Advances in Artificial Intelligence, Big Data and Algorithms
G. Grigoras and P. Lorenz (Eds.)

doi:10.3233/FAIA230788

Intelligent Association of CVE Vulnerabilities Based on Chain Reasoning

Xin SUN[1], Zixiang WANG
State Grid Zhejiang Electric Power Co., Ltd. Research Institute, Hangzhou, China

Abstract. Vulnerability database data sources use semi-structured and unstructured data expression, which can provide convenience for researchers in vulnerability analysis and systematic study of vulnerability mechanism and vulnerability content. However, the traditional vulnerability database has some problems, such as weak correlation attribute, redundant data format and low visualization degree, which are difficult to be understood and analysed by machines. This paper proposes a CVE vulnerability intelligent association based on chain reasoning. By analysing the vulnerability description and the relationship between different vulnerability databases, investigating the software and systems affected by each vulnerability, linking relevant knowledge, inferring the hidden relationship of each vulnerability, building the relationship between the entity nodes of the vulnerability knowledge graph, and finally importing the data into the neo4j diagram database to build the vulnerability knowledge graph. By calculating the number of software associated with vulnerabilities, the CWE chain relationship is deduced, and the vulnerability knowledge graph constructed is used to conduct preliminary inference on the vulnerability scanning results, so as to obtain the hidden relationship between vulnerabilities and optimize the vulnerability scanning results.

Keywords. CVE vulnerability, chain reasoning, knowledge graph, neo4j

1. Introduction

With the development of the Internet, in the process of software development, whether it is due to the lack of security coding awareness of programmers or the weakness of the programming language itself, security problems are likely to occur, and the emergence of software security vulnerabilities is almost inevitable [1]. Exploitation and attack methods are also developing rapidly [2]. For network security researchers, obtaining high-quality vulnerability information and data is of great value [3-4]. A large amount of existing vulnerability information is provided by vulnerability databases, open-source communities, security blogs, forums, etc. Although this can provide convenience for network security researchers to conduct vulnerability analysis and systematically study vulnerability mechanisms and vulnerability content, each vulnerability database is managed in a decentralized manner [5]. It is relatively difficult to integrate different vulnerability information [6].

Therefore, how to accurately, timely and effectively extract security vulnerabilities in software has become an urgent problem. At present, with the deepening of artificial intelligence, machine learning and graph theory, more and more researchers are trying

[1] Corresponding author: Xin SUN, State Grid Zhejiang Electric Power Co., Ltd. Research Institute; e-mail: 86503739@qq.com

to explore potential vulnerabilities through new technologies or methods [7], such as functional code similarity detection, vulnerability detection based on deep learning, and using code knowledge graph to locate vulnerabilities [8]. These new ideas and methods provide a direction for solving the problem of vulnerability mining technology, and make vulnerability mining technology more automatic and intelligent [9].

Knowledge graph is essentially a knowledge base called semantic network, that is a knowledge base with a directed graph structure, where the nodes of the graph represent entities or concepts, and the edges of the graph represent various semantic relationships between entities/concepts [10]. The concept of knowledge map can be combined with network security knowledge to build vulnerability knowledge base [11]. In this way, we can not only manage vulnerability data from the perspective of graph, but also use knowledge reasoning to discover hidden information. Due to the rapid update rate of the new version of the system and software, the following new vulnerabilities will also be discovered soon. CVE and CNNVD will add a large number of CVE numbers and CNNVD numbers every day. The original vulnerabilities retained by some companies may be released in the future. Some of the included vulnerabilities may have new affected targets and new repair plans. The real-time update of vulnerability database makes the knowledge graph need to acquire and integrate new knowledge in real time.

Foreign countries have carried out research on knowledge graphs for a long time. In order to facilitate the calculation of knowledge, the mainstream method of constructing knowledge graphs is to convert entities and relationships into vectors [12]. Entity relation extraction is an important part of information extraction and information retrieval. With the emergence of deep learning, some scholars have applied deep learning to entity extraction [13]. Although the domestic knowledge graph is still at a certain stage of development, there are also certain research results. In 2014, Chinese enterprises led by Baidu gradually entered the wave of knowledge graph. Since its official release, Baidu Knowledge Graph has more than billions of entities, which can gain a unified understanding of knowledge and solve problems in complex environments, and has been applied to many industries [14].

