The analysis method of security vulnerability based on the knowledge graph

Yongfu Wang
Sun Yat-sen University, Guangzhou
Guangdong, China
wangyf285@mail2.sysu.edu.cn

Ying Zhou
Sun Yat-sen University, Guangzhou
Guangdong, China
zhouying5@mail.sysu.edu.cn

Xiaohai Zou
Sun Yat-sen University, Guangzhou
Guangdong, China
zouxiaohai2000@sina.com

Quanqiang Miao
China Luoyang Electronic Equipment
Test Center, Luoyang Henan, China
miaoquanqiang@163.com

Wei Wang
Sun Yat-sen University, Guangzhou
Guangdong, China
wangw278@mail.sysu.edu.cn

ABSTRACT

Given the increasingly prominent network security issues, it is of great significance to deeply analyze the vulnerability of network space software and hardware resources. Although the existing Common Vulnerabilities and Exposures (CVE) security vulnerability database contains a wealth of vulnerability information, the information is poorly readable, the potential correlation is difficult to express intuitively, and the degree of visualization is insufficient. To solve the current problems, a method of constructing a knowledge graph of CVE security vulnerabilities is proposed. By acquiring raw data, ontology modeling, data extraction and import, the knowledge graph is imported into the Neo4j graph database to complete the construction of the CVE knowledge graph. Based on the knowledge graph, the in-depth analysis is performed from the cause dimension, time dimension and association dimension, and the results are displayed visually. Experiments show that this analysis method can intuitively and effectively mine the intrinsic value of CVE security vulnerability data.

CCS CONCEPTS

• **Security and privacy**; • **Software and application security**; • **Software security engineering**;

KEYWORDS

CVE, ontology modeling, knowledge graph, Neo4j graph database, association relationship

ACM Reference Format:
Yongfu Wang, Ying Zhou, Xiaohai Zou, Quanqiang Miao, and Wei Wang. 2020. The analysis method of security vulnerability based on the knowledge graph. In *2020 the 10th International Conference on Communication and Network Security (ICCNS 2020), November 27–29, 2020, Tokyo, Japan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3442520.3442535

ICCNS 2020, November 27–29, 2020, Tokyo, Japan

ACM ISBN 978-1-4503-8903-7/20/11…$15.00
https://doi.org/10.1145/3442520.3442535

1 INTRODUCTION

Nowadays, network security problems are becoming more and more serious with the continuous deepening of social informatization and intelligence. Among them, various attacks against existing vulnerabilities pose a great threat to the security of networks and information systems. Through an in-depth analysis of existing vulnerability information, digging out the implicit relationship between related information is of great significance for resisting external attacks and timely discovering and repairing vulnerabilities.

In terms of security vulnerability analysis, Minzhe Guo et al. [1] proposed an ontology-based method to model the Common Vulnerabilities and Exposures (CVE) security vulnerability library, with the help of the concepts and axioms in the ontology. And basic attributes to discover the complex relationships between individuals, between individuals and concepts, and between concepts. Shengzhi Qin et al. [2] used CVE, Common Platform Enumeration (CPE), Common Weakness Enumeration (CWE), Common Vulnerability Scoring System (CVSS) data set provided by National Vulnerability Database (NVD), build a vulnerability knowledge graph called VulKG, and propose an automated analysis and inference vulnerability model, which can automatically analyze and extract named entities in the vulnerability description field and add them to VulKG, and implement inference on this basis Function, its focus is on reasoning related CWE. Mittal S [3] and others based on multi-source data sets, combined with vector space and knowledge graph, used the advantages of vector space in similarity calculation and neighborhood search, and knowledge graph suitable for declarative information and perfect reasoning ability, constructed a The structure of vector space and knowledge graph can effectively answer those queries that cannot be processed by vector space or knowledge graph alone.

In terms of ontology data storage and information retrieval based on Neo4j graph database, Faming Gong et al. [4] proposed a domain ontology process based on Neo4j graph database to overcome the performance bottleneck encountered in storing data and searching information when processing large amounts of data. Konno et al. [5] described a two-layer knowledge graph database: a concept layer and an instance layer. The concept layer is the ontology model, and the instance layer is the instance data corresponding to the ontology model. The two-layer method is applied to retail business transaction data for business information query and reasoning. Eva

Maulina Aritonang et al. [6] designed an ontology-based semantic network for customs regulations to provide convenience to manage, users to search and track of the history of changes, and the relationship between the rules used by the organization.

Existing research work has found that there are still obvious shortcomings in the extraction and analysis of CVE vulnerability related information. At the same time, the storage and analysis of ontology models based on the knowledge graph have achieved good results. In response to this situation, this paper proposes a knowledge graph based the security vulnerability analysis method of CVE uses the CVE security vulnerability database information as the original data. After ontology modeling, the CVE knowledge graph is obtained. The knowledge graph is used to analyze the node attributes and association relationships in detail and display the results in a visual form, effectively Unearth the intrinsic value of CVE vulnerability data.

2 BACKGROUND

This section describes the security vulnerability library and knowledge graph that our approach relies on.

2.1 Security Vulnerability Library

The existing vulnerability databases mainly include CVE, National Vulnerability Database (NVD), SecurityFocus, China National Vulnerability Database of Information Security (CNNVD), China National Vulnerability Database (CNVD), WooYun, etc., among which the most famous is NVD [7]. NVD is supported by the U.S. government. The data sets it manages to include CVE [8], CWE [9], and CVSS [10]. The CVE data set is collected from the CVE vulnerability database, and thorough analysis and processing, it is associated with CWE, CVSS, etc. to form a more complete CVE security vulnerability library, that is, the CVE security vulnerability library selected in this article, contains various information security weaknesses and vulnerabilities. Its data source is authoritative, statistical fields are rich, and data samples are extensive.

