2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS) | 978-1-6654-7704-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/QRS57517.2022.00012

A Comprehensive Analysis of NVD Concurrency Vulnerabilities

Lili Bo[1,2], Xing Meng[1], Xiaobing Sun[1,*], Jingli Xia[1], and Xiaoxue Wu[1]

[1]Yangzhou University, Yangzhou, Jiangsu, China

[2]Key Laboratory of Safety-Critical Software Ministry of Industry and Information Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, China

lilibo@yzu.edu.cn, MX120200535@yzu.edu.cn, xbsun@ yzu.edu.cn

*corresponding author

***Abstract*—Concurrency vulnerabilities caused by synchronization problems will occur in the execution of multi-threaded programs, and the emergence of concurrency vulnerabilities often cause great threats to the system. Once the concurrency vulnerabilities are exploited, the system will suffer various attacks, seriously affecting its availability, confidentiality and security. In this paper, we extract 839 concurrency vulnerabilities from Common Vulnerabilities and Exposures (CVE), and conduct a comprehensive analysis of the trend, classifications, causes, severity, and impact. Finally, we obtained some findings: 1) From 1999 to 2021, the number of concurrency vulnerabilities disclosures show an overall upward trend. 2) In the distribution of concurrency vulnerability, race condition accounts for the largest proportion. 3) The overall severity of concurrency vulnerabilities is medium risk. 4) The number of concurrency vulnerabilities that can be exploited for local access and network access is almost equal, and nearly half of the concurrency vulnerabilities (377/839) can be accessed remotely. 5) The access complexity of 571 concurrency vulnerabilities is medium, and the number of concurrency vulnerabilities with high or low access complexity is almost equal. The results obtained through the empirical study can provide more support and guidance for research in the field of concurrency vulnerabilities.**

Keywords-Concurrency Vulnerability; CVE; Empirical study

I. Introduction

With the prevalence of multi-core operating systems and the significant improvement of multi-threaded program performance, concurrent programming has become the mainstream of program development. But this also leads to the emergence of concurrency vulnerabilities. The complexity of multi-threaded programs contributes to the uncertainty of thread scheduling, concealment and latency of the concurrency vulnerabilities. Compared with traditional software vulnerabilities, concurrency vulnerabilities are related to both the execution path of program instructions and the state of the software in the time dimension [1][2]. This brings a great challenge to detect and fix concurrency vulnerabilities.

Although there are large numbers of studies on solving software security problems [3][4], the researches on analysing concurrency vulnerabilities are much fewer [5]. The analysis of concurrency vulnerabilities can intuitively reflect the inherent characteristics of concurrency vulnerabilities. Therefore, it is critical to conduct a comprehensive analysis to concurrency vulnerabilities. In this paper, we focus on the concurrency vulnerabilities. After data extraction and cleaning, we analyze the concurrency vulnerabilities from multiple dimensions to obtain some findings.

The rest of the paper is organized as follows. Section 2 describes the main implementation process of constructing the concurrency vulnerability data set. In Section 3, we conduct a comprehensive analysis of concurrency vulnerabilities from multiple dimensions and obtain some valuable findings. Section 4 summarizes the related work and Section 5 concludes this paper.

II. Concurrency Vulnerability Dataset Construction

A. Data Collection

Common Vulnerabilities and Exposures (CVE)[1] consists of abundant vulnerability reports. National Vulnerability Database (NVD)[2] is another vulnerability library where the CVE list is fully synchronized with the vulnerability database. Thus, any updates to the CVE will immediately appear in the NVD. What's more, NVD is based on the information contained in the CVE records and provides additional information for each record, such as repair information, severity score, and impact rating. Since NVD provides richer information for each record and has more analysis dimensions, we use NVD as a direct data source for concurrency vulnerability data collection.

CVSS is the "Common Vulnerability Scoring System", which mainly helps to establish a standard for measuring the severity of vulnerabilities, so people can compare the severity of vulnerabilities to determine the priority of dealing with vulnerabilities. CVSS scores are evaluated on three metrics: base, temporal and environmental, where basescore represents the basic characteristics inherent in the vulnerability, which do not change over time or with the environment. The final CVSS score is a maximum of 10 and a minimum of 0. Based on the base score rating of CVSS

1 http://cve.mitre.org/

2 https://nvd.nist.gov/

V3.0[3], Vulnerabilities with a score between 0 and 3.9 are considered low-level vulnerabilities, those with a score between 4.0 and 6.9 are considered medium-level vulnerabilities, and scores between 7.0 and 10 are high risk, where scores of 9.0 to 10 are considered critical dangerous.

NVD database provides vulnerability data in xml and json formats. In total, we collected more than 180,000 json format vulnerability data in NVD ranging from 1999 to December 2021. The json format file contains CVE-ID, CVSS-v2 and the attributes under CVSS-v3, as well as a total of 63 fields such as publishedDate and lastModifiedDate.

B. Data Extraction

We use the following four steps to extract concurrency vulnerability data from the collected vulnerability data:

Step 1: **Choose the data extraction approach**. From the NVD summary word frequency distribution, we find that the words with high frequency do not contain concurrency vulnerability keywords. Based on this judgment, the concurrency vulnerability data occupies a relatively small proportion of the total vulnerability data. Finding a suitable automatic extraction method is time-consuming. Also, the extraction of vulnerability is incomplete using this automatic extraction method. Since no suitable algorithm can be found to extract concurrency vulnerability data, we use manual data extraction, that is, combining CWE and concurrency vulnerability keywords to extract concurrency vulnerability data.

Step 2: **Manual screening based on CWE**. CWE Version 4.4[4] publishes the class name, detailed description and application of the CWE type number used. For example, CWE-362[5] refers to use shared resources and improper synchronization for concurrent execution (“Race Condition”). Its detailed description is that, “The program contains a code sequence that can run concurrently with other code, and the code sequence requires temporary, exclusive access to a shared resource, but a timing window exists in which the shared resource can be modified by another code sequence that is operating concurrently”. CWE-662[6] refers to improper synchronization (“Improper Synchronization”). Its detailed description is that, “The software utilizes multiple threads or processes to allow temporary access to a shared resource that can only be exclusive to one process at a time, but it does not properly synchronize these actions, which might cause simultaneous accesses of this resource by multiple threads or processes”. CWE-367[7] refers to Time-of-check Time-of-use (TOCTOU) Race Condition(“Timing”). Its detailed description is that, “The software checks the state of a resource before using that resource, but the resource's state can change between the check and the use in a way that invalidates the results of the check. This can cause the software to perform invalid actions when the resource is in an unexpected state”. CWE-667 [8] refers to Improper Locking(“Lock”). Its detailed description is that, “The software does not properly acquire or release a lock on a resource, leading to unexpected resource state changes and behaviors”. From CWE-362, CWE-662, CWE-367 and CWE-667, we can see that CWE can reflect a coarse classification of concurrency vulnerabilities. Thus, we filter the concurrency vulnerability data with CWE. We collect the data which belongs to CWE-362, CWE-662, CWE-367, CWE-667 and finally extracting in total 1149 concurrency vulnerabilities.

