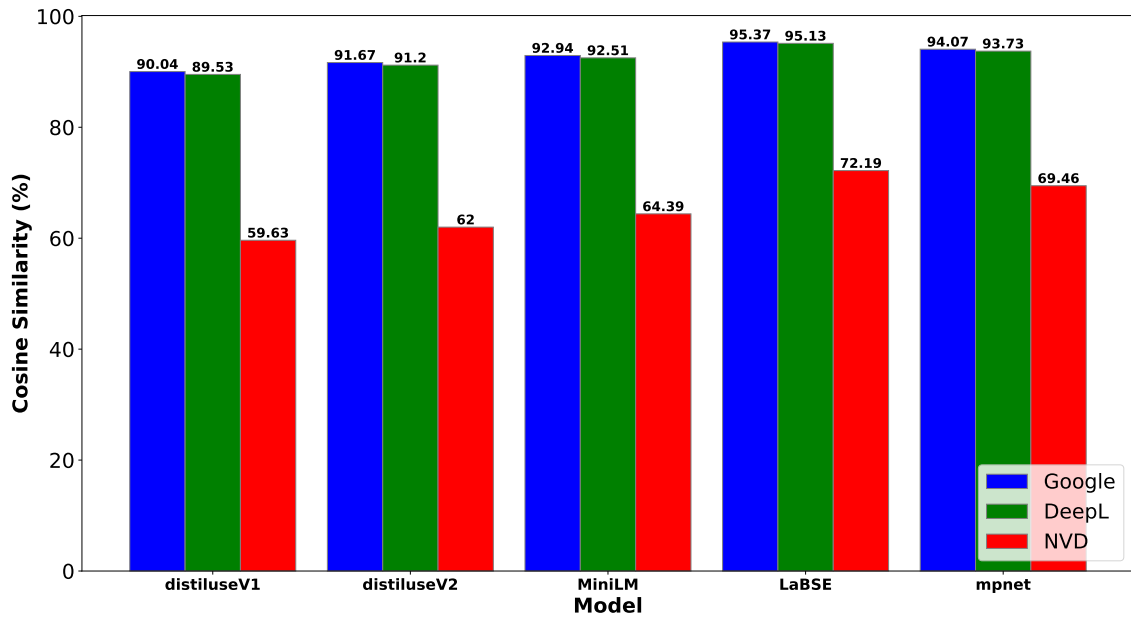


Figure 5.9: Weighted Average Text Similarity

description text of the 168,398 vulnerability entries — the entire dataset — that exist both in the CNNVD and NVD. Table 5.5 showed that the fastest model, MiniLM, took 74.06s to process 384 registers in a parallel processing approach leveraging the three cores of the CPU. Therefore, the same approach will take at least 9 hours for the entire dataset. So, this work uses a parallel processing approach leveraging a

GPU with CUDA to speed up the text processing. Table 5.6 lists the processing time for those 168,398 vulnerabilities.

Table 5.6: Models Processing Performance with GPU

<i>Model</i>	<i>Processing Time</i>
paraphrase-multilingual-MiniLM-L12-v2 (MiniLM) [98, 99]	1031.62s
LaBSE (LaBSE) [100]	4818.29s
paraphrase-multilingual-mpnet-base-v2 (mpnet) [98, 99]	3675.72s

Like the CPU approach with the sampled data, this work checks the cosine similarity between the Chinese text in the CNNVD with the correspondent description in the English language in the NVD, but now for the entire dataset using the three models about which Table 5.6 describes the time performance. Figure 5.10 shows the achieved results using the simple average of the text cosine similarities.

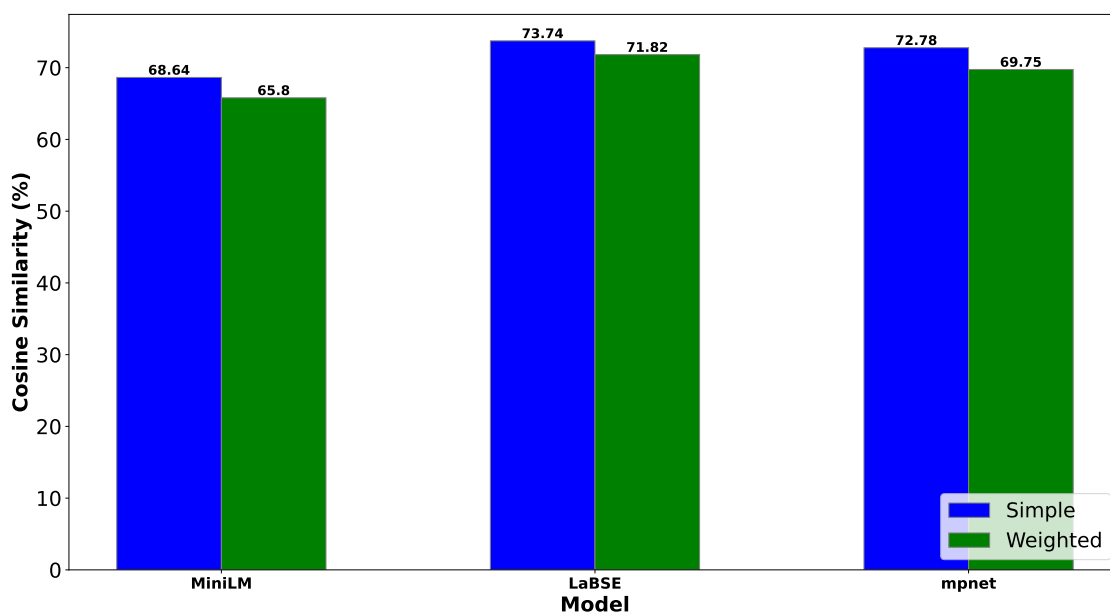


Figure 5.10: Text Similarity between the NVD and CNNVD

Figure 5.10 displays that the simple and weighted averages achieve close results. Besides, the results with the entire dataset are equivalent to those of the sampled vulnerabilities, indicating that the sampling approach to choosing the three best pre-trained models was correct. Moreover, Figure 5.10 reveals the LaBSE achieved the highest similarities with the simple and weighted averages, respectively: 73.74% and 71.82%.

## 5.3 EXPLOIT DATABASE RESULTS

This work implements a crawler to collect exploit data from the EDB and a program to parse the MSF modules. The first uses sequential calls to the EDB web page with HTML and Javascript scraping to capture the exploit source codes and their metadata. As described in section 4.2, this work did not use a multithreading approach to avoid blocking mechanisms. Because of that, it has taken almost five days to download the EDB data.

This section discusses the EDB records, comparing them with the exploited vulnerabilities in the NVD, NVD, and CNNVD. The CVE ID is the key to linking those databases and grasping if the Chinese ones are ahead of the NVD or leveraging the EDB for vulnerability inference.

### 5.3.1 EDB Analysis

During the EDB parsing, this work detected an inconsistency between the rendered Web Page and its internal metadata. The Browser may display exploited CVE IDs that differ slightly from those listed in the internal HTML code. Figures 5.11 and 5.12 show an example of this issue with the exploit id 900.

```
<link rel="canonical" href="https://www.exploit-db.com/exploits/900">
<meta name="description"
  content="Smail 3.2.0.120 - Remote Heap Overflow. CVE-15065CVE-2005-0892 . remote exploit for Linux platform">
<meta name="keywords" content="Linux, remote, CVE-15065CVE-2005-0892 ">
<meta name="author" content="infamous41md">

<meta property="og:title" content="Smail 3.2.0.120 - Remote Heap Overflow">
<meta property="og:type" content="article" />
<meta property="og:url" content="https://www.exploit-db.com/exploits/900">
<meta property="og:image" content="https://www.exploit-db.com/images/spider-orange.png" />
<meta property="og:site_name" content="Exploit Database" />
<meta property="article:published_time" content="2005-03-28" />
  <meta property="article:author" content="infamous41md" />
<meta property="article:authorUrl" content="https://www.exploit-db.com/?author=226" />
```

Figure 5.11: EDB Inconsistency

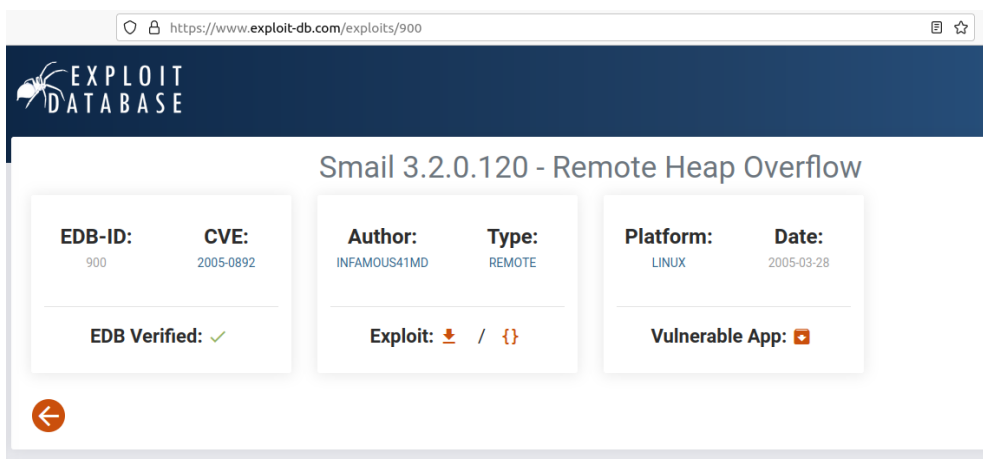


Figure 5.12: Rendered Web Page from EDB Inconsistency.