2.2 Knowledge Graph

In order to improve the quality of answers returned by search engines and the efficiency of user queries, Google proposed the concept of knowledge graphs in 2012 [11]. The knowledge graph is based on data sets, through analysis and processing, forming a network of relationships between data and displaying them through visualization. It is essentially a structured network with rich semantic relationships. In recent years, with the rapid development of artificial intelligence, knowledge graphs have also achieved considerable development and produced many excellent results [12-15].

The advantages of the knowledge graph are mainly highlighted in the search and visual display of knowledge. The attributes and association relationships in the knowledge graph can be analyzed through the analysis of the invisible needs of users and help guide decision-making.

There are many ways to store knowledge graphs. In view of the fact that graph databases are more conducive to query and storage, this article selects the Neo4j graph database as the storage method. Neo4j graph database [16] has the characteristics of a friendly interface, use of declarative query language Cypher, support for ACID transactions, high performance and open-source code, and has gradually become one of the most popular graph databases.

3 APPROACH

The construction process of the CVE knowledge graph proposed in this paper is shown in Figure 1. First, obtain the CVE security vulnerability database data from the NVD official website from 1999 to present as the original data for this experiment; secondly, perform ontology modeling of the data, that is, the modeling of nodes, relationships, and attributes, and obtain the CVE security vulnerability ontology model; finally based on this the ontology model extracts the acquired original data and imports it into the Neo4j graph database to complete the construction of the CVE knowledge graph. Each part will be explained in detail below.

3.1 Raw Data Acquisition

Analyzing the method of pulling data from the CVE security vulnerability database on the NVD official website, this article chooses to pull gz files in JSON format, including CVE-2002 to CVE-2020. After analyzing the data pull method, use the data pull script written in Python to obtain the original data of the CVE security vulnerability library. The steps are as follows.

Step 1: Determine the URL for pulling data: https://nvd.nist.gov/feeds/json/cve/1.0/nvdcve-1.0-{}.json.gz, where {} is used to format the year, Then pull the data year by year.

Step 2: Use the gzip library in Python to extract the pulled data to obtain the original data for each year.

3.2 Ontology Modeling

The concept of ontology originally originated from the field of philosophy and is a study of the exploration of existence and its relationship. In information and computer science, ontology is a method of defining concepts, attributes, and relationships in a specified field. In this paper, the ontology modeling of the CVE security vulnerability library is carried out to provide a template for the construction of the knowledge graph.

Figure 2 shows the CVE vulnerability sample entry (CVE-2014-0001) in JSON format. For each CVE vulnerability entry, it mainly contains the following fields: cve (main content of CVE), impact (CVSS2, CVSS3 score), publishedDate (public date), the cve field also contains the following fields: CVE_data_meta (which contains CVE ID number), affects (affected manufacturer, product name and product version), problemtype (associated CWE), references (source of data), description (main description of CVE), etc.

Through the analysis of the original data of security vulnerabilities, the modeling of the nodes, and their relationships in the CVE security vulnerability library is completed. In this experiment, nine types of nodes and eight relationships were extracted from the CVE security vulnerability library, and a CVE security vulnerability ontology model was constructed, as shown in Figure 3. The node model and relational model will be explained separately below.

3.2.1 Node Model. The nine types of nodes extracted from the CVE security vulnerability database are: CVE (vulnerable node type), CVSS2 (CVSS2 scoring node type), CVSS3 (CVSS3 scoring node type), AttackVector (attack vector node type), Vendor (vendor node type), Product (product node type), ProductVersion (product version

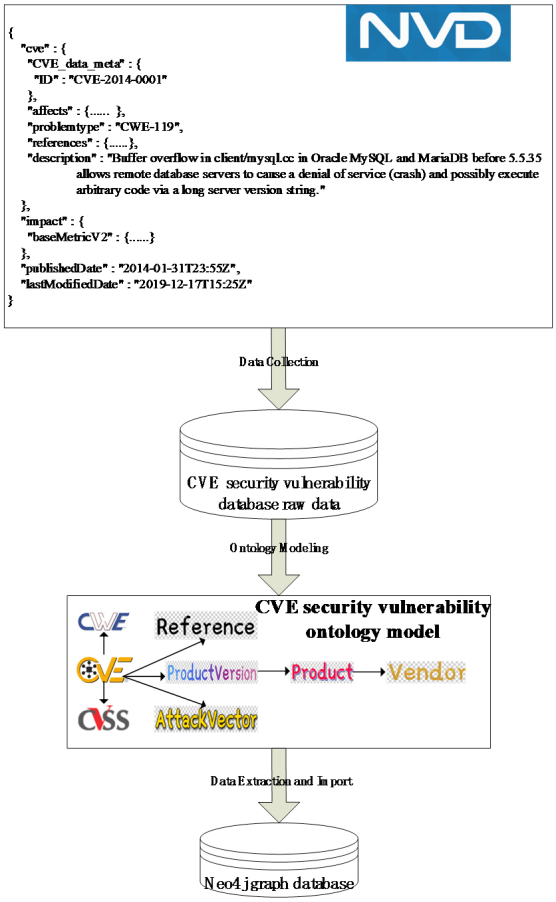

Figure 1: CVE knowledge graph construction process


```
{
  "cve" : {
    "CVE_data_meta" : {
      "ID" : "CVE-2014-0001"
    },
    "affects" : {..... },
    "problemtype" : "CWE-119",
    "references" : {.....},
    "description" : "Buffer overflow in client/mysql.cc in Oracle MySQL and MariaDB before 5.5.35
                     allows remote database servers to cause a denial of service (crash) and possibly execute
                     arbitrary code via a long server version string."
  },
  "impact" : {
    "baseMetricV2" : {.....}
  },
  "publishedDate" : "2014-01-31T23:55Z",
  "lastModifiedDate" : "2019-12-17T15:25Z"
}
```