Step 3: **Filter data by combining keywords**. Through a preliminary investigation of concurrency vulnerabilities, there is no an explicit categorization of concurrency vulnerabilities [6]. According to the related work [6][7][8][9], we use the concurrency-related keywords to filter out concurrency vulnerabilities from other CWE types to complement our dataset. The keywords follow the following keyword groups: [lock, deadlock, synchronization, thread, multiple-threads, concurrency, concurrent, parallel, atomicity, order violation, race, race condition]. After Step 3, we obtained in total 1180 possible concurrency vulnerabilities.

Step 4: **Filter data according to the vulnerability description**. As our data analysis will try to obtain the reasons and corresponding results from the description, the concurrency vulnerabilities are removed if they cannot be judged from their description. Take CVE-2020-9939[9] for example, its description is “This issue was addressed with improved checks. This issue is fixed in Mac OS Catalina 10.15.6. A local user may be able to load unsigned kernel extensions”. From the description of the CVE-2020-9939, we cannot get the reason and result of the concurrency vulnerability, thus we filter it out. Finally, we extract 839 concurrency vulnerabilities in total.

C. Data Cleaning

CVSS V2 can define and express the basic characteristics of vulnerabilities and the objective method of characterizing vulnerabilities provides users with a clear and intuitive representation of vulnerabilities. The extracted concurrency vulnerabilities contain 63 fields. Since we only study the trends, classifications, causes, and results of concurrency vulnerabilities, not all attributes are worthy of considering, so we finally retained CVE-ID, CWE, Reasons, Results and PublishedDate based on the CVSS V2 standard.

CVSS V2 mainly evaluates the vulnerabilities from three groups: Base Scores, Temporal, and Environmental. Base Scores represents the basic characteristics of the vulnerability and it does not change with time and the environment. Temporal changes with time and while stay unchanged with the environment.Environmental represents the characteristics related to the specific user environment. BaseScore scores the inherent characteristics of the entire vulnerability. The higher the score is, the higher risk the vulnerability has. Access

3 https://www.first.org/cvss/specification-document
4 https://nvd.nist.gov/vuln/categories
5 http://cwe.mitre.org/data/definitions/362.html
6 https://cwe.mitre.org/data/definitions/662.html
7 http://cwe.mitre.org/data/definitions/367.html
8 http://cwe.mitre.org/data/definitions/667.html
9 https://nvd.nist.gov/vuln/detail/CVE-2020-9939

Vector (AV), Access Complexity (AC), Authentication (AU) in the base assess how to access the vulnerability and whether additional conditions are required to exploit it. In addition, the impacts are divided into Confidentiality Impact (C), Integrity Impact (I), Availability Impact (A). As we study the inherent characteristics of concurrency vulnerabilities, we only consider the attributes under base.

Each concurrency vulnerability contains a detailed description, which introduces the location where the vulnerability occurred and the occurrence details, including the cause and the result. We manually analyze the reason as well as the result, and classify them. Some description follows the forms like "x allows remote attackers to y" or similar. Take CVE-2020-35451[10] for example, its description is that, "There is a race condition in OozieSharelibCLI in Apache Oozie before version 5.2.1 which allows a malicious attacker to replace the files in Oozie's sharelib during it's creation". The description can be divided into two parts. The first part is the reason, that is "x: race condition" and the second part is the result, that is "y: replace the files". Following this rule, the reason and result are extracted.

After data cleaning, 12 fields are chosen and extracted for 839 concurrency vulnerabilities. They are CVE-ID, CWE, Severity, BaseScore, Access Vector (AV), Access Complexity (AC), Authentication (AU), Confidentiality Impact (C), Integrity Impact (I), Availability Impact (A), Reason, Result.

III. Concurrency Vulnerability Data Analysis

In this section, we analyzed the data from single and multiple dimensions separately by using Tableau.

(1) The number of concurrency vulnerabilities by time: Figure 1 shows the trend of concurrency vulnerabilities from 1999 to 2021. It can be seen that from January 1999 to December 2021, the number of concurrency vulnerabilities disclosed shows an overall upward trend. This is closely related to the development of multi-core hardware and complex concurrent software. It also indicates the significance of the research on concurrency vulnerabilities.

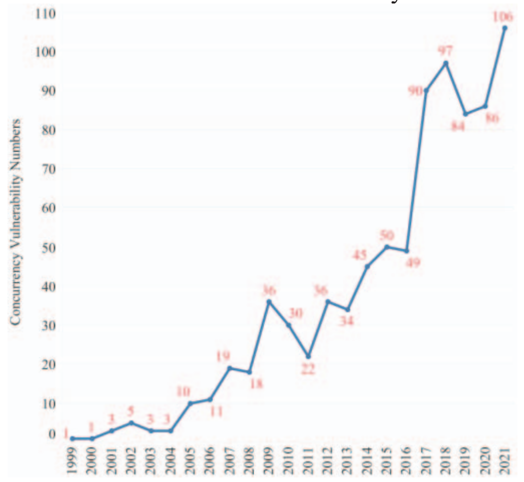


Fig.1 Number of concurrency vulnerabilities from 1999 to 2021

Finding 3.1: With the increase of software scale and software complexity, concurrency vulnerabilities are emerging rapidly, which may severely threaten software security. Thus, it is urgent to propose new technologies to detect, localize and fix concurrency vulnerabilities.

(2) Distribution of CWE concurrency vulnerabilities belong to: Figure 2 shows the distribution of CWE concurrency vulnerabilities belong to. It can be seen that, CWE-362 (Race Condition) accounts for the largest proportion of concurrency vulnerabilities, with 707 entries. At the same time, there are only five entries in CWE-662 (directly indicate improper synchronization). NVD-CVE-Other means that the CWE type number cannot be identified by the description or the CWE represents other vulnerability types, but the keyword look-up is indeed a concurrency vulnerability and the CWE type number is not used in NVD. There are 26 entries belonging to NVD-CVE-Other. This indicates that we cannot recognize concurrency vulnerabilities based on CWE alone. It is better to combine with keywords in the CVE description and the related references to extract them.