2. Vulnerability Sample Data Collection Method

This paper studies the collection of multi-source vulnerability data, mainly including the following foreign databases CVE [15], CWE [16], CAPEC [17], vulnerability disclosure websites CXSECURITY, Exploit-DB, PacketStorm, etc., and domestic databases, mainly CNNVD and CNVD [18].

Write python scripts to obtain vulnerability information in csv files of CVE official website, preliminarily screen CVE by character matching, generate web links to be crawled by using available CVE numbers, write intelligent crawler, generate links of CNVD, CNNVD, Exploit-DB [19] and other web pages [20] by algorithm. The data content of the corresponding database is crawled according to the link, and the important information and relationship related to the vulnerability are collected through the string-matching information extraction, text parsing and other technologies, including the release date of the vulnerability, the availability of the vulnerability, the impact of confidentiality and integrity, and a brief introduction of the vulnerability content, etc.

Take CVE and NVD for example, each vulnerability in the vulnerability database has unique ID. Different vulnerability databases cover different types of information, but basically include vulnerability description, risk level, affected product list, release time

and vulnerability source code, which provides a reliable basis for in-depth research and analysis of vulnerability causes and knowledge reasoning. At present, software vulnerabilities, open-source project information and source code management information exist in different places in different forms, with the following characteristics:

- Huge data scale and rapid growth rate: according to the data conveyed by the vulnerability database security website, the number of vulnerabilities increases rapidly as the number of software increases.
- The types are complex and diverse: common vulnerability types include SQL injection, command injection, cross-site scripting attacks, input validation errors, etc.
- Data dispersion, heterogeneity, and multi-source: For open source software systems, although the source code of vulnerabilities is detectable, it is not easy to obtain the source code of vulnerabilities because of the different storage location and release methods of these source codes and the characteristics of heterogeneity and multi-source.

According to the above vulnerability characteristics, vulnerability data collection methods can be divided into three types: manual, semi-automatic collection, batch import from third-party systems and online scanning with the help of web crawler. Only in terms of collection efficiency, web crawlers and third-party systems are more efficient in bulk import. However, manual or semi-automatic methods have some advantages over the other two methods in terms of flexibility and reliability. Considering the large amount of vulnerability data and required information, the third-party data system may not be able to provide complete and valuable data information. Therefore, the collection of vulnerability data is mainly based on web crawler, supplemented by manual or semi-automatic batch collection.

The basic principle of vulnerability source code web crawler is to obtain all source code file data related to vulnerability under the path by generating the basic information of network path and linking the URL of CVE. The source code file is stored in the local repository. The corresponding CVE-ID, CWE-ID, vulnerability description, release time, and other basic information are stored locally. The framework for web crawler to acquire vulnerability knowledge is shown in Figure 1.

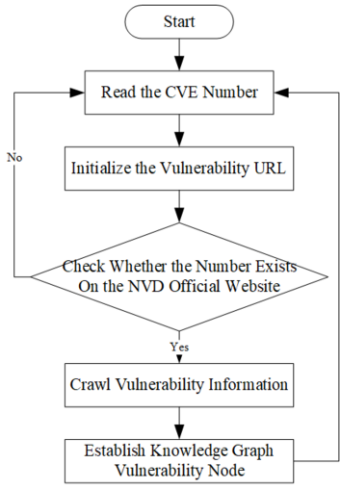

Figure 1. Web Crawler Framework for Collecting Information

Obtain the csv file of the CWE official website. Generate CWE nodes and internal basic relationships based on the information in the csv file. According to the acquired knowledge, links between entity nodes are further established, such as the relationship between each CWE number: parent relationship, child relationship, member relationship. The establishment of these relationships can facilitate the subsequent application of knowledge graph.