Figure 2: CVE vulnerability sample entry in JSON format (CVE-2014-0001)

node type), CWE (weakness category node type) and Reference (reference node type), the node and its attribute details are shown in Table 1

3.2.2 Relational Model. The eight types of relationships extracted from the CVE security vulnerability database are: CVE and CWE, CVE and CVSS2, CVE and CVSS3, CVE and AttackVector, CVE and Reference, CVE and ProductVersion, ProductVersion and Product, Product and Vendor, the relationship between nodes and their attributes The detailed information is shown in Table 2, where the AFFECTS relationship indicates the product version affected by CVE, the REFERENCE relationship indicates the reference of CVE data, the SCORED relationship indicates the CVSS score of the CVE, and the ATTACKABLE_THROUGH relationship indicates the

Figure 3: CVE security vulnerability ontology model

Table 1: Node model

Node Type	Main Attributes
CVE	name,description,published
CVSS2	name, base_score, exploitability_score, impact_score, access_vector, ...
CVSS3	name, base_score, exploitability_score, impact_score, access_vector, ...
AttackVector	name
Vendor	name
Product	name
ProductVersion	name, version_value
CWE	name
Reference	name, url, source

Table 2: Relational model

Source Node Type	Relationship Types and Attributes	Target Node Type
CVE	AFFECTS	ProductVersion
CVE	REFERENCE	Reference
CVE	SCORED	CVSS2
CVE	SCORED	CVSS3
CVE	ATTACKABLE_THROUGH{cvss_version:2/3}	AttackVector
CVE	PROBLEM_TYPE	CWE
ProductVersion	VERSION_OF	Product
Product	MADE_BY	Vendor

attack vector used by the CVE (the cvss version attribute indicates the CVSS version), PROBLEM_TYPE indicates the CWE type to which the CVE belongs, the VERSION OF relationship indicates the product corresponding to the product version, and MADE_BY indicates the manufacturer to which the product belongs.

3.3 Data Extraction And Import

The following is to analyze the pros and cons of the three main methods of importing data into the Neo4j graph database, to select the appropriate method as the data import method of this article.

(1) Apoc plug-in method

The plugin is officially provided by Neo4j. It needs to be downloaded from GitHub and placed in the plugin folder of the Neo4j project. Then add the corresponding configuration in the neo4j.conf file and restart the database before it can be used. Users can conveniently use the function interface provided by the plug-in to query and add, for example, you can use the apoc.load.json() function to read JSON files, and then use the Cypher language to directly manipulate the JSON data in the memory. It has the advantages of less code, no need to provide the foreign key information required by the relationship when storing, fewer cycle times, and data insertion without stopping the machine.

(2) Python library method, such as py2neo

Py2neo is a library file for Python operating Neo4j database. The graph can be used to create database connections, Node is used to create nodes, Relationship is used to create relationships, and create is used to complete data insertion. If you use the Python library for data storage, you need to read the JSON file by yourself, and you need to call the database connection to send and execute