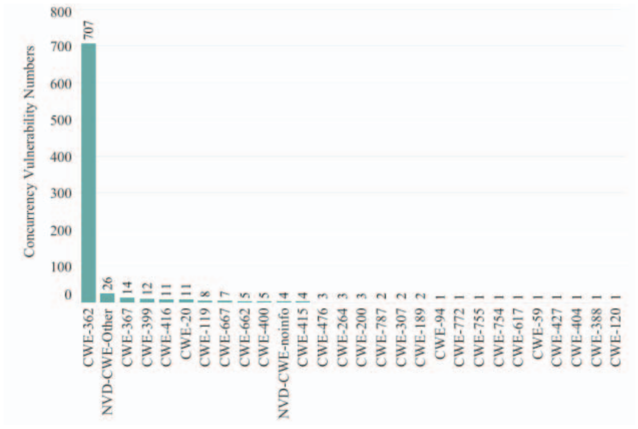


Fig.2 CWE distribution

Furthermore, we use LDA topic modeling on 707 CVE-362 entries to discover the topics. LDA (Latent Dirichlet Allocation) [10] is a common topic generation model that is often used to mine text for potentially implicit topics that summarize the semantic information in the corpus well. Suppose we set the number of topics in the corpus to k, there are a total of n non-repeating words in the corpus. The key idea behind LDA is that it represents each document in the corpus as a probability distribution over k topics, and each topic as a probability distribution over n words. This means that we can describe the entire corpus in terms of a collection of topics, where each document consists of one or more topics and where words in the corpus can occur in one or more topics.

10 https://nvd.nist.gov/vuln/detail/CVE-2020-35451

Before constructing the LDA topic model, we need to pre-processing the data for 707 entries, which includes the following processing:(1) Remove the codes contained in the descriptions; (2) Remove stop words; (3) Normalize the word stem of each word; (4) Drop low-frequency words.

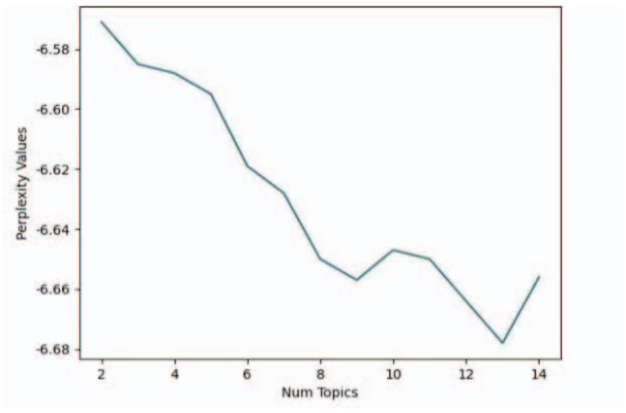


Fig.3 Topic perplexity values

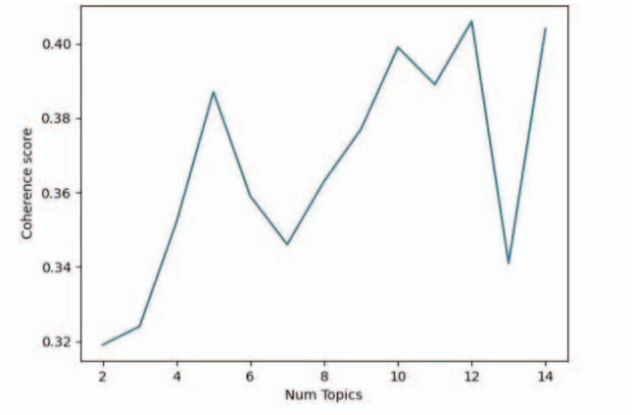


Fig.4 Topic coherence score

Table 1: CWE-362 topic names and top 10 topic words

No	Topic name	Topic words
1	file system	use vulner file system user cause race compil xen condit
2	code execute	affect issue race condit attack allow arbitrari execute discov code
3	dential of service	use allow snapdragon condit race denial service access cause guest
4	pv operate	guest pv operate use ubuntu paget promote allow samsung issue
5	kernel	condit race kernel linux android user allow local use function
6	privilege	allow race condit cause service denial attack version user prior
7	local execute	privilege user local race condit android need allow could execute
8	release version	attack earlier cause allow release condit via service race version
9	window attack	window sp server condit race alllow attack user microsoft vulner
10	remote attack	allow race condit via attack service cause remote dential user
11	user attack	allow attack condit access user server race use vulner affect
12	local service	allow condit race user via cause local service dential kernel

Next, we determine the optimal number of topics by topic perplexity and topic coherence. The higher the topic perplexity is the better and the lower the topic coherence is the better, they both vary positively with the number of topics, but the number of topics cannot be infinite, so a balance needs to be found in the image of perplexity and coherence, so we finally set the number of topics value to 12 based on the results in Figure 3 and Figure 4. As shown in Table 1, we obtain the distribution of topic words for the 12 topics and read the words under each topic to identify each topic.

Finding 3.2: Concurrency vulnerabilities caused by race condition are the most common. Race condition involves serious remote attacks to various services, from kernel to file system.

(3) Severity of concurrency vulnerabilities: BaseScore represents the intrinsic characteristics of a software vulnerability that do not change over time or with the environment. The higher score the vulnerability has, the more risk it produces. Figure 5 shows the BaseScore distribution of concurrency vulnerabilities. We can see that there are 564 concurrency vulnerabilities with scores between 4.0 and 6.9, and 151 concurrency vulnerabilities with scores above 7.0. Besides, there are 115 concurrency vulnerabilities with a score of 6.9 (the maximum score of 10).There are four concurrency vulnerabilities that have a score of 10.

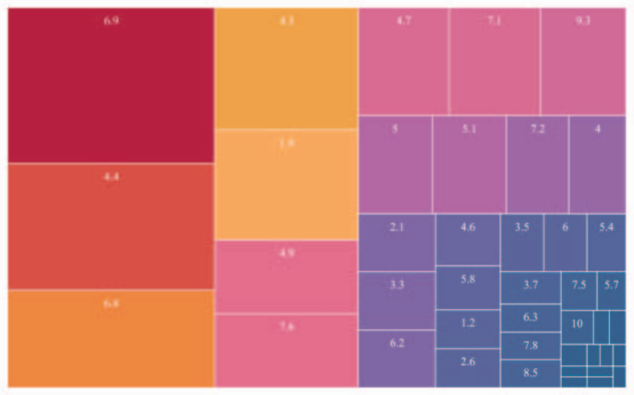


Fig.5 BaseScore distribution

Finding 3.3: The overall risk of concurrency vulnerabilities is medium risk (between 4.0 and 6.9), it accounts for 67.2%. The numbers of concurrency vulnerabilities with high risk (above 7.0) and low risk (less than 4.0) are almost equal (151 and 124, respectively).