While the keywords metadata in Figure 5.11 lists the CVE-15065 and CVE-2005-0892, the rendered Web Page in Figure 5.12 shows only the CVE-2005-0892. The CVE-15065 is malformed since there is no mention of the year part in the code. Despite seeming a minor issue, if someone uses the search page to look

for exploits to the wrong CVE-15065, it will return the results listed in Figure 5.13.

### Exploit Database Advanced Search

**Title**

**CVE**

**Content**

Verified
  Has App
  No Metasploit

---

Show 15

Date	D	A	V	Title
2005-03-28	↓	☑	✓	Smail 3.2.0.120 - Remote Heap Overflow
2005-03-25	↓	☑	✓	Smail 3 - Multiple Remote/Local Vulnerabilities

Showing 1 to 2 of 2 entries

Figure 5.13: EDB Wrong Search

Figure 5.13 shows that the search mechanism of the EDB does not check the CVE ID in the NVD. After detecting that issue, this work evaluated a cross-reference between the EDB and NVD using the CVE ID. In the first round, this work found 12 entries in the EDB with at least one of the following errors: escape characters, wrong dash separators, and malformed IDs. After correcting them, this work evaluated a second cross-reference and identified 35 CVEs in the EDB that did not exist in the NVD. Table 5.7 categorizes those missing CVEs.

Table 5.7: CVEs in the EDB without being in the NVD

<i><b>CVEs</b></i>	<i><b>Observation</b></i>
CVE-2021-44673	The NVD published it on 03/10/2022.
CVE-2021-3560	The NVD published it on 02/16/2022.
CVE-2021-35380	The NVD published it on 02/15/2022.
CVE-2019-89242	It neither exists in Mitre CVE nor in the NVD.
CVE-2019-5797, CVE-2019-17591, CVE-2019-13491, CVE-2017-2796, CVE-2017-12854, CVE-2017-11197, CVE-2016-2534, CVE-2014-5470, CVE-2014-5469, CVE-2014-5329, CVE-2014-3736, CVE-2014-3212, CVE-2014-2923, CVE-2014-2239, CVE-2013-3307, CVE-2013-2649, CVE-2013-1916, CVE-2013-1891, CVE-2012-6679, CVE-2012-6664, CVE-2012-4749, CVE-2012-4748, CVE-2012-2765, CVE-2012-1309, CVE-2012-1305, CVE-2012-1304, CVE-2012-1300, CVE-2011-1656, CVE-2010-2782, CVE-2010-0368, CVE-2005-0153	They exist in Mitre CVE with “RESERVED” status, but are not listed in the NVD.
<b>Total</b>	<b>35</b>

Table 5.7 shows that despite the Mitre CVE and NVD being under the same authority – the DHS

as explained in 2.2 – the supposed online synchronization between them is not happening. It may be a filtering mechanism to shred the vulnerability entries with the “RESERVED” status from the NVD. However, subsection 5.2.2 revealed that the NVD does not remove entries with “REJECT” status. It is dangerous since a “RESERVED” status, unlike the “REJECT” one, might indicate an existing vulnerability that has not completed the registration flow. Even if there is no existence or description in the NVD, the attacker can infer the vulnerability characteristics from the exploit in the EDB.

For a fair comparison between the national databases and EDB, this work only considers entries in them before 01/25/2022. Thus, Table 5.7 also reveals that the EDB is ahead of the NVD in some cases. It mentioned the CVEs 2021-44673, 2021-3560, and 2021-35380 before the NVD. Also, the nonexistent CVE-2019-89242 listed in Table 5.7 and the typos indicate manual registration and the absence of checking procedures in the EDB.

After finding those 35 records cataloged in Table 5.7, this work searched for them in the CNVD and CNNVD. The results corroborate the differences between the vulnerability databases presented in subsection 5.2.2. Considering those 35 CVEs that the NVD does not have, 15 are in CNNVD, and two are in the CNVD. Of those two, one is listed only in the CNVD. Table 5.8 describes those findings.

Table 5.8: CVEs mentioned in EDB and listed only in CNVD or CNNVD

<i>CVE</i>	<i>Chinese ID</i>	<i>Chinese Title</i>	<i>In CNVD</i>	<i>In CNNVD</i>
CVE-2011-1656	CNNVD-201107-051	NetBSD /libc/net/栈缓冲区错误漏洞	✗	✓
CVE-2014-3736	CNNVD-201406-575	ALLPlayer'.wav'文件处理内存损坏漏洞	✗	✓
CVE-2014-5329	CNNVD-201410-1274	多款TripodWorks GIGAPOD产品远程拒绝服务漏洞	✗	✓
CVE-2014-3212	CNNVD-201406-559	KMPlayer 远程基于栈的缓冲区错误漏洞	✗	✓
CVE-2014-2239	CNNVD-201501-100	Lazarus Guestbook SQL注入漏洞和HTML注入漏洞	✗	✓
CVE-2017-12854	CNNVD-201708-629	✗	✗	✓
CVE-2017-11197	CNNVD-201707-514	✗	✗	✓
CVE-2013-1891	CNNVD-201303-399	OpenCart 'filemanager.php' 多个目录遍历漏洞	✗	✓
CVE-2013-2649	CNNVD-201304-225	Hero Framework 多个跨站脚本漏洞	✗	✓
CVE-2013-1916	CNNVD-201303-530	WordPress User Photo 'user-photo.php'任意文件上传漏洞	✗	✓
CVE-2019-5797	CNNVD-201903-464	Google Chrome 竞争条件漏洞	✗	✓
CVE-2012-1300	CNNVD-201203-466	phpFox 'ajax.php' 远程命令执行漏洞	✗	✓
CVE-2019-89242	CNNVD-202102-067	WordPress 安全漏洞	✗	✓
CVE-2021-3560	CNNVD-202106-211	polkit 权限许可和访问控制问题漏洞	✗	✓
CVE-2019-13491	CNVD-2020-35173	Tenda D301跨站脚本漏洞	✓	✓
	CNNVD-201907-713	Tenda D301 跨站脚本漏洞		
CVE-2016-2534	CNVD-2020-35173	Jive Forums 目录遍历漏洞	✓	✗
<b>Total: 16</b>			<b>2</b>	<b>15</b>

Table 5.8 reveals that the CNVD misses most of those CVEs. That is compatible with the findings regarding the CNVD XML files, as they have vulnerabilities later than 2015. However, it does not explain the absence of the CVEs 2019-5797, 2019-89242, and 2021-3560, as the NVD released them after 2015. Thus, this work decided to look for them in the CNVD Web Page displayed in Figure 5.14 considering a possible failure only in the XML feed and not in the Web interface.

After searching for the CVE-2019-5797 and CVE-2019-89242, the Web Page printed in Figure 5.14 did not return any results. It shows no inconsistency between the XML file and Web page regarding those two vulnerabilities. However, after looking for the CVE-2021-3560, it returned six entries as printed in

**高级搜索** ×

**漏洞基本信息**

关键字:   仅标题  标题和描述 与 ▾

CNVD编号:  (如:CNVD-2010-00001) 与 ▾

发布日期范围:   与 ▾

参考信息:   ▾

厂商首字母: **A B C D E F G H I J K L M N O P Q R S T U V W X Y Z**

厂商:  ▾

产品首字母: **A B C D E F G H I J K L M N O P Q R S T U V W X Y Z**

产品:  ▾

版本:  ▾

**可选属性条件**

漏洞产生原因	漏洞引发的威胁	漏洞严重程度	漏洞利用的攻击位置
<input type="checkbox"/> 其他错误 <input type="checkbox"/> 意外情况处理错误 <input type="checkbox"/> 未知错误 <input type="checkbox"/> 环境错误 <input type="checkbox"/> 竞争条件 <input type="checkbox"/> 设计错误 <input type="checkbox"/> 访问验证错误 <input type="checkbox"/> 输入验证错误 <input type="checkbox"/> 边界条件错误 <input type="checkbox"/> 配置错误	<input type="checkbox"/> 其它 <input type="checkbox"/> 拒绝服务 <input type="checkbox"/> 普通用户访问权限获取 <input type="checkbox"/> 未授权的信息修改 <input type="checkbox"/> 未授权的信息泄露 <input type="checkbox"/> 未知 <input type="checkbox"/> 管理员访问权限获取	<input type="checkbox"/> 中 <input type="checkbox"/> 低 <input type="checkbox"/> 高	<input type="checkbox"/> 其他 <input type="checkbox"/> 本地 <input type="checkbox"/> 远程

Figure 5.14: The CNVD Web Search.