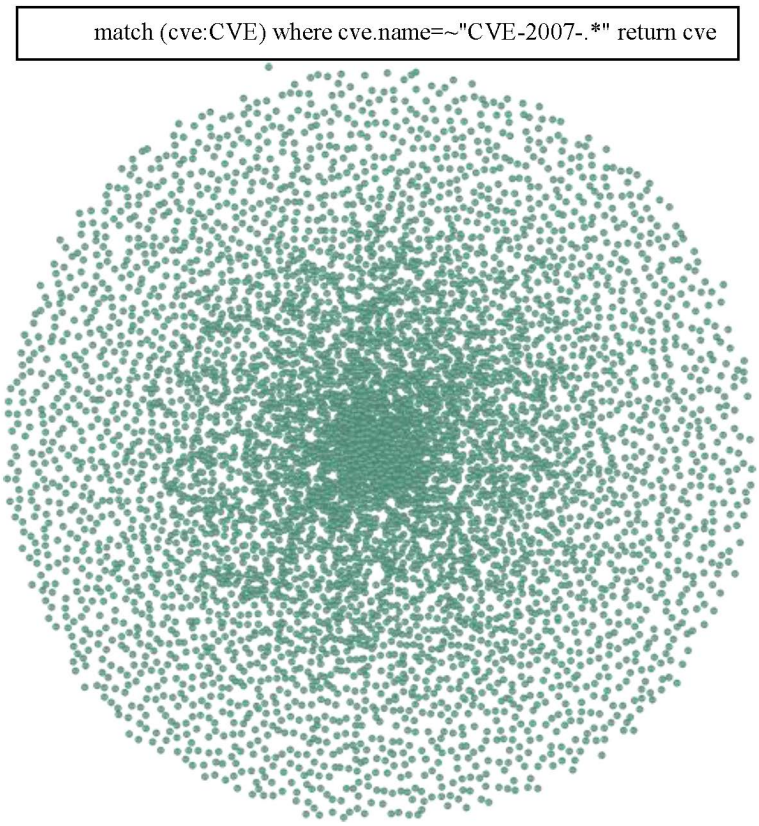

Figure 4: CVE vulnerability in 2007

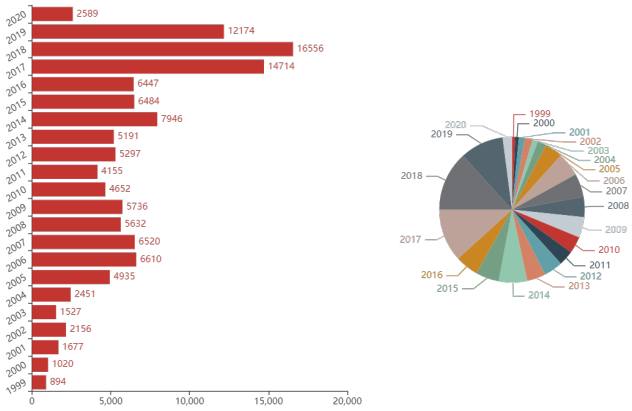

Figure 5: Statistical analysis of CVE vulnerability time dimension

commands every time you insert data. It has the advantages of clear code and convenient interface, but it needs to install the Python environment and corresponding The library and the intermediate links need to handle the affairs themselves.

(3) The neo4j-import method

Neo4j-import is an official data import tool. First, you need to write a script to store the data in the CVS file, and then use the corresponding syntax to insert the data. It has the advantages of fast speed and fewer resources, but it can only be inserted Operate

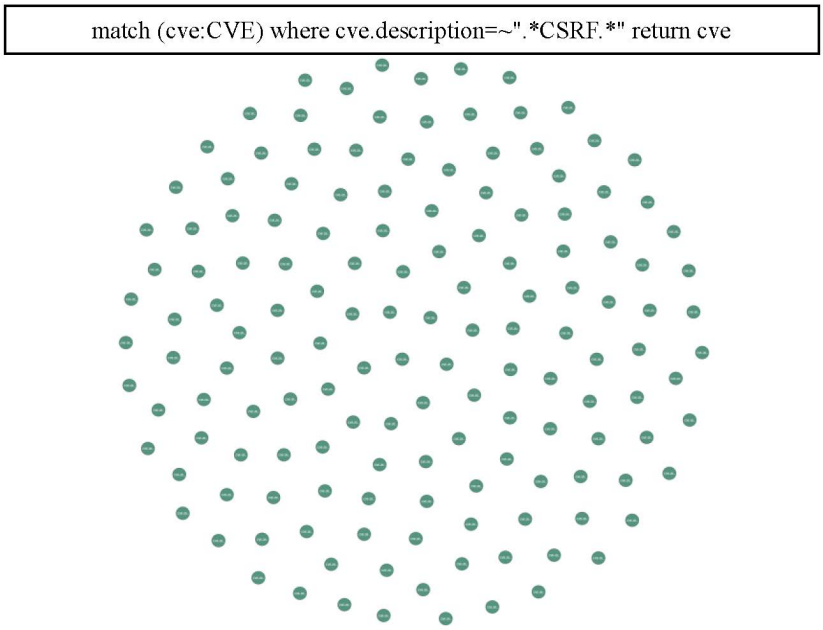

Figure 6: CVE vulnerability due to CSRF

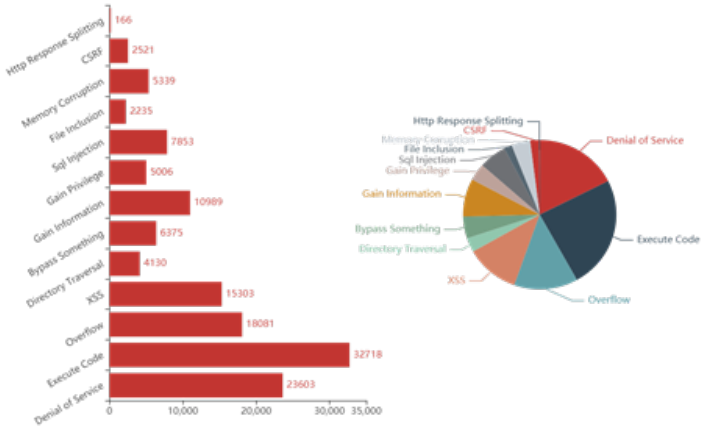

Figure 7: Dimensional statistical analysis of the causes of CVE vulnerabilities

the new database and the database needs to be stopped during the insertion process.

Taking into account the versatility, convenience, and efficiency of inserting data, the Apoc plug-in method is finally used to import data.

In summary, based on the node model and the relationship model, this article traverses the experimental data and completes the import of relationships and attributes at the same time during the process of node import. The algorithm is shown below.

4 EXPERIMENT ANALYSIS

After completing the construction of the CVE knowledge graph, you can further use the Web management tools provided by Neo4j to complete data CRUD and other operations through the Cypher query language. This section analyzes the CVE knowledge graph from the attributes of the CVE node itself and the relationship between these nodes, and then excavates some of the deep-level valuable information.