(4) The exploitability of concurrency vulnerabilities: Access Vector reflects that how the vulnerability is exploited. There are three manners to exploit vulnerabilities: local network, adjacent network and remote network. The distribution of access vector is shown in Figure 6.

Figure 6 shows that among 839 concurrency vulnerabilities, 447 of them can be exploited through local access and 377 of them can be exploited through remote network access. Nearly half of the concurrency vulnerabilities can be accessed remotely. Furthermore, we explored the characteristics of concurrency vulnerabilities considering both severity and exploitability. Among 377 concurrency vulnerabilities, 127 concurrency vulnerabilities have a BaseScore more than 7.0. Only 37 of them have a BaseScore of 9.0 or more, and critical dangerous vulnerabilities account for 29% of high-risk vulnerabilities. The remaining 250 concurrency vulnerabilities that can be exploited via remote network access all score below 7.0 and can be considered low risk and medium risk. Take CVE-2020-14059[11] for example, it can be remotely accessed to exploit concurrency vulnerabilities, but it has a BaseScore of 4.0, For CVE-2020-3894[12], but it only has a BaseScore of 2.6. More than half of the concurrency vulnerabilities (462/839) are local network and adjacent network and only 24/462 concurrency vulnerabilities have a BaseScore more than 7.0.

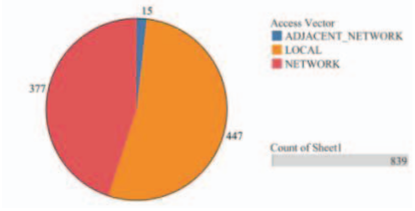


Fig.6 Access Vector distribution

Finding 3.4: Most of the concurrency vulnerabilities are exploited through local access (53%), the next is remote network access (45%). Moreover, not all concurrency vulnerabilities exploited via remote network access are high-risk.

(5) The complexity of the attack: Access Complexity shows the complexity of the attack by the attacker after gaining access and exploiting the vulnerability. The distribution of Access Complexity is given in Figure 7. We can see that 571 concurrency vulnerabilities have the medium access complexity. The number of concurrency vulnerabilities that have the high and low access complexity is almost equal. Low and medium complexity attacks are more likely to cause concurrency vulnerabilities, which has implications for how developers can improve the complexity of their code.

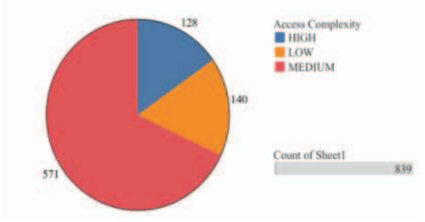


Fig.7 Access Complexity distribution

Moreover, we explore the times of target authentication to exploit the concurrency vulnerabilities, which is shown in Figure 8. NONE means that no authentication instance is required, and SINGLE means that a single authentication instance is required. The fewer authentication instances, the higher risk of the vulnerability. Among 839 concurrency vulnerabilities, 785 of them can be exploited by attackers without verification to the target. The result indicates that it is easy to attack directly from concurrency vulnerabilities. In CVE-2013-1895[13], the py-bcrypt module before 0.3 for Python does not properly handle concurrent memory access, which allows attackers to bypass authentication via multiple authentication requests, which trigger the password hash to be overwritten. Therefore, it is very important for network security workers to enhance security privileges and strengthen authentication.

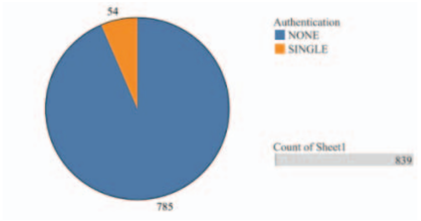
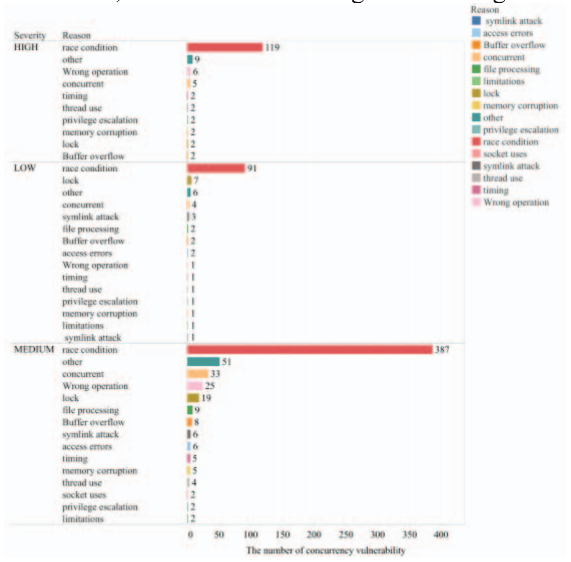


Fig.8 Authentication distribution

Finding 3.5: Most attackers often use concurrency vulnerabilities to conduct direct attacks due to low access complexity. And it is easy to attack directly from concurrency vulnerabilities because most concurrency vulnerabilities can be exploited by attackers without verification.

(6) Severity of concurrency vulnerabilities with different reasons and results: To explore the distribution of reasons and results corresponding to high, medium and low (vulnerability severity) risks of concurrency vulnerabilities, we conduct a combination analysis of severity of concurrency vulnerabilities, which is shown in Figure 9 and Figure 10.



11 https://nvd.nist.gov/vuln/detail/CVE-2020-14059
12 https://nvd.nist.gov/vuln/detail/CVE-2020-3894

13 https://nvd.nist.gov/vuln/detail/CVE-2013-1895

Fig.9 Severity corresponding reasons

As can be seen in Figure 9, despite the vulnerability severity (high, medium, and low risk), the concurrency vulnerabilities caused by race condition are the most, and in Figure 10, the result of the concurrency vulnerabilities is the denial of service, which accounts for much higher proportions than other results. Our findings enable researchers to be aware of the hazards posed by race condition and to take appropriate measures to improve network security by avoiding denial-of-service attacks, so future research should try to employ multi-processing concurrent servers for clients to reduce the severity caused by concurrency vulnerabilities.