Figure 5.15.

编号:  开始时间:

结束时间:

漏洞标题	危害级别	点击数	评论	关注	时间↓
› Oracle Java SE和Oracle GraalVM Enterpris...	<span style="display: inline-block; width: 15px; height: 10px; background-color: yellow;"></span> 中	51	0	0	2021-10-19
› Oracle PeopleSoft Enterprise PeopleTools...	<span style="display: inline-block; width: 15px; height: 10px; background-color: yellow;"></span> 中	56	0	0	2021-10-19
› Oracle MySQL Server拒绝服务漏洞 (CNVD-20...	<span style="display: inline-block; width: 15px; height: 10px; background-color: yellow;"></span> 中	54	0	0	2021-10-19
› Oracle MySQL Server存在未明漏洞 (CNVD-20...	<span style="display: inline-block; width: 15px; height: 10px; background-color: yellow;"></span> 中	59	0	0	2021-10-19
› Oracle MySQL Server拒绝服务漏洞 (CNVD-20...	<span style="display: inline-block; width: 15px; height: 10px; background-color: green;"></span> 低	50	0	0	2021-10-19
› Oracle MySQL Server存在未明漏洞 (CNVD-20...	<span style="display: inline-block; width: 15px; height: 10px; background-color: yellow;"></span> 中	54	0	0	2021-10-19

**共 6 条**

Figure 5.15: Search results for CVE-2021-3560 in the CNVD Web.

Thereby, this work profiled those six returned vulnerabilities from Figure 5.15 regarding the CVE-2021-3560 in the respective CNVD detail page. Awkwardly, none of them has any relation to the CVE-2021-3560.



However, those six results have CVEs in which the first thirteen characters match the string pattern “CVE-2021-3560”. It shows search issues on the CNVD Web Page.

This work repeated the same procedure for the CVE-2016-2534 but in the CNNVD. It searched for the CVE-2016-2534 on the CNNVD Web page because it was the only one not found in the XML files. It returned no results, as printed in Figure 5.16, indicating no inconsistency between the Web and file interfaces in the CNNVD.

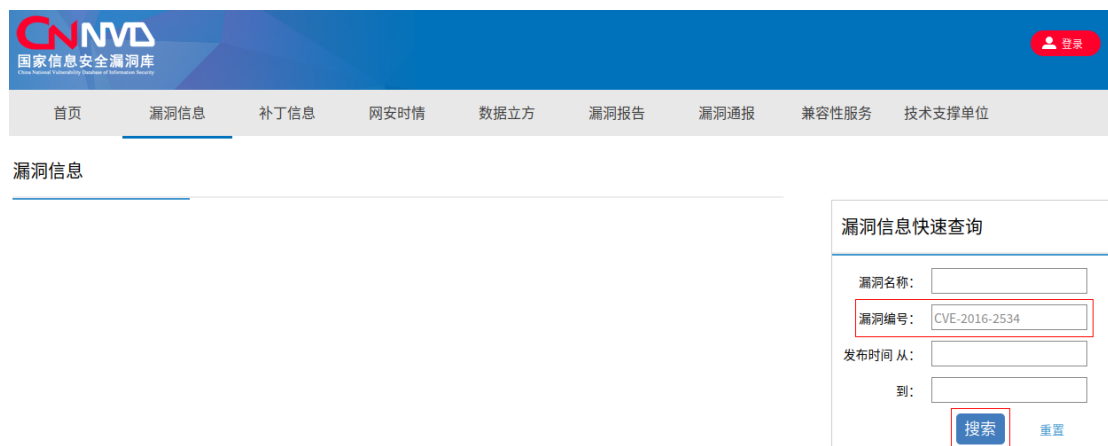


Figure 5.16: The CNNVD Web Search.

This work demonstrates through data presented in Tables 5.7 and 5.8 from the EDB, CNVD, and CNNVD that they are ahead of the NVD. However, the question of whether the Chinese databases are leveraging the EDB to achieve new vulnerabilities nonexistent in the NVD was still open. Table 5.9 lists those 16 vulnerabilities in the CNNVD, and their correspondent exploit titles to answer that question. It enables comparing them.

Table 5.9: CNNVD Vulnerabilities and Their Exploits in EDB

CVE	EDB ID	EDB Title	Chinese Title
CVE-2011-1656	35919	NetBSD 5.1 - 'libc/net' Multiple Stack Buffer Overflows	NetBSD /'libc/net/'栈缓冲区错误漏洞
CVE-2014-3736	39183	ALLPlayer - '.wav' File Processing Memory Corruption	ALLPlayer'.wav'文件处理内存损坏漏洞
CVE-2014-5329	17696 18221	Apache - Remote Memory Exhaustion (Denial of Service) Apache - Denial of Service	多款TripodWorks GIGAPOD产品远程拒绝服务漏洞
CVE-2014-3212	39181	Intel Indeo - Video Memory Corruption	KMPlayer 远程基于栈的缓冲区错误漏洞
CVE-2014-2239	35605	Lazarus Guestbook 1.22 - Multiple Vulnerabilities	Lazarus Guestbook SQL注入漏洞和HTML注入漏洞
CVE-2017-12854	44065	Sophos XG Firewall 16.05.4 MR-4 - Path Traversal.	✗
CVE-2017-11197	42319	CyberArk Viewfinity 5.5.10.95 - Local Privilege Escalation	✗
CVE-2013-1891	24877	OpenCart 1.5.5.1 - 'FileManager.php' Directory Traversal Arbitrary File Access	OpenCart 'filemanager.php' 多个目录遍历漏洞
CVE-2013-2649	38461 38462	Hero Framework - '/users/login?Username' Cross-Site Scripting Hero Framework - '/users/forgot_password?error' Cross-Site Scripting	Hero Framework 多个跨站脚本漏洞
CVE-2013-1916	16181	WordPress Plugin User Photo Component - Arbitrary File Upload	WordPress User Photo 'user-photo.php'任意文件上传漏洞
CVE-2019-5797	46565	Google Chrome < M73 - Double-Destruction Race in StoragePartitionService	Google Chrome 竞争条件漏洞
CVE-2012-1300	18655	PHPFox 3.0.1 - 'ajax.php' Remote Command Execution	phpFox 'ajax.php' 远程命令执行漏洞
CVE-2019-89242	49512	WordPress 5.0.0 - Image Remote Code Execution	WordPress 安全漏洞
CVE-2021-3560	50550 50011	Polkit Authentication bypass Local Privesc - Paper Polkit 0.105-26 0.117-2 - Local Privilege Escalation	polkit 权限许可和访问控制问题漏洞
CVE-2019-13491	47107	Tenda D301 v2 Modem Router - Persistent Cross-Site Scripting	Tenda D301 跨站脚本漏洞
CVE-2016-2534	39405	Jive Forums 5.5.25 - Directory Traversal	Jive Forums 目录遍历漏洞

Table 5.9 shows that CVEs 2014-5329, 2013-2649, 2021-3560 have more than one exploit. It also shows that although most EDB and Chinese titles look alike, they are completely different in the CVEs 2014-5329, 2014-3212, 2017-12854, and 2017-11197. In the last two, the CNNVD does not provide any title or reference, only the following description: “此编号已经被预留用于标识某安全漏洞”, which means an entry code reserved to identify a security vulnerability in the future. It shows that CNNVD is tracking some “RESERVED” status entries in the MITRE CVE, but not all of them. Moreover, among those 16 vulnerabilities listed in Table 5.9, the CNNVD references the EDB as a source of data for the CVEs 2021-3560, 2019-5797, 2019-89242 and 2019-13491. In the last two, the CNNVD references only the EDB. It is another clue that the Chinese databases use human analysts to populate them.

Beyond the cross-reference between the CNNVD and EDB, this work performs a classification taxonomy and authors ranking for the latter in Tables 5.10 and 5.11, respectively.

Table 5.10: Exploit Classification

<i>Type</i>	<i>Description</i>	<i>Amount</i>	<i>Checked(%)</i>
webapps	Target Web Applications	26,260	70.50%
remote	Enable attacks from outside	7,184	88.99%
dos	Denial of Service	6,810	80.46%
local	Requires local access to the target	4665	70.05%
papers	Documents describing attacks	1,672	79.13%
shellcode	Shell code based attacks	1,051	45.77%
<b>Total</b>		<b>47,642</b>	<b>74.43%</b>

Table 5.11: Top 20 Exploit Authors

<i>Rank</i>	<i>Author</i>	<i>Quantity</i>
1	Metasploit	1778
2	Google Security Research	1152
3	Ihsan Sencan	944
4	LiquidWorm	629
5	High-Tech Bridge SA	424
6	Luigi Auriemma	419
7	anonymous	379
8	rgod	333
9	Vulnerability-Lab	312
10	indoushka	297
11	r0t	257
12	hyp3rlinx	228
13	ZoRLu	225
14	ajann	204
15	Lostmon	188
16	shinnai	179
17	Moudi	174
18	laurent gaffe	162
19	GulfTech Security	159
20	GoLd_M	152
<b>Total</b>		<b>8,595</b>

Table 5.10 shows that the EDB has 47,642 vulnerability entries, and almost 50% are exploiting Web Applications. Besides, 74,43% of the exploits have already been tested by the EDB team. Table 5.10 also shows that the remote type has the highest percentage of tested exploits: 88.99%. Moreover, despite the last entry id being 50,850 and following a sequential pattern, there are 47,642 exploits. After profiling the historical series, this work found 3,208 missing ids, the same number of missing exploits.