3. Construction of Vulnerability Knowledge Graph

In this paper, the construction of vulnerability knowledge graph is studied. Through the python programming mode of neo4j graph database, the above summarized knowledge is used to write scripts to build the vulnerability knowledge graph.

3.1. Design of Vulnerability Knowledge Graph

Vulnerability is the most important issue in cyberspace security. In the design of security knowledge graph, the vulnerability itself should be the center and the core of entity association, and the information of vulnerability attack benefit, function object, attack principle, defense scheme and other information with the vulnerability. According to the principle of taking vulnerabilities as the central point can be divided into the following categories according to the relationship between entities and vulnerabilities:

(1) Vulnerability targets: software, operating system, version and supplier.

(2) Vulnerability, vulnerability type: CVE (including basic vulnerability information), CNNVD, CNVD (mainly mapping with CVE), CWE, and CAPEC.

(3) Vulnerability disclosure information: Supplement basic vulnerability information, including vulnerability type, programming language, POC, EXP.

3.2. Entity Extraction

In this paper, entity extraction, relationship extraction and knowledge fusion are carried out on the data obtained by web crawler successively. The node set and attribute set of the entity are obtained by entity extraction, and the relationship between entities is given by relation extraction. In order to visually display the relationship between nodes, vulnerability information can be stored in batch in the neo4j graph database in two ways: Python calls the py2neo library or uses the LOAD CSV statement of cypher language. Just in terms of importing patterns, the first pattern is easy to use and can be imported in real time, but is relatively slow. The second method is the official import tool, which is relatively fast, but during execution, the CSV file must be inserted into the neo4j/import directory and the default code must be changed to UTF-8. In addition, when there are too many fields, the import instruction needs to translate key characters such as single quotation marks, which is complicated, and nodes and relationships cannot be created dynamically. Therefore, when importing semantic vulnerability information into data, this paper first analyzed the key information in CSV files, then imported the determined data into neo4j database, and then combined with Python to call py2neo library for data batch import. The flowchart is shown in figure 2:

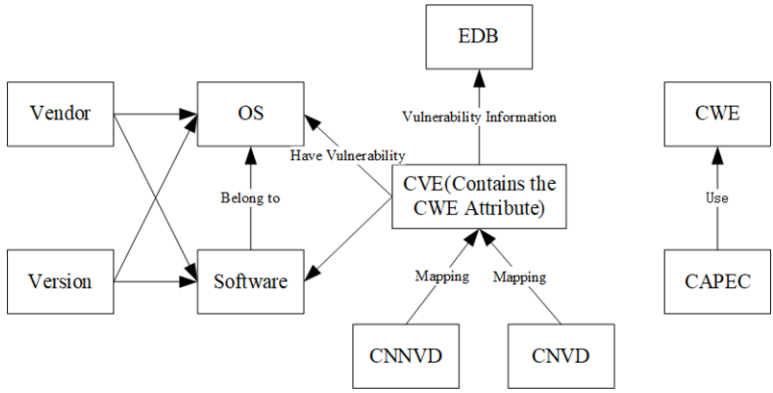

Figure 2. Nodes and Relationships of Vulnerability Knowledge Graph

In CVE website can be downloaded via the link (https://www.cve.org/Downloads/allitems.csv) has been registered in CVE official vulnerability data obtained, including code, description and status. The CVE number is obtained by matching the 'REJECT' and 'RESERVED' strings in the descriptor and discarding them. Vulnerabilities collected on the NVD website can also be classified into different states, including received, waiting to be analysed, analysing, analysed, and repaired. Different states may lead to different pages when crawling.

3.3. Relationship Construction

The relationship between CAPEC and CWE is that of "tactics and implementers", that is, "the attack behaviour represented by a certain attack type acts on a certain vulnerability of a certain component". The official definitions of CWE (Enumeration of software vulnerability Types) and CAPEC (enumeration and classification of attack types) include the relationship between them and within CWE and CAPEC, which can be directly used to build the knowledge graph. Both CAPEC and CWE also have relationships between different entities in their standards. CAPEC's description of the attack type relationship is stored in the standard "Related_Attack_Patterns" field, and CWE's description of the vulnerability relationship is stored in this standard.