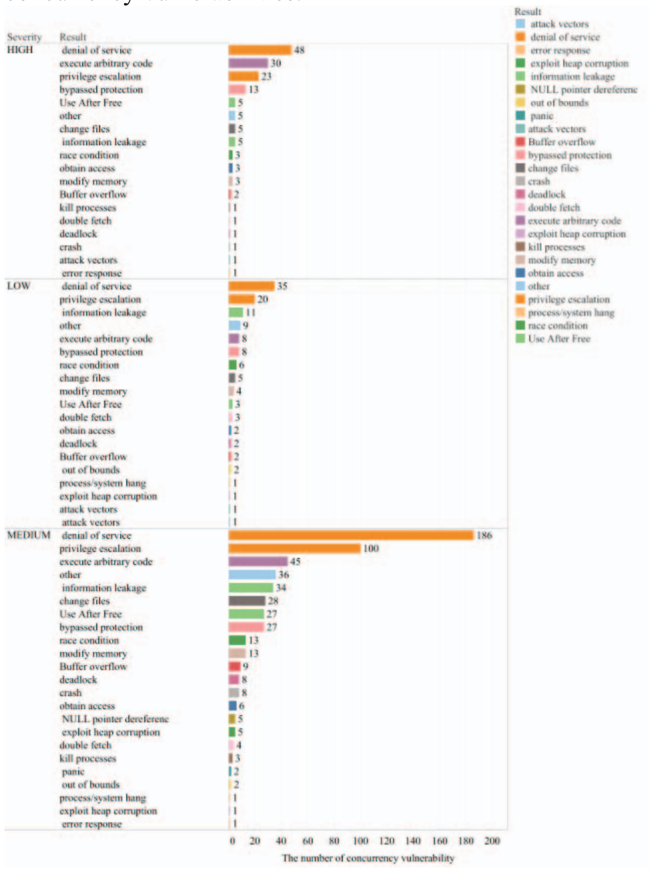


Fig.10 Severity corresponding results

It can also be found in Figure 10 that, in the MEDIUM case with the most vulnerabilities, the result of the concurrency vulnerabilities is the denial of service, which accounts for much higher proportions than other results, followed by privilege escalation. From the distribution in HIGH, we can see that concurrency vulnerabilities that lead to denial of service, privilege escalation and execute arbitrary code account for the most.

Finding 3.6: Race condition is the most important cause of concurrency vulnerabilities corresponding to high, medium, and low risk severity levels and denial of service occurs most in medium-risk concurrency vulnerabilities.

(7) Confidentiality Impact of concurrency vulnerabilities with different reasons and results: To explore the distribution of the impact of successful exploitation of concurrency vulnerabilities on confidentiality (confidentiality refers to restricting the access and disclosure of information to authorized users, and preventing access or disclosure to unauthorized persons), and the causes as well as results of concurrency vulnerabilities under the influence, we conduct a combination analysis of confidentiality impact of concurrency vulnerabilities, which is shown in Figure 11 and Figure 12.

In Figure 11, None indicates that it has no impact on the confidentiality of the system, Partial indicates that a considerable amount of information is disclosed, and Complete indicates that the vulnerability may cause all system files to be leaked.

Through the result in Figure 11 and Figure 12, it can be found that the number of concurrency vulnerabilities in each situation is roughly equal, and the distribution of concurrency vulnerabilities caused by race conditions is also the same. Race conditions can lead to denial of service and execute arbitrary code, which are more likely to cause the disclosure of system files, so improving access rights is a good solution for network security workers.

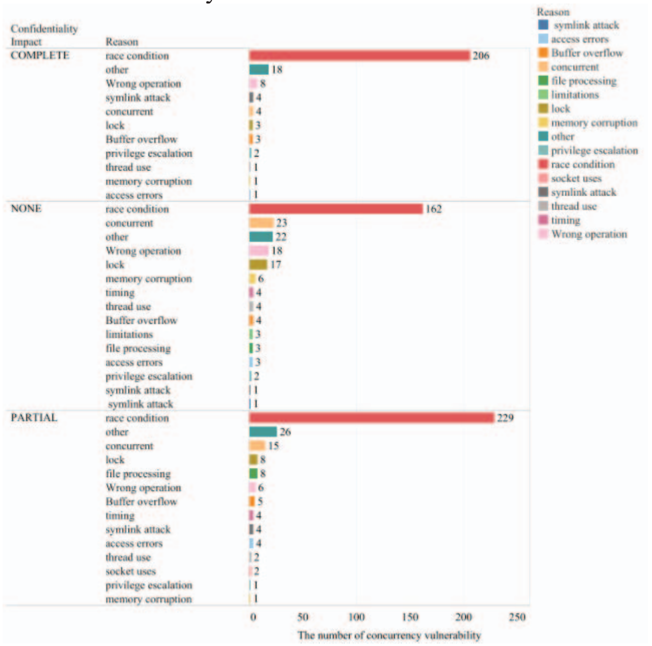


Fig.11 Confidentiality Impact corresponding reasons

Figure 12 shows the distribution of results caused by concurrency vulnerabilities in the confidentiality impact. From the Figure 12, we can see that execute arbitrary code caused by concurrency vulnerability is distributed in COMPLETE significantly higher than in the other two cases, and execute arbitrary code was more likely to cause the system file to be leaked. Besides, denial of service can also easily cause the system file to be complete leaked.

Finding 3.7: Denial of service, execute arbitrary code and privilege escalation are more likely to cause system files to be leaked.

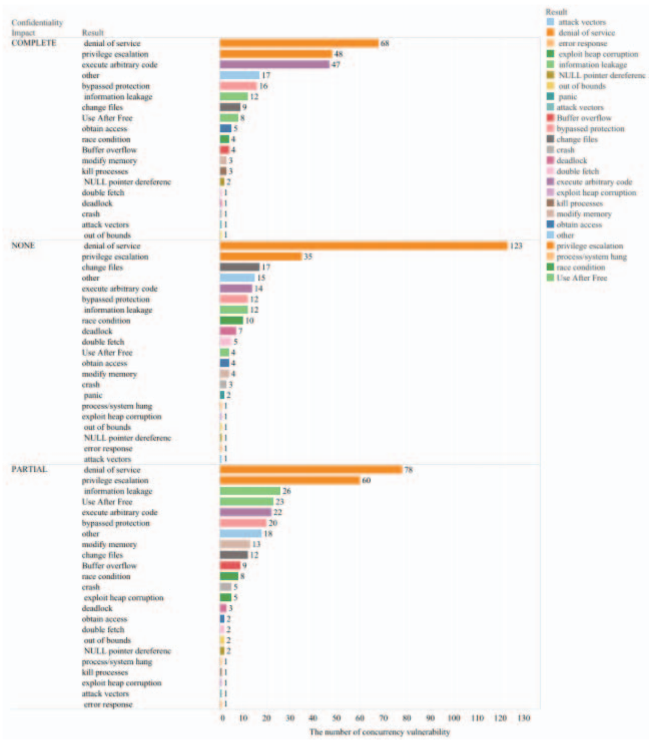


Fig.12 Confidentiality Impact corresponding results

(8) Integrity impact of concurrency vulnerabilities with different reasons and results.