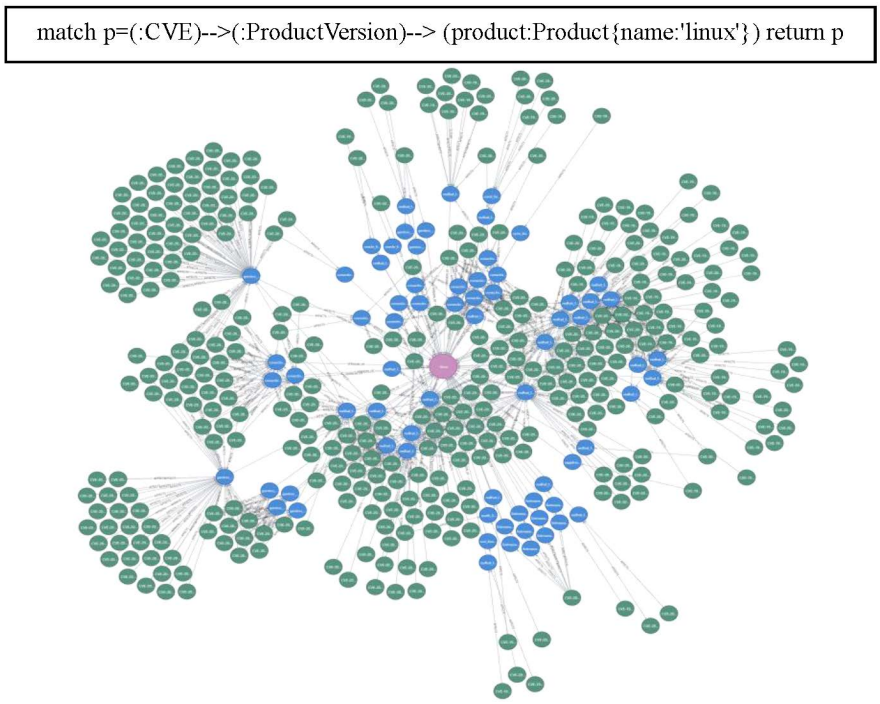

Figure 8: Analysis of linux product vulnerabilities

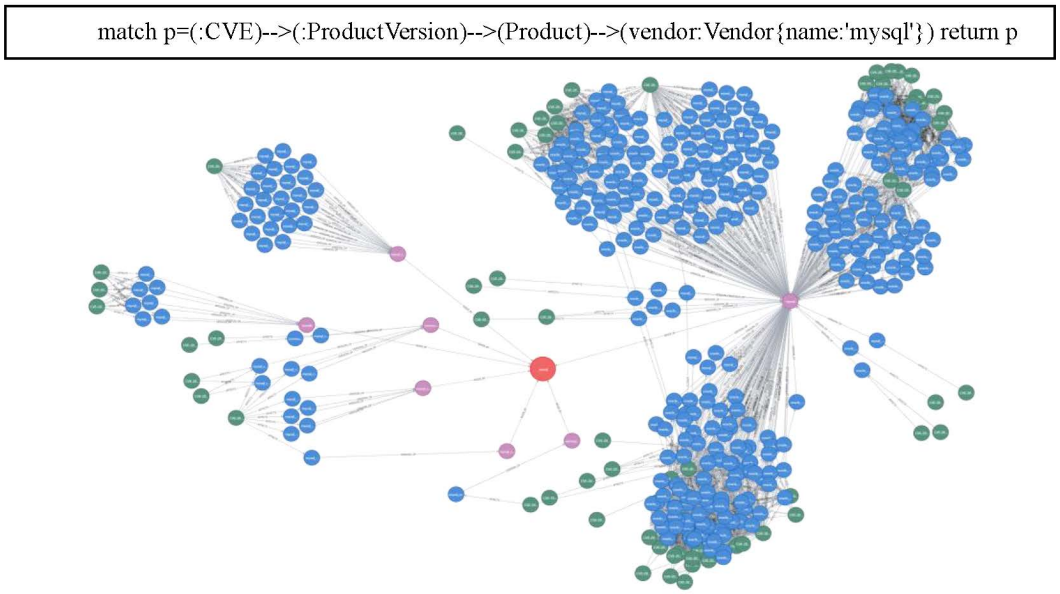

Figure 9: Vulnerability analysis of products manufactured by mysql manufacturers

4.1 CVE Node Attribute Analysis

The CVE node includes attributes such as the name, description, and published. In order to have a deeper understanding of the characteristics of the vulnerability, the CVE node attributes can be analyzed by using the established CVE knowledge graph from multiple dimensions such as time and cause. Starting from the time dimension, all the CVE vulnerabilities in 2007 were queried, and the query sentence was constructed as follows. The result is shown in Figure 4. The result shows that there are 6520 CVE vulnerabilities in 2007. At the same time, the CVE knowledge graph constructed in this article can be used to statistically analyze the number of