4. CWE Chain Reasoning

Because there are abundant associations among entities in the knowledge graph, hidden knowledge can be deduced from it. CWE's description of vulnerability enumeration types and its own association are rich in vulnerability information, including vulnerability types, affected products and the relationship between software vulnerabilities. A CWE chain is a sequence of two or more independent vulnerability types that can be closely linked together in software. By counting the number of CWE chain associations and reasoning, the potential rules and knowledge can be obtained, and then the knowledge graph is applied to the detection of compound vulnerabilities.

Inspired by association rules, CWE chain reasoning seeks to gain hidden rules or knowledge through reasoning, which lays the foundation for in-depth study of vulnerabilities. Because a single CWE number is associated with multiple CVE numbers, and because a single CVE has one or more affected products, this knowledge is closely

linked through the CWE chain by affecting multiple CVE, and thus products and software. By analysing from the perspective of association rules, it may be possible to obtain the relationship between the products affected by a particular CVE and CWE. A vulnerability of some type, "X" can directly create the conditions necessary for another type of vulnerability, "Y" to enter a vulnerable state. When this happens, CWE refers to "X" as the cause of "Y" and "Y" as the result of "X". For example, CWE-918 is often referred to as server-side request forgery (SSRF) and CWE-79 is referred to as cross-site scripting attack (XSS). In some cases, cross-site scripting attacks can be performed through server-side forgery requests. In this case, cross-site scripting attacks are necessary to spoof server-side requests.

In general, chain confidence is an important index of CWE chain reasoning. It is used to measure the correlation degree of CWE chain, so as to discover the hidden relationship between CWE chains. In vulnerability knowledge graph, chain confidence is used to calculate the conditional probability that CWE exists in the same product at the same time, as shown in chain confidence calculation Eq. (1).

$$Chain\ Confidence = C_2(\Pr with\ CWE - X \cap \Pr with\ CWE - Y)/C_1(\Pr with\ CWE - X) \quad (1)$$

Where, X and Y represent the number of CWE. $C_1(\Pr with\ CWE - X)$ said is a kind of leak CWE-X contains the number of affected products, $C_2(\Pr with\ CWE - X \cap \Pr with\ CWE - Y)$ contains a CWE said - the number of X and CWE - Y products.

5. Conclusion

In this paper, knowledge extraction from scattered and complex vulnerability information, construction of vulnerability knowledge map, excavation of vulnerability correlation, vulnerability prediction, the main work is divided into two parts, the construction of vulnerability knowledge map and CWE chain reasoning. In terms of construction, crawler technology is used to extract vulnerability information such as CVE, CWE, CAPEC, CNVD and CNNVD and vulnerability disclosure information such as utilize-DB to collect and extract knowledge, establish vulnerability entities in knowledge map, and extract dependencies between software.

Acknowledgments

This paper is supported by the science and technology project of State Grid Corporation of China: " Research on Key Technologies for Vulnerability Mining in Intelligent Electric Power IoT Terminals" (Grand No. 5700-202219198A-1-1-ZN).

References

[1] Song, R., Gao, S., Song, Y., & Xiao, B. (2022, July).: A Traceable and Privacy-Preserving Data Exchange Scheme based on Non-Fungible Token and Zero-Knowledge. In 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS) (pp. 224-234). IEEE.