To get the integrity impact of concurrency vulnerabilities, we explore the distribution of the impact of successfully exploiting concurrency vulnerabilities on integrity, as well as the distribution of the causes and results of concurrency vulnerabilities at each impact level. The results are shown in Figure 13 and Figure 14.

In Figure 13, Integrity Impact means that concurrency vulnerabilities are exploited to affect the integrity of the system, where NONE means that it does not affect the integrity of the system, PARTIAL means that some system files or information may be modified, but the attacker's scope of influence is limited and can not control the content to be modified, COMPLETE means that the system protection is completely lost, and the attacker can modify any file on the target system, which will cause the entire system to be destroyed [11].From Figure 13, we can find that the distribution of various situations is quite balanced. In all cases, the number of concurrency vulnerabilities caused by race conditions are the largest. The concurrency vulnerabilities caused by the lock and wrong operation in NONE are more than that in COMPLETE and PARTIAL.

From Figure 14, we can find that for the concurrency vulnerabilities leading to executing arbitrary code, the distribution in COMPLETE is significantly more than the distribution in the other two cases. For the concurrency vulnerabilities leading to denial of service, the distribution in NONE is significantly more than the distribution in the other two cases.

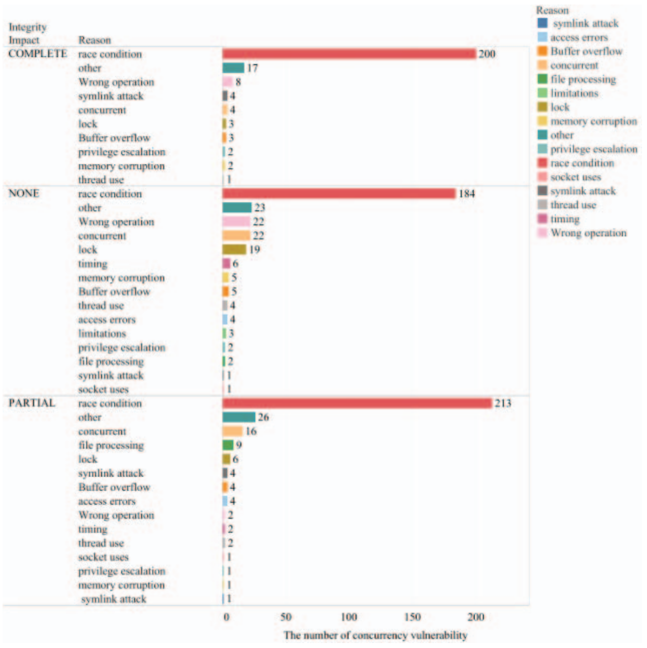


Fig.13 Integrity Impact corresponding reasons

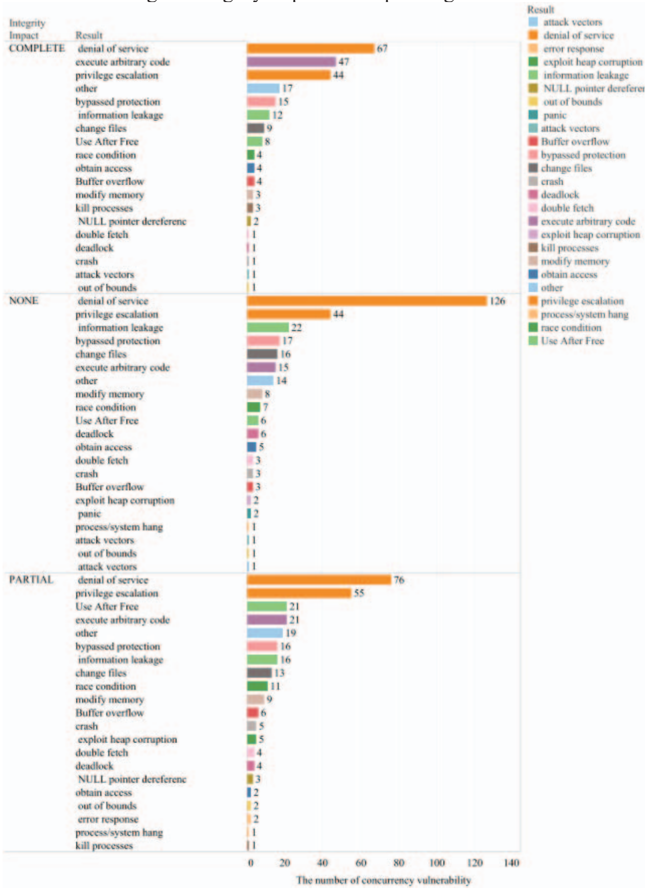


Fig.14 Integrity Impact corresponding results

Finding 3.8: For concurrency vulnerabilities leading to denial of service, the distribution in NONE is significantly more than the distribution in COMPLETE and PARTIAL.

(9) Availability impact of concurrency vulnerabilities with different reasons and results: To get the availability impact of concurrency vulnerabilities, we explore the distribution of the causes and results under the hierarchical distribution of the impact on availability. The results are shown in Figure 15 and Figure 16.

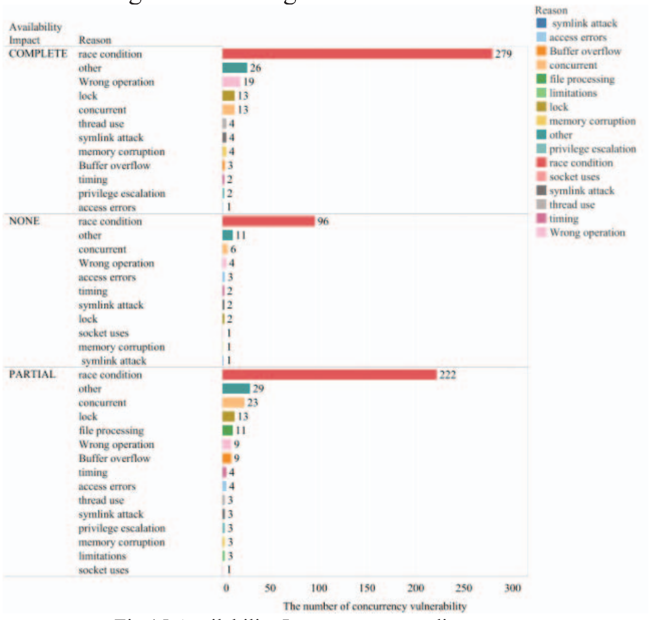


Fig.15 Availability Impact corresponding reasons

In Figure 15, NONE means no impact on the availability of the system, PARTIAL means performance degradation or interruption of resource availability, and COMPLETE means all affected resources are closed, and the attacker can make the resource unavailable completely [11]. In Figure 15, we can see that the concurrency vulnerabilities distributed in PARTIAL and COMPLETE are significantly more than that in NONE, this indicates that concurrency vulnerabilities have a greater impact on the system. As seen in Figure 16, concurrency vulnerabilities triggered by race condition have a greater impact on availability, and if a concurrency vulnerability results in a denial of service and execution of arbitrary code, there is a greater probability that the vulnerability will be exploited with a complete loss of availability.