This work also decided to rank the exploit authors according to the number of artifacts submitted to the EDB. Table 5.11 shows that the top 20 authors submitted 8,595 exploits to the EDB. Since there are 10,232 authors, Table 5.11 indicates that 0.20% of the authors submitted 18.04% of the exploits in the EDB. Moreover, Table 5.11 also shows that Metasploit and Google Security Research are the two most active authors in the EDB.

In order to reach a better situation awareness of the top 20 most active exploit makers, this work decided to build a heatmap for them regarding the used programming languages. Figure 5.17 shows most of them are text-based artifacts. It means that they are white papers or comments on how to exploit some system flaw. Another finding is that Metasploit submits its exploits to the EDB in a standard way: ruby based, as expected by the MSF. It is also possible to grasp some degree of specialization in the authors: *rgod* has more expertise in PHP. On the other hand, *Google Security Research*, *anonymous* and *Luigi Auriemma* have more capacity for building c-based exploits, which require greater programming skills. It is also possible to see that *LiquidWorm* likes to use Python.

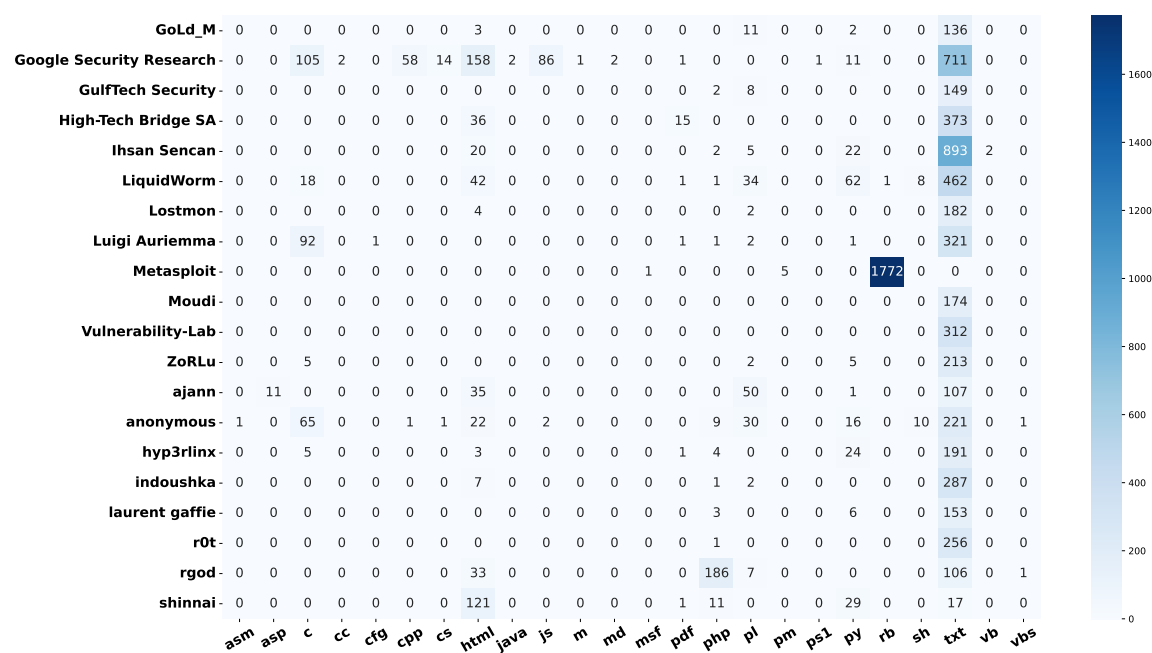


Figure 5.17: Heatmap: Authors vs. Programming Language.

From those 8,595 exploits in the Table 5.11, 4,953 were created after 2010. Due to that, this work got into the top 20 most active authors in the EDB since 2020. Table 5.12 presents those results. It shows, that despite authors like *LiquidWorm*, *Vulnerability-Lab* and *Metasploit* keeping very active, most of them are new authors.

Table 5.12: Top 20 Exploit Authors since 2020

<i>Rank</i>	<i>Author</i>	<i>Quantity</i>
1	LiquidWorm	112
2	Vulnerability-Lab	48
3	boku	48
4	Metasploit	44
5	Ismail Tasdelen	31
6	Ron Jost	26
7	Besim	23
8	SunCSR	21
9	Luis Martínez	21
10	Brian Rodriguez	19
11	Metin Yunus Kandemir	19
12	hyp3rlinx	18
13	0xB9	18
14	Ismael Nava	17
15	Mohammed Alshehri	17
16	Aryan Chehrehgani	16
17	BKpatron	15
18	Matthew Aberegg	15
19	1F98D	15
20	Saeed Bala Ahmed	15
<b>Total</b>		<b>8,595</b>

## 5.4 SCANNER RESULTS

This work uses the attack scenario depicted in Figure 5.1: the attacker outside the topology against the Metasploitable III VM on the DMZ. In this scenario, the attacker runs the scanning process in the host machine against the guest VM emulated by GNS3. He uses the Python Program described in the subsection 4.3 to perform the attack. Table 5.13 lists the detected services.

Table 5.13: Detected Services

<i>Service CPE</i>	<i>Description</i>	<i>Version</i>
cpe:/a:proftpd:proftpd:1.3.5	FTP Server	1.3.5
cpe:/o:linux:linux_kernel	Linux OS Kernel	✗
cpe:/a:apache:http_server:2.4.7	HTTP Server	2.4.7
cpe:/a:samba:samba	Samba File Server	✗
cpe:/a:apple:cups:1.7	Printer Service	1.7
cpe:/a:mysql:mysql	Relational Database	✗
cpe:/a:mortbay:jetty:8.1.7.v20120910	Java Web Server and Servlet Container	8.1.7.v20120910
<b>Total</b>	<b>7</b>	<b>4</b>

Table 5.13 reveals that the Python program based on Nmap step found seven exposed CPEs. Moreover, it identified the version of four of them: *cpe:/a:proftpd:proftpd:1.3.5*, *cpe:/a:apache:http\_server:2.4.7*, *cpe:/a:apple:cups:1.7*, and *cpe:/a:mortbay:jetty:8.1.7.v20120910*.

One of the main purposes of the Scanner module is to return a precise list of CPEs to match them to their possible vulnerabilities. Thus, this work uses only the CPEs with versions to evaluate the vulnerability databases in a precise way in the Matcher module.

## 5.5 MATCHER RESULTS

For the Matcher module, this work did not use the CNVD in the mix of vulnerability databases. The CNVD has only entries after January 2015, so it is different from the entire NVD and CNNVD. Four of those seven vulnerable CPEs include the software version. So, this work seeks vulnerabilities regarding them in the NVD, CNNVD, and Vulners, showing the results listed in Table 5.14.

Table 5.14: Vulnerability Summary

<i>CPE</i>	<i>NVD</i>	<i>CNNVD</i>	<i>Vulners</i>
<i>cpe:/a:proftpd:proftpd:1.3.5</i>	11	11	9
<i>cpe:/a:apache:http_server:2.4.7</i>	46	47	44
<i>cpe:/a:apple:cups:1.7</i>	9	8	5
<i>cpe:/a:mortbay:jetty:8.1.7.v20120910</i>	0	0	0
	66	66	58

Table 5.14 reveals that the Apache webserver has much more vulnerabilities than the other CPEs. It also indicates that the NVD and the CNNVD present almost identical performance in vulnerability identification. Despite Vulners being an established security service, it returns fewer vulnerability entries than this work approach with a mix of the NVD and CNNVD. After the aggregated view in Table 5.14, Table 5.15 presents the vulnerabilities the NVD, CNNVD, or Vulners do not have.

Table 5.15: Unidentified Vulnerabilities

<i>CPE</i>	<i>NVD</i>	<i>CNNVD</i>	<i>Vulners</i>
<i>cpe:/a:proftpd:proftpd:1.3.5</i>	None	None	CVE-2019-12815, CVE-2019-19269
<i>cpe:/a:apache:http_server:2.4.7</i>	CVE-2018-17189	None	CVE-2014-3523, CVE-2018-17189, CVE-2020-13938
<i>cpe:/a:apple:cups:1.7</i>	None	CVE-2014-9679	CVE-2014-9679, CVE-2017-18190, CVE-2017-18248, CVE-2018-4300
	1	1	9

Table 5.15 shows the NVD and CNNVD detected the same CVEs regarding the *cpe:/a:proftpd:proftpd:1.3.5*.