[2] Yang, L., Song, Y., Gao, S., Hu, A., & Xiao, B. (2022). Griffin: Real-time network intrusion detection system via ensemble of autoencoder in SDN. IEEE Transactions on Network and Service Management, 19(3), 2269-2281.
[3] Chen, Q., Song, Y., Jennings, B., Zhang, F., Xiao, B., & Gao, S. (2021, December). IoT-ID: robust IoT device identification based on feature drift adaptation. In 2021 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.
[4] Song, Y., Chen, B., Wu, T., Zheng, T., Chen, H., & Wang, J. (2021). Enhancing packet-level Wi-Fi device authentication protocol leveraging channel state information. Wireless Communications and Mobile Computing, 2021, 1-12.
[5] Song, Y., Geng, Y., Wang, J., Gao, S., & Shi, W. (2021). Permission Sensitivity-Based Malicious Application Detection for Android. Security and Communication Networks, 2021, 1-12.
[6] Chen, B., Song, Y., Zhu, Z., Gao, S., Wang, J., & Hu, A. (2021, June). Authenticating mobile wireless device through per-packet channel state information. In 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W) (pp. 78-84). IEEE.
[7] Wu, T., Song, Y., Zhang, F., Gao, S., & Chen, B. (2021, June). My site knows where you are: a novel browser fingerprint to track user position. In ICC 2021-IEEE International Conference on Communications (pp. 1-6). IEEE.
[8] Ma, X., Song, Y., Wang, Z., Gao, S., Xiao, B., & Hu, A. (2021, June). You Can Hear But You Cannot Record: Privacy Protection by Jamming Audio Recording. In ICC 2021-IEEE International Conference on Communications (pp. 1-6). IEEE.
[9] Chen, B., Song, Y., Wu, T., Zheng, T., Chen, H., Wang, J., & Li, T. (2021). Enhancing Wi-Fi Device Authentication Protocol Leveraging Channel State Information. In Mobile Multimedia Communications: 14th EAI International Conference, Mobimedia 2021, Virtual Event, July 23-25, 2021, Proceedings 14 (pp. 33-46). Springer International Publishing.
[10] Song, Y., Zhou, K., & Chen, X. (2012). Fake bts attacks of gsm system on software radio platform. Journal of Networks, 7(2), 275.
[11] Song, R., Song, Y., Gao, S., Xiao, B., & Hu, A. (2018, December). I know what you type: Leaking user privacy via novel frequency-based side-channel attacks. In 2018 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.
[12] Song, Y., Huang, Q., Yang, J., Fan, M., Hu, A., & Jiang, Y. (2019, May). IoT device fingerprinting for relieving pressure in the access control. In Proceedings of the ACM Turing Celebration Conference-China (pp. 1-8).
[13] Song, R., Song, Y., Liu, Z., Tan, M., & Zhou, K. (2019). GaiaWorld: a novel blockchain system based on competitive PoS consensus mechanism. CMC-COMPUTERS MATERIALS & CONTINUA, 60(3), 973-987.
[14] Shi, C., Song, R., Qi, X., Song, Y., Xiao, B., & Lu, S. (2020, June). ClickGuard: Exposing hidden click fraud via mobile sensor side-channel analysis. In ICC 2020-2020 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.
[15] Yang, L., Song, Y., Gao, S., Xiao, B., & Hu, A. (2020, December). Griffin: an ensemble of autoencoders for anomaly traffic detection in SDN. In GLOBECOM 2020-2020 IEEE Global Communications Conference (pp. 1-6). IEEE.
[16] Gao, S., Peng, Z., Xiao, B., Hu, A., Song, Y., & Ren, K. (2020). Detection and mitigation of DoS attacks in software defined networks. IEEE/ACM Transactions on Networking, 28(3), 1419-1433.
[17] R. Song, Y. Song, Q. Dong, A. Hu and S. Gao, "WebLogger: Stealing your personal PINs via mobile web application," 2017 9th International Conference on Wireless Communications and Signal Processing (WCSP), Nanjing, China, 2017, pp. 1-6.
[18] Y. Song, X. Zhu, Y. Hong, H. Zhang and H. Tan, "A Mobile Communication Honeypot Observing System," 2012 Fourth International Conference on Multimedia Information Networking and Security, Nanjing, China, 2012, pp. 861-865.
[19] Y. Gala, N. Vanjari, D. Doshi and I. Radhanpurwala, "AI based Techniques for Network-based Intrusion Detection System: A Review," 2023 10th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2023, pp. 1544-1551.
[20] B. Charyyev and M. H. Gunes, "Locality-Sensitive IoT Network Traffic Fingerprinting for Device Identification," in IEEE Internet of Things Journal, vol. 8, no. 3, pp. 1272-1281, 1 Feb.1, 2021.