Finding 3.9: Concurrency vulnerabilities have a much greater impact on availability, especially for concurrency vulnerabilities caused by race condition. if the concurrency vulnerabilities cause denial of service and executing arbitrary code, that the probability of complete loss of availability after the vulnerability is exploited is greater.

(10) Summary. In this section, we mainly analyze the trend, classification, cause, impact, and severity of concurrency vulnerabilities, which can provide a new classification basis for concurrency vulnerability analysis, and other findings we obtained are also instructive for researchers. First of all, in Figures 1 and Figure 2, the number of concurrency vulnerabilities is increasing, and race condition is very common, which requires developers to pay attention to concurrency vulnerabilities. In Figure 5, medium-risk concurrency vulnerabilities account for the majority, researchers need to pay more attention to fix medium-risk vulnerabilities. our research in Figure 6 find that concurrency vulnerabilities are often exploited by remote network access and local access, so researchers need to raise the level of remote access and local access protection and set access permissions. In Figures 7 and 8, attackers are more likely to exploit concurrency vulnerabilities to directly attack networks with lower access complexity, so network security workers need to increase the number of authentication instances and code complexity to improve network reliability. When it comes to the severity caused by concurrency vulnerabilities, vulnerability researchers prioritize related work based on severity, and our findings in Figures 9 and 10 can provide guidance for vulnerability repair, such as researchers should pay more attention to denial of service, execute arbitrary code and privilege escalation, etc. At the same time, for some other vulnerability types, such as lock, socket uses, memory corruption, they also need to attract the attention of researchers.

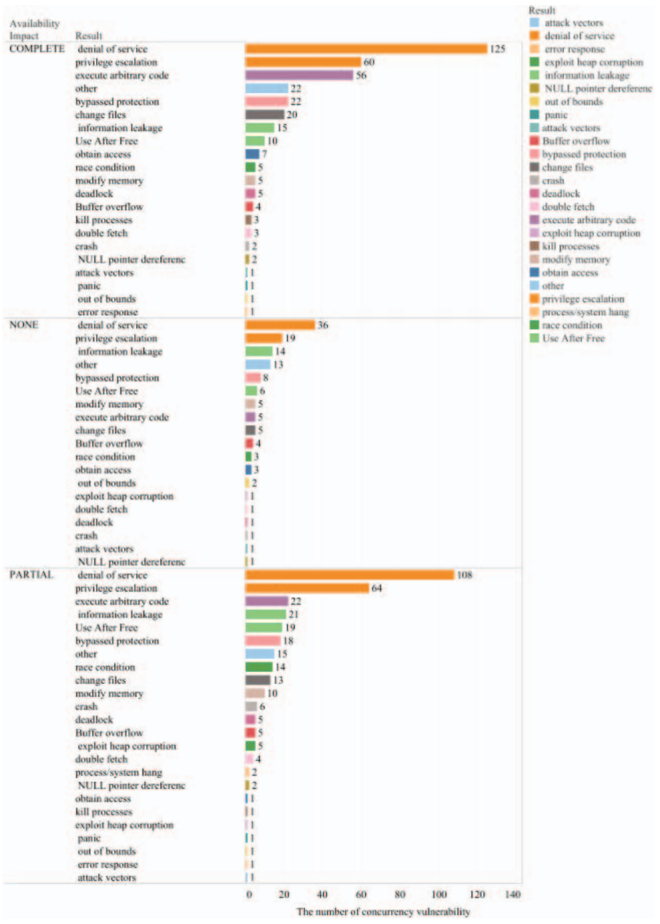


Fig.16 Availability Impact corresponding results

IV. Related Work

Vulnerability analysis is an old and new topic in the field of information technology [12][13][14], In 2010, Yang et al. [15][16] first conducted a research proving the feasibility of exploiting program vulnerabilities to attack against parallel programs. Lu et al. [17][18] systematically studied various problems in parallel programs, such as data race and deadlock. In order to investigate the common types of vulnerabilities and observe new trends in vulnerabilities, some researchers focus on collecting and analyzing the vulnerability database data. Stephan Neuhaus et al. [19] conducted the vulnerability research to analyze the security trends based on the CVE topic model. After their vulnerability research, they observed the trends that, PHP declines, occasionally SQL injection, buffer overflow flattens after the decline; Format String declines sharply, SQL Injection and XSS stay strong and rise. Cross-Site Request Forgery likes a sleeping giant, and is shaking. Application Servers rises sharply. Jin et al. [20] did similar work to our work. They used the data analysis tool Tableau for data analysis of information security vulnerabilities, visualizing the development trend of vulnerabilities to intuitively understand the development trend of vulnerabilities and summarize the characteristics of vulnerabilities in products from manufacturers of different scales. Our work is the first to conduct a comprehensive study on the intrinsic characteristics of different types of concurrency vulnerabilities in CVE.

In addition, from the perspective of the system, the researchers studied the concurrency vulnerabilities on the Windows as well as the Linux platforms and proposed a variety of detection methods, e.g., the method using the full simulation memory access tracking [21], the method based on static pattern matching [5], formal definition [22], the vulnerability detection from the perspective of caching mechanism [23] etc. Wang [5] proposed a method based on static pattern matching to detect concurrency vulnerabilities in Linux. Xu et al. [22]proposed formal definition of concurrency vulnerabilities to detect the concurrency vulnerabilities. Schwarz et al. [23] proposed to use trusted execution environment methods, transactional memory technology to maintain data consistency. Jurczyk and Coldwind [21] developed Bochspwn, a method based on Bochs [24], using fully simulated memory access tracking to detect concurrency vulnerabilities on the Windows operating system. Milena et al. [25] conducted a concurrency vulnerabilities search based on bounded model checking. The researches on vulnerability detection lead to the development of the researches on vulnerability protection, which are mainly carried out from the following four aspects: fetching data once, using the same data, overwriting data, and adding synchronization operations.