Otherwise, Vulners failed to identify two vulnerabilities: CVE-2019-12815 and CVE-2019-19269. The first is a critical one related to the CVE-2015-3306 that affects the ProFTPD up to version 1.3.5b and allows remote code execution. The second is a NULL Pointer Dereference with medium impact. Table 5.15 also shows that the NVD misses the CVE-2018-1718 regarding the *cpe:/a:apache:http\_server:2.4.7*, while Vulners misses the CVE-2014-3523, CVE-2018-17189 and CVE-2020-13938. There is an issue with the CVE-2018-17189: the NVD says in the description that it affects the Apache HTTP server versions 2.4.37 and below. However, the NVD does not include versions before 2.4.17 in its CPEs affected list. However, the CNNVD includes these versions in its CPE list. The CVE-2014-3523 and CVE-2020-13938 that Vulners did not detect depend on the running OS: it may affect the Apache Server when running on Windows.

Regarding the *cpe:/a:apple:cups:1.7*, the CNNVD and Vulners did not detect the CVE-2014-9679, which depends on the running OS: it affects CUPS before version 2.0.2 when running on Linux OS. The other three, CVE-2017-18190, CVE-2017-18248, and CVE-2018-4300, which only Vulners did not detect, have a CPE description in a version range, not in a CPE list. It suggests that Vulners may have issues in precisely identifying vulnerabilities when the NVD does not provide an exhaustive list of affected CPEs.

The Matcher module uses the previous list of CVEs to look for possible correspondent exploits in the EDB/MSF. Table 5.16 lists those findings.

Table 5.16: List of Exploits

<i>CVE</i>	<i>Exploit ID</i>	<i>Description</i>	<i>Checked</i>
CVE-2014-0226	34133	Apache 2.4.7 mod_status - Scoreboard Handling Race Condition	✗
CVE-2015-3306	36742	ProFTPD 1.3.5 - File Copy	✓
CVE-2015-3306	36803	ProFTPD 1.3.5 - 'mod_copy' Remote Command Execution	✗
CVE-2015-3306	37262	ProFTPD 1.3.5 - 'mod_copy' Command Execution (Metasploit)	✓
CVE-2015-3306	49908	ProFTPD 1.3.5 - 'mod_copy' Remote Command Execution	✓
CVE-2016-0736	40961	Apache mod_session_crypto - Padding Oracle	✗
CVE-2017-9798	42745	Apache < 2.2.34 / < 2.4.27 - OPTIONS Memory Leak	✗
CVE-2019-10092	47688	Apache Httpd mod_proxy - Error Page Cross-Site Scripting	✗
CVE-2019-10098	47689	Apache Httpd mod_rewrite - Open Redirects	✗

Table 5.16 shows that this work detected nine exploits in its database. They target six vulnerabilities (CVEs) since four exploits address the same vulnerability, CVE-2015-3306. Moreover, the EDB team checked only three of those nine artifacts, all targeting the CVE-2015-3306. Most of the information security assessment and penetration testers would stop in this step, since there is a list of vulnerabilities with the correspondent exploits, being the CVE-2015-3306 a strong candidate to be a target of exploitation. Nonetheless, this work proposes a Recommender Vulnerability-Exploit System (REVS) using TOPSIS with entropy-based weights as described in section 4.5 to rank the vulnerabilities using their features and their possible matching exploits.

## 5.6 RECOMMENDER RESULTS

Lastly, this work ranks the vulnerabilities using the procedure described in Section 4.5. As the first step, this work calculates the weights for the nine criteria of the decision matrix. Table 5.17 presents those weights values in four scenarios: the first two using the original proposal considering possible correspondent exploits and the other two with only the CVSS features.

Table 5.17: Criteria Weight Values

<i>Features</i>	<i>Scenario 1</i>	<i>Scenario 2</i>	<i>Scenario 3</i>	<i>Scenario 4</i>
<i>Access Vector</i>	0.20%	11.11%	0.80%	14.29%
<i>Access Complexity</i>	0.10%	11.11%	0.38%	14.29%
<i>Authentication</i>	0.01%	11.11%	0.06%	14.29%
<i>Confidentiality Impact</i>	9.40%	11.11%	37.13%	14.29%
<i>Integrity Impact</i>	7.96%	11.11%	31.44%	14.29%
<i>Availability Impact</i>	6.10%	11.11%	24.10%	14.29%
<i>Time (Days)</i>	1.55%	11.11%	6.09%	14.29%
<i>Qt. Exploits</i>	28.70%	11.11%	✗	✗
<i>Qt. Checked</i>	45.99%	11.11%	✗	✗

Table 5.17 shows that in the first scenario, taking into account the existent artifacts in the EDB, the exploit features have the most weight values, with almost 75%. Without considering the artifacts in the EDB, the third scenario shows that the confidentiality, integrity, and availability impact features gain most of the weight, almost 98%. It indicates two things: the exploit features carry most of the information in the decision process of which vulnerabilities should be handled first. Table 5.17 also shows that without considering the exploits, the confidentiality, integrity, and availability impact features carry most of the information for the decision process. Besides, Table 5.17 has two more scenarios, 2 and 4, considering an equally weighted approach.

Table 5.18 shows the results using the scenarios for the top 10 CVEs ranking with TOPSIS using entropy-based and equally weighted criteria. Table 5.18 indicates that, despite the scenario, this work ranks the CVE-2015-3306 as the riskiest. It also shows that despite shifting from an entropy-based to an equally weighted strategy in scenarios 1 and 2, the top three vulnerabilities keep the same. The exploit existence for those three vulnerabilities can explain it. Besides, those two scenarios return the CVEs 2021-44790, 2021-39275, 2021-40438, and 2021-26691, which shows the importance of the time feature as a cost criterium that the decision process must minimize.

Lastly, scenarios 3 and 4, an entropy-based approach against an equally weighted without considering exploit features, show virtually the same results. There is only an interchange in the CVEs 2021-26691 and 2021-40438 positions. Those top 10 vulnerabilities have very similar CVSS features; therefore, the time feature has more influence on the decision process.

Table 5.18: TOPSIS Vulnerability Rank

<i>Rank</i>	<i>Scenario 1</i>	<i>Scenario 2</i>	<i>Scenario 3</i>	<i>Scenario 4</i>
1	CVE-2015-3306	CVE-2015-3306	CVE-2015-3306	CVE-2015-3306
2	CVE-2014-0226	CVE-2014-0226	CVE-2021-44790	CVE-2021-44790
3	CVE-2019-10098	CVE-2019-10098	CVE-2021-39275	CVE-2021-39275
4	CVE-2017-9798	CVE-2021-44790	CVE-2021-40438	CVE-2021-26691
5	CVE-2016-0736	CVE-2021-39275	CVE-2021-26691	CVE-2021-40438
6	CVE-2019-10092	CVE-2021-40438	CVE-2020-35452	CVE-2020-35452
7	CVE-2021-44790	CVE-2021-26691	CVE-2019-12815	CVE-2019-12815
8	CVE-2021-39275	CVE-2020-35452	CVE-2019-0217	CVE-2019-0217
9	CVE-2021-40438	CVE-2019-10092	CVE-2018-1312	CVE-2018-1312
10	CVE-2021-26691	CVE-2019-12815	CVE-2017-15715	CVE-2017-15715

## 5.7 SUMMARY

This chapter explained the experimental results achieved with the Cyber Intelligence Pipeline. It compares three national vulnerability databases, the NVD, CNVD, and CNNVD, and also linked them to the Exploit Database (EDB). It demonstrated that the China vulnerability database is not a mere translation of the NVD, even for the registers that they share. It revealed that the China vulnerability databases are ahead of the USA and leveraging exploit data to infer reserved vulnerabilities that do not exist in the NVD. The pipeline also outperforms Vulners in vulnerability detection, enhancing information security assessment and penetration testing. Lastly, it improves the vulnerability ranking through exploit criteria and entropy-bases TOPSIS decision process.



## 6 CONCLUSIONS AND FUTURE WORKS

This work created an ETL data pipeline to compare three national vulnerability databases from USA and China: the NVD, CNVD, and CNNVD. Besides, this work also used that pipeline to get, process, and link the EDB to them. This work downloaded those national databases simultaneously during the Harvesting Step to enable a fair comparison between them. Moreover, it leverages a Python parallel multiprocessing approach to optimize the transform and load phases of the ETL. Furthermore, it uses pre-trained neural networks with GPU parallel processing to perform multilingual NLP and compare vulnerability descriptions in English and Chinese.

Despite claiming themselves as public XML databases, the CNVD and CNNVD hinder access. Several sign-in issues demanded building custom crawlers and leveraging data scrape techniques to download the XML dataset. Although the CNVD has 99,261 entries, the historical series started in 2015, it has six weeks without data in its XML data feed, and previous files need to be updated, suggesting an intention to hide information or process issues. Most text data are in Mandarin except for URLs, software, and vendor names. Those two vulnerability databases have content tailored to the local Chinese community.