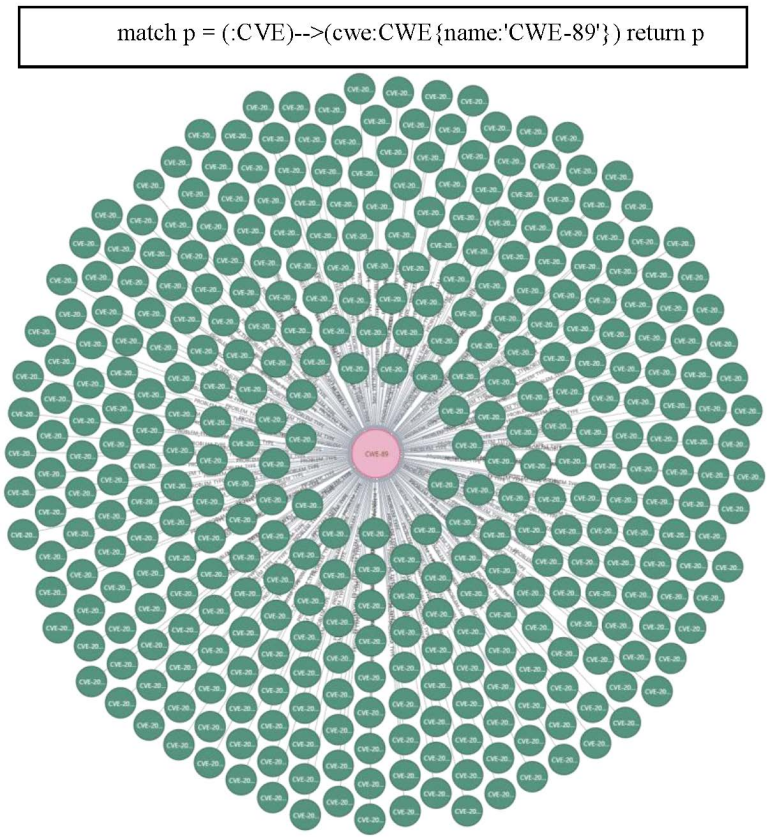

Figure 10: CVE vulnerabilities belonging to the CWE-89 category

Algorithm 1 Data extraction and import algorithm

Require: nvd is CVE for all years
for i=2002; i<=2020; i++ do
vuln ← nvd[i]
corresponding attributes of cve ←vuln directly related to CVE attributes
Store cve as a CVE type node in the database
corresponding attributes of cvss2 ←vuln directly related to CVSS2 attributes
Store cvss2 as a CVSS2 type node in the database. Create a SCORED relationship from cve to cvss2
corresponding attributes of cvss3 ←vuln directly related to CVSS3 attributes
Store cvss3 as a CVSS3 type node in the database. Create a SCORED relationship from cve to cvss3
attack_vector.name ←cvss3.attackVector
Save attack_vector as an Attack Vector type node in the database, create an ATTACK_ABLE_THROUGH relationship from cve to attack_vector, including the attribute cvss_version is 3.
The same is true for other node types, and the relationship type is based on the relational model
end for

CVE vulnerabilities each year. The statistical results are shown in Figure 5

It can be concluded that the number of CVE vulnerabilities is increasing day by day. In 2018, the number of CVE vulnerabilities is the largest, of which 2020 The number of vulnerabilities only includes part, so the number is small. By analyzing the time dimension, the trend of the number of CVE vulnerabilities over time can be obtained intuitively and efficiently

Starting from the dimension of the cause, the number of vulnerabilities caused by Cross-site request forgery (CSRF) from 1999 to 2020 is counted. The query sentence is constructed as follows. The result is shown in Figure 6. The statistical results are shown in Figure 7.

The results show that from 1999 to 2020, there were 2521 CVE vulnerabilities due to CSRF. Besides, the
categories of causes include denial of service, execution code, overflow, Cross-Site Scripting (XSS), directory traversal, bypassing something, gain information, gain privileges, SQL injection, file inclusion, memory corruption and Http response splitting, and the number of vulnerabilities is statistically analyzed for all-cause categories.

Through the analysis of the dimension of the causes, it can be concluded that the number of vulnerabilities caused by the execution code is the largest, which can guide the security analyst to

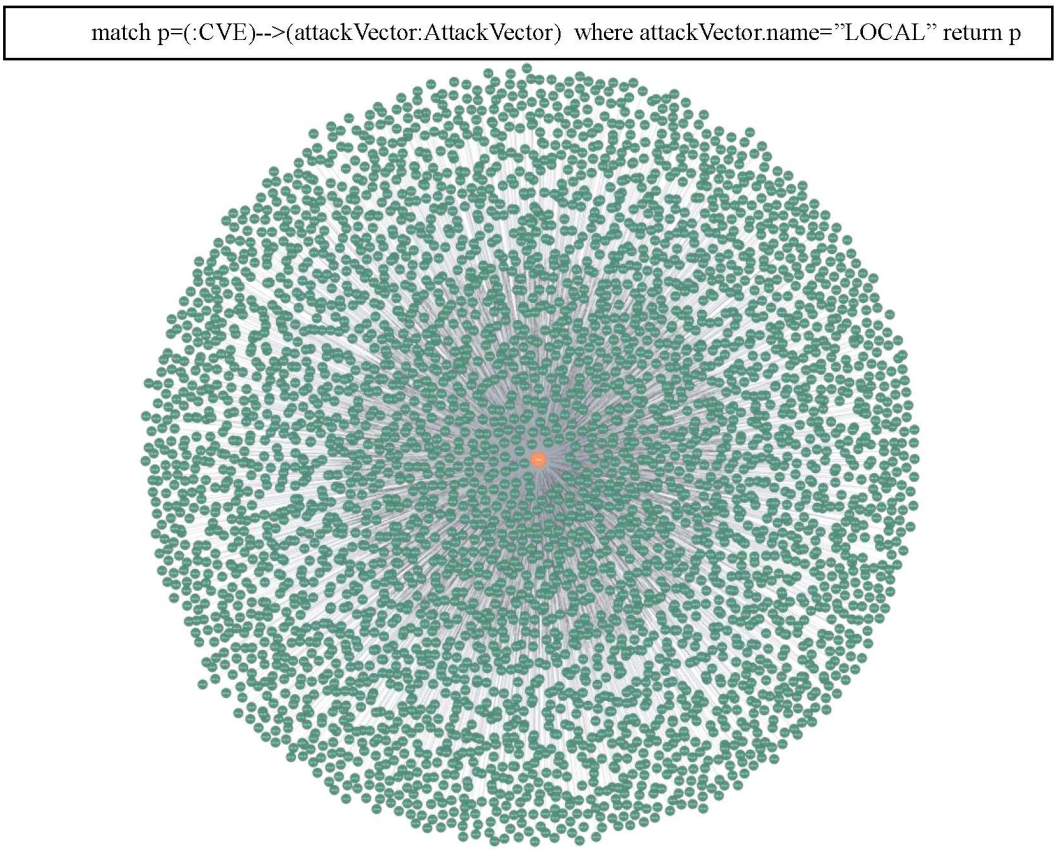

Figure 11: CVE vulnerability with LOACL attack vector

strengthen the investigation of the security vulnerabilities caused by the execution code.