Different with all above work, we conducted a comprehensive analysis of NVD concurrency vulnerabilities, including the classifications, the reasons, the severity and the impacts. Moreover, we combined multiple elements to explore the relationship of vulnerability classifications, root cause, severity and impacts. This is important for understanding the intrinsic characteristics of concurrency vulnerabilities.

V. Conclusions

With the development of multi-core systems, concurrent programming is widely used, producing concurrency vulnerabilities inevitably. In this paper, we focus on the types of concurrency vulnerabilities. We extract concurrency vulnerabilities from the vulnerability database CVE, and analyze the trends, classifications, causes, impacts and severity. Our empirical study on CVE shows that: (1) Race condition is the most common type of concurrency vulnerabilities and it is the most significant cause of concurrency vulnerabilities corresponding to high, medium and low risk severity. (2) Most of the concurrency vulnerabilities belong to medium risk. (3) Concurrency vulnerabilities caused by exploiting local network access are the most. (4) When examining the integrity impact on the system caused by exploiting concurrency vulnerabilities, the distribution of concurrency vulnerabilities that lead to the execution of arbitrary code is significantly greater in COMPLETE than in NONE and PARTIAL. (5) Authentication instance can greatly reduce the risk of exploiting concurrency vulnerabilities. Besides, we examine the main results of high-risk concurrency vulnerabilities, which are denial of service, execute arbitrary code and privilege escalation. These findings can guide developers and network security administrators to improve system security. For example, when a low privilege account elevates privileges and executes arbitrary code as a system user, it is necessary to first check the security logs and system files for substitution.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper

Acknowledgment

This work is supported by the National Natural Science Foundation of China (No. 61872312, No. 61972335, No. 62002309); the Yangzhou city-Yangzhou University Science and Technology Cooperation Fund Project (No.YZU201902); the Six Talent Peaks Project in Jiangsu Province (No. RJFW-053), the Jiangsu “333” Project, the Open Funds of State Key Laboratory for Novel Software Technology of Nanjing University (No. KFKT2022B17), the Key Laboratory of Safety-Critical Software Ministry of Industry and Information Technology (No. NJ2020022), the Natural Science Research Project of Universities in Jiangsu Province (No. 20KJB520024), and Yangzhou University Top-level Talents Support Program (2019).

References

[1] Y Cai, B Zhu, R Meng, et al. Detecting concurrency memory corruption vulnerabilities[C]. Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), 2019: 706-717.
[2] M Stephen. Programming Language Vulnerabilities Proposals to Include Concurrency Paradigms[J]. Ada letters, 2013, 33(1).
[3] S Wu, C Zhang, F Wang. Extracting Software Security Concerns of Problem Frames Based on a Mapping Study[J]. APSEC Workshops, 2017: 121-125.
[4] Z Zeineb, R Yves. Static code analysis for software security verification: problems and approaches[J]. COMPSAC Workshops, 2014: 102-109.
[5] P Wang, J Krinke, K Lu, et al. How double-fetch situations turn into double-fetch vulnerabilities: A study of double fetches in the linux kernel[C]. Proceedings of the 2017 26th USENIX Security Symposium (USENIX Security), 2017: 1-16.
[6] J Wang, W Dou, Y Gao, et al. A comprehensive study on real world concurrency bugs in Node.js[C]. Proceedings of the 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), 2017: 520-531.
[7] S Ahmed, M Bagherzadeh. What do concurrency developers ask about?: a large-scale study using stack overflow[J]. ESEM 2018: 30:1-30:10.
[8] H Fu, Z Wang, X Chen, et al. A systematic survey on automated concurrency bug detection, exposing, avoidance, and fixing techniques[J]. Software Quality Journal, 2018: 26(3).
[9] S Lu, S Park, E Seo, Y Zhou. Learning from mistakes: a comprehensive study on real world concurrency bug characteristics[J]. ASPLOS, 2008: 329-339.
[10] D Blei, A Ng, M Jordan. Latent dirichlet allocation[J]. Journal of machine Learning research, 2012: 993-1022.
[11] Y Wei, X Sun, L Bo, et al. A comprehensive study on security bug characteristics[J]. Journal of Software Evolution and Process, 2021, 33(10).
[12] J Sanghoon, K Kang. AutoVAS: An Automated Vulnerability Analysis System with a Deep Learning Approach[J]. Computers&Security, 2021, 106: 102308.
[13] S Nagesh, W Matthias, et al. An analysis of the vulnerability of two common deep learning-based medical image segmentation techniques to model inversion attacks[J]. Sensors, 2021, 21(11): 3874.
[14] P Cigoj, Z Stepancic, et al. A large-scale security analysis of web vulnerability: findings, challenges and remedies [J]. ICCSA (5), 2020: 763-771.
[15] J Yang, A Cui, et al. Concurrency attacks[C]. Proceedings of the 2012 4th Usenix Conference on Hot Topics in Parallelism. USENIX Association(HotSec), 2012:15-15.
[16] J Yang, P Twohey, D Engler, et al. Using model checking to find serious file system errors[J]. ACM Transactions on Computer Systems, 2006, 24(4):393-423.
[17] S Lu, S Park, Y Zhou. Finding atomicity-violation bugs through unserializable interleaving testing[J]. IEEE Trans. Software Eng, 2012:38(4).
[18] S Lu, J Tucek, F Qin, et al. AVIO: detecting atomicity violations via access interleaving invariants[J].ACM SIGOPS Operating Systems Review, 2006, 40(5): 37-48.
[19] N Stephan, Z Thomas. Security trend analysis with CVE topic models[C]. Proceedings of the 2010 21st International Symposium on Software Reliability Engineering (ISSRE), 2010: 111-120.
[20] R Jin, J Nan. Combining sources from CVE and CNNVD: Data analysis in information security vulnerabilities[J]. Journal of Physics: Conference Series, 2021,1800(1).
[21] M Jurczyk, G Coldwind. Bochspwn: Identifying 0-days via system-wide memory access pattern analysis[J]. 2016Black Hat USA Briefings (Black Hat USA), 2013.
[22] M Xu, C Qian, K Lu, et al. Precise and scalable detection of double-fetch bugs in OS kernels[C]. Proceedings of the 2018 39th IEEE Symposium on Security and Privacy (SP), 2018: 661-678.
[23] M Schwarz, D Gruss, et al. Automated detection, exploitation, and elimination of double-fetch bugs using Modern CPU Features[C]. Proceedings of the 2018 13th on Asia Conference on Computer and Communications Security(CCS), 2018: 587-600.
[24] https://bochs.sourceforge.io/.
[25] R Yoo, C Hughes, et al. Performance evaluation of Intel® transactional synchronization extensions for high-performance computing[C]. Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis(SC), 2013: 1-11.