The CNNVD has 1,661 more entries than the NVD, the standard vulnerability database. That database has 25 classified registers, which is compliant with Chinese legislation but against the best practices of information sharing regarding system vulnerabilities. Moreover, the CNNVD contains at least 35 and 5 more Huawei and ZTE product entries than the NVD. That database resembles this but has fewer CVSS metrics. Besides, the CNNVD provides an extensive list of vulnerable CPEs but does not offer vulnerable software combinations like the NVD. The CNNVD also provides richer information regarding external references and possible solutions. Further, a temporal correlation of 0.917560 between the NVD and CNNVD indicates shared sources or that they are tracking each other. On the other hand, the pre-trained multilingual model LaBSE achieved a cosine similarity between the NVD and CNNVD around 72%. It indicates that the latter is not just a translated version of the earlier and human intervention by cyber analysts to build the CNNVD.

The Chinese national vulnerabilities are comprehensive databases, particularly the CNNVD. It has 3.2% of the entries without URL references, while the NVD has 13.43% with the same issue. For that reason, the CNNVD is a promising source of extra data regarding exploits and possible solutions to vulnerabilities. Moreover, the CNNVD has 5.52% of the entries without CVE mapping, possibly suggesting system flaws only known to the Chinese community.

After collecting and linking the 47642 exploits from the EDB with the national databases, this work found 35 exploited CVEs that do not exist in the NVD. However, the Chinese databases have 16 of those 35, indicating they are ahead of the NVD. They use several data sources to keep track of up-to-date vulnerabilities and overcome the NVD. This work demonstrates that the CNNVD tracks entries in the MITRE CVE before going to the NVD. Moreover, the CNNVD is leveraging exploit data in the EDB to infer system vulnerabilities.

This work evaluates network scanning with a mix of the NVD and CNNVD as vulnerability data sources. REVS performed better than an established commercial database like Vulners, which did not identify nine

vulnerabilities detected by this work. Vulners has issues returning vulnerabilities about CPEs described as version range or logical expressions. The entropy-based criteria values indicate that the exploit features have more influence in the multicriteria decision than the CVSS parameters. TOPSIS ranking method with those entropy-based weights classified the CVE-2015-3306, CVE-2014-0226, and CVE-2019-10098 riskiest vulnerabilities in the emulated environment. As far as the author knows, this is the first approach to leverage a mix of the NVD and CNNVD for network scanning and perform a thorough comparison between them, including multilingual NLP.

## **6.1 FUTURE WORKS**

As future work, this research suggests a complete Natural Language Processing (NLP) customized to the Chinese language to extract more features from the CNVD and CNNVD. One suggested approach is the Han Language Processing (HanLP) library with pre-trained models.

A second idea for future work is the creation of training and test datasets joining the NVD, CNVD, CNNVD, and EDB for vulnerability exploitation prediction. Since the relational database is available in this work, it may provide ML model training and evaluation features to predict vulnerability exploitation and prioritization. Lastly, several other MCDA algorithms, like VIKOR, can be used to rank vulnerabilities, leaving an idea for future work to compare those methods and evaluate a sensitivity analysis of the weight values in the vulnerability ranking results.

## REFERENCES

- 1 MITRE. *The CVE Structure*. 2022. <https://www.cve.org/ProgramOrganization/Structure>. [Online; accessed 09-April-2022].
- 2 FRODRIGUEZ. *Maven Repository*. 2021. <https://mvnrepository.com/>. [Online; accessed 06-December-2021].
- 3 MITRE. *The CWE Framework*. 2021. <https://cwe.mitre.org/>. [Online; accessed 06-December-2021].
- 4 FIRST. *Common Vulnerability Scoring System v3.1: Specification Document*. 2021. <https://www.first.org/cvss/v3.1/specification-document>. [Online; accessed 06-December-2021].
- 5 MITRE. *The CAPEC Framework*. 2021. <https://capec.mitre.org/>. [Online; accessed 06-December-2021].
- 6 NIST. *National Vulnerability Database (NVD)*. 2021. <https://nvd.nist.gov/vuln/data-feeds>. [Online; accessed 10-January-2022].
- 7 CNCERT/CC. *China National Vulnerability Database (CNVD)*. 2021. <https://www.cnvd.org.cn/shareData/list>. [Online; accessed 10-January-2022].
- 8 CNITSEC. *China National Vulnerability Database of Information Security (CNVD)*. 2021. <http://www.cnnvd.org.cn/web/xxk/xmlDown.tag>. [Online; accessed 10-January-2022].
- 9 JVN. *The Japan Vulnerability Notes iPedia (JVND)*. 2022. <https://jvndb.jvn.jp/en/>. [Online; accessed 22-April-2022].
- 10 OFFSEC. *The Exploit Database*. 2022. <https://www.exploit-db.com/>. [Online; accessed 10-April-2022].
- 11 VULNERS, I. *VULNERS, INC*. 2021. <https://vulners.com/>. [Online; accessed 22-April-2022].
- 12 RYTEL, M.; FELKNER, A.; JANISZEWSKI, M. Towards a safer internet of things—a survey of iot vulnerability data sources. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 20, n. 21, p. 5969, 2020.
- 13 JANISZEWSKI, M.; FELKNER, A.; LEWANDOWSKI, P.; RYTEL, M.; ROMANOWSKI, H. Automatic actionable information processing and trust management towards safer internet of things. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 21, n. 13, p. 4359, 2021.
- 14 BEJARANO, M. H.; RODRÍGUEZ, R. J.; MERSEGUER, J. A vision for improving business continuity through cyber-resilience mechanisms and frameworks. In: IEEE. *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.], 2021. p. 1–5.
- 15 LALLIE, H. S.; SHEPHERD, L. A.; NURSE, J. R.; EROLA, A.; EPIPHANIOU, G.; MAPLE, C.; BELLEKENS, X. Cyber security in the age of covid-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic. *Computers & Security*, Elsevier, v. 105, p. 102248, 2021.
- 16 AMARULLAH, A. H.; RUNTURAMBI, A. J. S.; WIDIWAN, B. Analyzing cyber crimes during covid-19 time in indonesia. In: IEEE. *2021 3rd International Conference on Computer Communication and the Internet (ICCCI)*. [S.l.], 2021. p. 78–83.
- 17 DU, G.; LONG, C.; YU, J.; WAN, W.; ZHAO, J.; WEI, J. A real-time big data framework for network security situation monitoring. In: *ICEIS (I)*. [S.l.: s.n.], 2019. p. 167–175.

- 18 WILLIAMS, M. A.; DEY, S.; BARRANCO, R. C.; NAIM, S. M.; HOSSAIN, M. S.; AKBAR, M. Analyzing evolving trends of vulnerabilities in national vulnerability database. In: IEEE. *2018 IEEE International Conference on Big Data (Big Data)*. [S.l.], 2018. p. 3011–3020.
- 19 JIA, Y.; QI, Y.; SHANG, H.; JIANG, R.; LI, A. A practical approach to constructing a knowledge graph for cybersecurity. *Engineering*, Elsevier, v. 4, n. 1, p. 53–60, 2018.
- 20 WANG, J. A.; GUO, M. Ovm: an ontology for vulnerability management. In: *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*. [S.l.: s.n.], 2009. p. 1–4.
- 21 GAO, J.-b.; ZHANG, B.-w.; CHEN, X.-h.; LUO, Z. Ontology-based model of network and computer attacks for security assessment. *Journal of Shanghai Jiaotong University (Science)*, Springer, v. 18, n. 5, p. 554–562, 2013.
- 22 MITRE. *The MITRE Corporation*. 2021. <https://www.mitre.org/>. [Online; accessed 06-December-2021].
- 23 RODRIGUEZ, L. G. A.; TRAZZI, J. S.; FOSSALUZA, V.; CAMPIOLO, R.; BATISTA, D. M. Analysis of vulnerability disclosure delays from the national vulnerability database. In: SBC. *Anais do I Workshop de Segurança Cibernética em Dispositivos Conectados*. [S.l.], 2018.
- 24 JIN, R.; NAN, J. Combining sources from cve and cnvd: Data analysis in information security vulnerabilities. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2021. v. 1800, n. 1, p. 012004.
- 25 FENG, X.; LIAO, X.; WANG, X.; WANG, H.; LI, Q.; YANG, K.; ZHU, H.; SUN, L. Understanding and securing device vulnerabilities through automated bug report analysis. In: *SEC'19: Proceedings of the 28th USENIX Conference on Security Symposium*. [S.l.: s.n.], 2019.
- 26 GONZALEZ, D.; HASTINGS, H.; MIRAKHORLI, M. Automated characterization of software vulnerabilities. In: IEEE. *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. [S.l.], 2019. p. 135–139.
- 27 MORIUCHI, P.; LADD, B. *China's Ministry of State Security Likely Influences National Network Vulnerability Publications*. 2021. <https://www.recordedfuture.com/chinese-mss-vulnerability-influence/>. [Online; accessed 06-December-2021].
- 28 KANAKOGI, K.; WASHIZAKI, H.; FUKAZAWA, Y.; OGATA, S.; OKUBO, T.; KATO, T.; KANUKA, H.; HAZEYAMA, A.; YOSHIOKA, N. Tracing capec attack patterns from cve vulnerability information using natural language processing technique. In: *Proceedings of the 54th Hawaii International Conference on System Sciences*. [S.l.: s.n.], 2021. p. 6996.
- 29 BERTOGLIO, D. D.; ZORZO, A. F. Overview and open issues on penetration test. *Journal of the Brazilian Computer Society*, SpringerOpen, v. 23, n. 1, p. 1–16, 2017.
- 30 KANAKOGI, K.; WASHIZAKI, H.; FUKAZAWA, Y.; OGATA, S.; OKUBO, T.; KATO, T.; KANUKA, H.; HAZEYAMA, A.; YOSHIOKA, N. Tracing cve vulnerability information to capec attack patterns using natural language processing techniques. *Information*, Multidisciplinary Digital Publishing Institute, v. 12, n. 8, p. 298, 2021.
- 31 TSANTILIS, I.; DASAKLIS, T. K.; DOULIGERIS, C.; PATSAKIS, C. Searching deterministic chaotic properties in system-wide vulnerability datasets. In: MULTIDISCIPLINARY DIGITAL PUBLISHING INSTITUTE. *Informatics*. [S.l.], 2021. v. 8, n. 4, p. 86.