4.2 Relationship Analysis

The analysis of the node's own attributes is not enough to show the power of the knowledge graph. In this section, the actual value of the knowledge graph is further explored by analyzing some of the association relationships and the visual display of the results.

4.2.1 CVE-ProductVersion-Product relationship analysis. The CVE-ProductVersion-Product relationship describes the relationship between CVE vulnerabilities, product versions, and products, and describes the existence of a CVE vulnerability information in a specific version of a product. Take the Linux product as an example, the structure of the query sentence is as follows, the result is shown in Figure 8. It can be found that there are 83 versions of Linux products that have vulnerabilities, and the total number of vulnerabilities is 433. Purple represents the Product node, blue represents the ProductVersion node and Green represents CVE nodes. This analysis can be used to evaluate the safety of the product. The analysis of this relationship can be used to guide product users whether there are vulnerabilities in the current product version, which is conducive to timely discovery, tracking and repair of vulnerabilities.

4.2.2 CVE-ProductVersion-Product-Vendor relationship analysis. The CVE-ProductVersion-Product-Vendor relationship records the existence of CVE vulnerability information in the specified version number of a certain manufacturer's product. The analysis of this relationship can also guide consumers to determine the safety of the products manufactured by the vendor, and at the same time, the vendor can analyze the existence of vulnerabilities in the product, so that the vendor can fix the product vulnerabilities in a targeted manner. Take the MySQL manufacturer as an example, the query statement is constructed as follows, and the result is shown in Figure 9. It can be found that the MySQL vendor has 7 products, 359 versions of the 7 products have vulnerabilities, and the total number of vulnerabilities is 67. The red represents the Vendor node, other colors are the same as above.

4.2.3 CVE-CWE relationship analysis. The CVE-CWE relationship records that a certain CVE vulnerability belongs to a certain CWE. Through the analysis of the relationship, it can be concluded that the CVE vulnerabilities of a certain vulnerability category are. Take CWE89 as an example. The weakness category represented by it is SQL injection. The query statement is constructed as follows. The result is shown in Figure 10. There are 359 CVE vulnerabilities whose weakness category is SQL injection. The pink color represents the CWE node. Green represents CVE nodes.

```
match p=(:CVE)-->(cvss3:CVSS3) where
cvss3.exploitability_score <4 return p
```

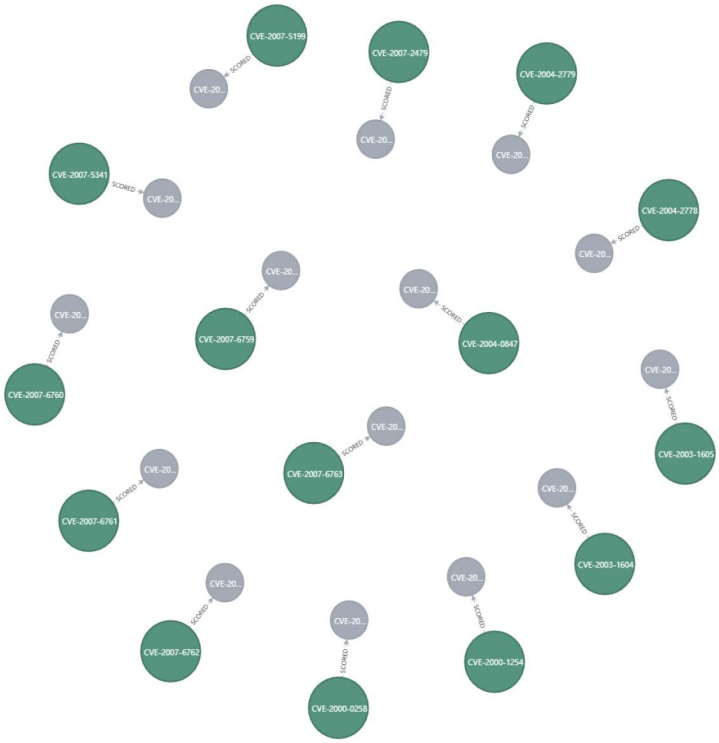

Figure 12: Vulnerabilities in CVSS3 with an exploit score less than 4 (low risk)

```
match p=(:CVE)-->(:=Reference) where cve.name=" CVE-2007-0053" return p
```

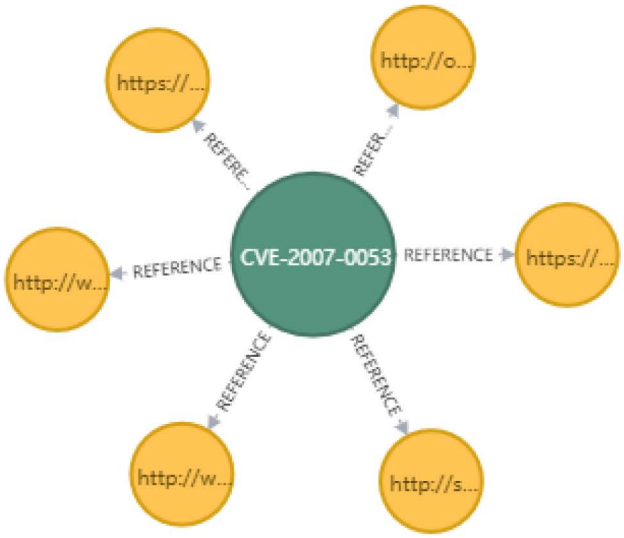

Figure 13: CVE-2007-0053 Vulnerability Reference

4.2.4 CVE-AttackVector relationship analysis. The CVE-AttackVector relationship records the means by which CVE vulnerabilities are attacked, and can guide security personnel to mitigate vulnerabilities. Taking the attack vector as LOACL as an example, the query statement is constructed as follows, and the result is shown in Figure 11, where orange represents the AttackVector node, and the green represents the CVE node.