- 32 PHAM, V.; DANG, T. Cvexplorer: Multidimensional visualization for common vulnerabilities and exposures. In: IEEE. *2018 IEEE International Conference on Big Data (Big Data)*. [S.l.], 2018. p. 1296–1301.
- 33 HOUSEHOLDER, A. D.; CHRABASZCZ, J.; NOVELLY, T.; WARREN, D.; SPRING, J. M. Historical analysis of exploit availability timelines. In: *13th {USENIX} Workshop on Cyber Security Experimentation and Test ({CSET} 20)*. [S.l.: s.n.], 2020.
- 34 SHI, K.; DAI, Y.; XU, J. Construction of a security vulnerability identification system based on machine learning. *Journal of Sensors*, Hindawi, v. 2020, 2020.
- 35 SHAH, S.; MEHTRE, B. M. An overview of vulnerability assessment and penetration testing techniques. *Journal of Computer Virology and Hacking Techniques*, Springer, v. 11, n. 1, p. 27–49, 2015.
- 36 YAQOOB, I.; HUSSAIN, S. A.; MAMOON, S.; NASEER, N.; AKRAM, J.; REHMAN, A. ur. Penetration testing and vulnerability assessment. *Journal of Network Communications and Emerging Technologies (JNCET) www.jncet.org*, v. 7, n. 8, 2017.
- 37 GHANEM, M. C.; CHEN, T. M. Reinforcement learning for efficient network penetration testing. *Information*, Multidisciplinary Digital Publishing Institute, v. 11, n. 1, p. 6, 2020.
- 38 ZHOU, S.; LIU, J.; HOU, D.; ZHONG, X.; ZHANG, Y. Autonomous penetration testing based on improved deep q-network. *Applied Sciences*, Multidisciplinary Digital Publishing Institute, v. 11, n. 19, p. 8823, 2021.
- 39 AWISZUS, M.; ROSENHAHN, B. Markov chain neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. [S.l.: s.n.], 2018. p. 2180–2187.
- 40 CHENG, P.; WANG, L.; JAJODIA, S.; SINGHAL, A. Aggregating cvss base scores for semantics-rich network security metrics. In: IEEE. *2012 IEEE 31st Symposium on Reliable Distributed Systems*. [S.l.], 2012. p. 31–40.
- 41 POLATIDIS, N.; PAVLIDIS, M.; MOURATIDIS, H. Cyber-attack path discovery in a dynamic supply chain maritime risk management system. *Computer Standards & Interfaces*, Elsevier, v. 56, p. 74–82, 2018.
- 42 POLATIDIS, N.; GEORGIADIS, C. K. A dynamic multi-level collaborative filtering method for improved recommendations. *Computer Standards & Interfaces*, Elsevier, v. 51, p. 14–21, 2017.
- 43 LIMA, J. C. D.; ROCHA, C. C.; AUGUSTIN, I. et al. A context-aware recommendation system to behavioral based authentication in mobile and pervasive environments. In: IEEE. *2011 IFIP 9th International Conference on Embedded and Ubiquitous Computing*. [S.l.], 2011. p. 312–319.
- 44 DOŽIĆ, S. Multi-criteria decision making methods: Application in the aviation industry. *Journal of Air Transport Management*, Elsevier, v. 79, p. 101683, 2019.
- 45 KHOLIDY, H. A. Multi-layer attack graph analysis in the 5g edge network using a dynamic hexagonal fuzzy method. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 22, n. 1, p. 9, 2022.
- 46 LIU, N.; ZHANG, J.; ZHANG, H.; LIU, W. Security assessment for communication networks of power control systems using attack graph and mcdm. *IEEE Transactions on Power Delivery*, IEEE, v. 25, n. 3, p. 1492–1500, 2010.
- 47 ALHARBI, A.; SEH, A. H.; ALOSAIMI, W.; ALYAMI, H.; AGRAWAL, A.; KUMAR, R.; KHAN, R. A. Analyzing the impact of cyber security related attributes for intrusion detection systems. *Sustainability*, Multidisciplinary Digital Publishing Institute, v. 13, n. 22, p. 12337, 2021.

- 48 BYERS, R.; WALTERMIRE, D.; TURNER, C. et al. *Collaborative Vulnerability Metadata Acceptance Process (CVMAP) for CVE Numbering Authorities (CNAs) and Authorized Data Publishers*. [S.l.], 2020.
- 49 ANWAR, A.; ABUSNAINA, A.; CHEN, S.; LI, F.; MOHAISEN, D. Cleaning the nvd: Comprehensive quality assessment, improvements, and analyses. *IEEE Transactions on Dependable and Secure Computing*, IEEE, 2021.
- 50 HORAWALAVITHANA, S.; BHATTACHARJEE, A.; LIU, R.; CHOUDHURY, N.; HALL, L. O.; IAMNITCHI, A. Mentions of security vulnerabilities on reddit, twitter and github. In: *IEEE/WIC/ACM International Conference on Web Intelligence*. [S.l.: s.n.], 2019. p. 200–207.
- 51 SHIM, Y.; SHIN, D.-H. Neo-techno nationalism: The case of china’s handset industry. *Telecommunications Policy*, Elsevier, v. 40, n. 2-3, p. 197–209, 2016.
- 52 REN, H.; LI, X.; LEI, L.; OU, G.; SUN, H.; WU, G.; TIAN, X.; HU, J.; ZHANG, Y. Detecting inconsistent vulnerable software version in security vulnerability reports. In: SPRINGER. *International Conference on Frontiers in Cyber Security*. [S.l.], 2021. p. 78–99.
- 53 OLSWANG, A.; GONDA, T.; PUZIS, R.; SHANI, G.; SHAPIRA, B.; TRACTINSKY, N. Prioritizing vulnerability patches in large networks. *Expert Systems with Applications*, Elsevier, p. 116467, 2022.
- 54 JACOBS, J.; ROMANOSKY, S.; EDWARDS, B.; ADJERID, I.; ROYTMAN, M. Exploit prediction scoring system (epss). *Digital Threats: Research and Practice*, ACM New York, NY, USA, v. 2, n. 3, p. 1–17, 2021.
- 55 FALL, D.; KADOBAYASHI, Y. The common vulnerability scoring system vs. rock star vulnerabilities: Why the discrepancy? In: *ICISSP*. [S.l.: s.n.], 2019. p. 405–411.
- 56 FORAIN, I.; ALBUQUERQUE, R. de O.; JÚNIOR, R. T. de S. Revs: A vulnerability ranking tool for enterprise security. In: *ICEIS (2)*. [S.l.: s.n.], 2022. p. 126–133.
- 57 FORAIN, I.; ALBUQUERQUE, R. de O.; JÚNIOR, R. T. de S. Towards system security: What a comparison of national vulnerability databases reveals. In: *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.: s.n.], 2022. p. 1–6.
- 58 ROGERS, R. *Nessus network auditing*. [S.l.]: Elsevier, 2011.
- 59 JETTY, S. *Network Scanning Cookbook: Practical Network Security Using Nmap and Nessus 7*. [S.l.]: Packt Publishing Ltd, 2018.
- 60 AKSU, M. U.; ALTUNCU, E.; BICAKCI, K. A first look at the usability of openvas vulnerability scanner. In: *Workshop on usable security (USEC)*. [S.l.: s.n.], 2019.
- 61 LYON, G. F. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. [S.l.]: Insecure. Com LLC (US), 2008.
- 62 AL., Q. et. *OWASP Nettacker*. 2021. <https://owasp.org/www-project-nettacker/>. [Online; accessed 06-December-2021].
- 63 ZAMIRI-GOURABI, M.-R.; QALAEI, A. R.; AZAD, B. A. Gas what? i can see your gaspots. studying the fingerprintability of ics honeypots in the wild. In: *Proceedings of the fifth annual industrial control system security (icss) workshop*. [S.l.: s.n.], 2019. p. 30–37.
- 64 SINGH, A.; JASWAL, N.; AGARWAL, M.; TEIXEIRA, D. *Metasploit Penetration Testing Cookbook: Evade antiviruses, bypass firewalls, and exploit complex environments with the most widely used penetration testing framework*. [S.l.]: Packt Publishing Ltd, 2018.

- 65 MITRE. *The CVE Framework*. 2021. <https://cve.mitre.org/>. [Online; accessed 06-December-2021].
- 66 DONG, Y.; GUO, W.; CHEN, Y.; XING, X.; ZHANG, Y.; WANG, G. Towards the detection of inconsistencies in public security vulnerability reports. In: *28th {USENIX} Security Symposium ({USENIX} Security 19)*. [S.l.: s.n.], 2019. p. 869–885.
- 67 CHEIKES, B. A.; CHEIKES, B. A.; KENT, K. A.; WALTERMIRE, D. *Common platform enumeration: Naming specification version 2.3*. [S.l.]: US Department of Commerce, National Institute of Standards and Technology, 2011.
- 68 TOVARŇÁK, D.; SADLEK, L.; ČELEDA, P. Graph-based cpe matching for identification of vulnerable asset configurations. In: *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. [S.l.: s.n.], 2021. p. 986–991.
- 69 WÅREUS, E.; HELL, M. Automated cpe labeling of cve summaries with machine learning. In: SPRINGER. *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. [S.l.], 2020. p. 3–22.
- 70 MELL, P.; SCARFONE, K.; ROMANOSKY, S. Common vulnerability scoring system. *IEEE Security & Privacy*, IEEE, v. 4, n. 6, p. 85–89, 2006.
- 71 MELL, P.; SCARFONE, K.; ROMANOSKY, S. et al. A complete guide to the common vulnerability scoring system version 2.0. In: *Published by FIRST-forum of incident response and security teams*. [S.l.: s.n.], 2007. v. 1, p. 23.
- 72 ABRAHAM, R.; ARORA, D.; COLES, M.; ECKERT, M.; HEITMAN, M.; MANION, A.; MOORE, S.; ROMANOWSKY, S.; SCARFONE, K.; STUPPI, J. et al. Common vulnerability scoring system v3. 0: Specification document. *First*, 2015.
- 73 CNCERT/CC. *The National Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC)*. 2021. <https://www.cert.org.cn/publish/english/index.html>. [Online; accessed 10-January-2022].
- 74 CNITSEC. *The China Information Technology Security Evaluation Center (CNITSEC)*. 2021. <http://www.itsec.gov.cn/>. [Online; accessed 10-January-2022].
- 75 SECURITYFOCUS. *Bug Traq*. 2022. <https://bugtraq.securityfocus.com/archive>. [Online; accessed 22-April-2022].
- 76 RAPID7. *Metasploit DB*. 2022. <https://www.rapid7.com/db/>. [Online; accessed 10-April-2022].
- 77 JASWAL, N. *Mastering Metasploit: Exploit systems, cover your tracks, and bypass security controls with the Metasploit 5.0 framework*. [S.l.]: Packt Publishing Ltd, 2020.
- 78 OFFSEC. *Offensive Security*. 2022. <https://www.offensive-security.com/>. [Online; accessed 10-April-2022].
- 79 VALEA, O.; OPRIȘA, C. Towards pentesting automation using the metasploit framework. In: IEEE. *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*. [S.l.], 2020. p. 171–178.
- 80 HU, Z.; BEURAN, R.; TAN, Y. Automated penetration testing using deep reinforcement learning. In: IEEE. *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. [S.l.], 2020. p. 2–10.
- 81 HU Zhenguo. *Automated Penetration Testing Using Deep Reinforcement Learning*. 63 p. Dissertação (Mestrado) — Japan Advanced Institute of Science and Technology JAIST, 2021.

- 82 OU, X.; GOVINDAVAJHALA, S.; APPEL, A. W. et al. Mulval: A logic-based network security analyzer. In: BALTIMORE, MD. *USENIX security symposium*. [S.l.], 2005. v. 8, p. 113–128.
- 83 JIANG, Y.; JEUSFELD, M.; DING, J. Evaluating the data inconsistency of open-source vulnerability repositories. In: *The 16th International Conference on Availability, Reliability and Security*. [S.l.: s.n.], 2021. p. 1–10.
- 84 LIU, K.; ZHOU, Y.; WANG, Q.; ZHU, X. Vulnerability severity prediction with deep neural network. In: IEEE. *2019 5th International Conference on Big Data and Information Analytics (BigDIA)*. [S.l.], 2019. p. 114–119.
- 85 PAWLICKA, A.; PAWLICKI, M.; KOZIK, R.; CHORAŚ, R. S. A systematic review of recommender systems and their applications in cybersecurity. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 21, n. 15, p. 5248, 2021.
- 86 POLATIDIS, N.; PIMENIDIS, E.; PAVLIDIS, M.; PAPASTERGIOU, S.; MOURATIDIS, H. From product recommendation to cyber-attack prediction: generating attack graphs and predicting future attacks. *Evolving Systems*, Springer, v. 11, n. 3, p. 479–490, 2020.
- 87 BĄCZKIEWICZ, A.; WĄTRÓBSKI, J. Crispyn—a python library for determining criteria significance with objective weighting methods. *SoftwareX*, Elsevier, v. 19, p. 101166, 2022.
- 88 WĄTRÓBSKI, J.; BĄCZKIEWICZ, A.; SAŁABUN, W. pyrepo-mcda—reference objects based mcda software package. *SoftwareX*, Elsevier, v. 19, p. 101107, 2022.
- 89 WĄTRÓBSKI, J.; BĄCZKIEWICZ, A.; SAŁABUN, W. Version [1.1]—[pyrepo-mcda—reference objects based mcda software package]. *SoftwareX*, Elsevier, v. 19, p. 101197, 2022.
- 90 SHANNON, C. E. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, ACM New York, NY, USA, v. 5, n. 1, p. 3–55, 2001.
- 91 SIBAL, R.; SHARMA, R.; SABHARWAL, S. Prioritizing software vulnerability types using multi-criteria decision-making techniques. *Life Cycle Reliability and Safety Engineering*, Springer, v. 6, n. 1, p. 57–67, 2017.
- 92 SHARMA, R.; SIBAL, R.; SABHARWAL, S. Software vulnerability prioritization: A comparative study using topsis and vikor techniques. In: *System performance and management analytics*. [S.l.]: Springer, 2019. p. 405–418.
- 93 CHEN, C.-H. A novel multi-criteria decision-making model for building material supplier selection based on entropy-ahp weighted topsis. *Entropy*, MDPI, v. 22, n. 2, p. 259, 2020.
- 94 CROCFER, N. *CVE Alerting Platform*. 2020. <<https://github.com/opencve/opencve>>. [Online; accessed 06-December-2021].
- 95 ROTEN, T.; WANG, X. *GitHub - tsroten/hanzidentifier: Python module that identifies Chinese text as being Simplified or Traditional*. 2015. <<https://github.com/tsroten/hanzidentifier>>. [Online; accessed 2022-03-20].
- 96 YUE, B.; GALLANT, A. *GitHub - berniey/hanziconv: Hanzi Converter for Traditional and Simplified Chinese*. 2016. <<https://github.com/berniey/hanziconv>>. [Online; accessed 2022-03-20].
- 97 REIMERS, N. *SBERT.Net*. 2022. <<https://www.sbert.net/>>. [Online; accessed 06-July-2022].
- 98 REIMERS, N.; GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2019. Disponível em: <<https://arxiv.org/abs/1908.10084>>.



- 99 REIMERS, N.; GUREVYCH, I. Making monolingual sentence embeddings multilingual using knowledge distillation. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2020. Disponível em: <<https://arxiv.org/abs/2004.09813>>.
- 100 FENG, F.; YANG, Y.; CER, D.; ARIVAZHAGAN, N.; WANG, W. Language-agnostic bert sentence embedding. *arXiv preprint arXiv:2007.01852*, 2020.
- 101 FATAHALIAN, K.; SUGERMAN, J.; HANRAHAN, P. Understanding the efficiency of gpu algorithms for matrix-matrix multiplication. In: *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. [S.l.: s.n.], 2004. p. 133–137.
- 102 HWANG, C.-L.; YOON, K. Multiple attribute decision making: a state of the art survey. *Lecture Notes in Economics and Mathematical Systems*, Springer-Verlag Berlin, v. 186, n. 1, 1981.
- 103 CHEN, S.-J.; HWANG, C.-L. Fuzzy multiple attribute decision making methods. *Fuzzy multiple attribute decision making*, Springer, p. 289–486, 1992.
- 104 OPRICOVIC, S.; TZENG, G.-H. Compromise solution by mcdm methods: A comparative analysis of vikor and topsis. *European journal of operational research*, Elsevier, v. 156, n. 2, p. 445–455, 2004.
- 105 RAPID7. *The Metasploitable 3 Linux VM*. 2021. <https://github.com/rapid7/metasploitable3>. [Online; accessed 10-January-2022].