4.2.5 CVE-CVSS2, CVE-CVSS3 relationship analysis. The relationship between CVE-CVSS2 and CVE-CVSS3 records the CVSS score of CVE vulnerabilities. Manufacturers can judge the danger of CVE vulnerabilities based on the CVSS score, which is helpful for the timely repair of high-risk vulnerabilities. Taking the vulnerability exploit score of less than 4 (low risk) in CVSS3 as an example, the query statement is as follows, and the result is shown in Figure 12, where gray represents the CVSS3 node and green represents the CVE node.

4.2.6 CVE-Reference relationship analysis. The CVE-Reference relationship records the sources of CVE vulnerability data and references to vulnerability descriptions, etc. You can learn more about the details of CVE vulnerabilities by querying Reference. Take the CVE-20070053 vulnerability as an example. The result is shown in Figure 13. The dark orange represents the Reference node and the green represents the CVE node

5 CONCLUSION

This paper uses the CVE security vulnerability database as the original data, through the acquisition of original data, ontology modeling, and data extraction and import, the construction of the CVE security vulnerability knowledge graph based on the Neo4j graph database is completed, and the association between the CVE security vulnerability database data is solved Problems such as low performance, abundant data but low utilization rate and insufficient visualization. At the same time, the detailed analysis of the CVE knowledge graph was carried out from the dimension of the cause of the vulnerabilities, the time dimension, and the correlation dimension, and the results were displayed in a visualized form, and deep valuable information was discovered. Therefore, the method proposed in this paper has good feasibility and has the advantages of higher visualization, stronger relevance, and easier analysis compared with the existing analysis methods.

The follow-up work can be carried out from the following two aspects: on the one hand, increase the relationship between CVE vulnerabilities and enrich the node content; on the other hand, the knowledge graph can be encapsulated, which is conducive to the further utilization of the CVE knowledge graph.

REFERENCES

[1] Guo, M.; Wang, J.A. An ontology-based approach to model common vulnerabilities and exposures in information security. ASEE Southest Section Conference, 2009.

[2] Qin, S.; Chow, K. Automatic Analysis and Reasoning Based on Vulnerability Knowledge Graph. In CyberspaceData and Intelligence, and Cyber-Living, Syndrome,and Health; Springer, 2019; pp. 3–19.

[3] Mittal, S.; Joshi, A.; Finin, T. Thinking, fast and slow: Combining vector spaces and knowledge graphs. arXivpreprintarXiv:1708.03310 2017.

[4] Gong, F.; Ma, Y.; Gong, W.; Li, X.; Li, C.; Yuan, X. Neo4j graph database realizes efficient storage performance of oilfield ontology. PloS one 2018, 13, e0207595.

[5] Konno, T.; Huang, R.; Ban, T.; Huang, C. Goods recommendation based on retail knowledge in a Neo4j graph database combined with an inference mechanism implemented in jess. 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). IEEE, 2017, pp. 1–8.

[6] Aritonang, E.M.; Seminar, K.B.; Wahjuni, S.; Purbo, O.W.; others. Modelling Ontology and Semantic Network of Regulation in Customs and Excise. TELKOMNIKA Telecommunication Computing Electronics andControl 2017, 15, 1934–1942.

[7] Booth, H.; Rike, D.; Witte, G. The national vulnerability database (nvd): Overview. Technical report, National Institute of Standards and Technology, 2013.

[8] Özkan, S. Cve details. Retrieved 2017, 16, 2017.

[9] Martin, R.A. Common weakness enumeration. MitreCorporation 2007.

[10] Mell, P.; Scarfone, K.; Romanosky, S. A complete guide to the common vulnerability scoring system version 2.0. Published by FIRST-forum of incident response and security teams, 2007, Vol. 1, p. 23.

[11] Singhal, A. Introducing the knowledge graph: things, not strings. Official google blog 2012, 5.

[12] Jia, Y.; Qi, Y.; Shang, H.; Jiang, R.; Li, A. A practical approach to constructing a knowledge graph for cybersecurity. Engineering 2018, 4, 53–60.

[13] Asamoah, C.; Tao, L.; Gai, K.; Jiang, N. Powering filtration process of cyber security ecosystem using knowledge graph. 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud). IEEE, 2016, pp. 240–246.

[14] Iannacone, M.; Bohn, S.; Nakamura, G.; Gerth, J.; Huffer, K.; Bridges, R.; Ferragut, E.; Goodall, J. Developing an ontology for cyber security knowledge graphs. Proceedings of the 10th Annual Cyber and Information Security Research Conference, 2015, pp. 1–4.

[15] Noel, S.; Harley, E.; Tam, K.H.; Limiero, M.; Share, M. CyGraph: graph-based analytics and visualization for cybersecurity. In Handbook of Statistics; Elsevier, 2016; Vol. 35, pp. 117–167.

[16] Miller,J.J.Graphdatabaseapplicationsandconceptswith Neo4j. Proceedingsofthe-SouthernAssociation for Information Systems Conference, Atlanta, GA, USA, 2013, Vol. 2324. 17. Bowman, M.; Debray, S.K.; Peterson, L.L. Reasoning about naming systems. ACM Transactions on ProgrammingLanguages and Systems (TOPLAS) 1993, 15, 795–825.