Optimization Method of Web Fuzzy Test Cases Based on Genetic Algorithm

Sheng Qu1, Zheng Zhang1*, Bolin Ma1, and Yuwen Shao1

1 Computer technology, Information Engineering University, Zhengzhou, Henan, 450001, China
*Corresponding author’s e-mail: ponyzhang@126.com

Abstract. In order to solve the problems of low code coverage, few vulnerabilities found, and poor fuzzing effect caused by the small number of test cases and single types in Web fuzzing, on the basis of studying the current Web fuzzing methods, the existing fuzzing Web applications are tested Program research. A genetic algorithm-based method for optimizing fuzzing test cases for Web applications is proposed. It analyzes and counts the traffic of public network website business with Web service attack characteristics, and uses genetic algorithms to generate a large number of test cases with various types to explore the Web service vulnerability that exists. Based on the creation of a Web attack signature database with weights, this method uses genetic algorithms to randomly pre-generate the test cases of the fuzzing test, and uses the response of the Web service to repeatedly iterate the weights of different attack signatures in the Web attack signature database. So as to generate the best test cases. Experimental analysis shows that this method effectively finds security vulnerabilities in Web applications.

1. Introduction

2019 CNVD [1] The vulnerabilities included in the classification according to the type of impact object, as shown in Figure 1, the proportion The top three are application vulnerabilities (56.2%), web application vulnerabilities (23.3%), and operating system vulnerabilities (10.3%). From the classification of vulnerabilities included in CNVD in 2019 according to the types of affected objects, it can be known that the impact of web application vulnerabilities accounts for a relatively large proportion.
Figure 1. Vulnerabilities included in CNVD in 2019 are classified and counted according to the types of affected objects.

In recent years, the number of active users of the Internet and Web applications has been increasing, and the interface between Web applications and the Internet has led to a higher degree of security risks. Because web applications are completely open to global users, the security of web application servers cannot be ignored. At present, there are some methods to manage the security risks of Web applications, such as firewalls, defensive coding, monitoring and auditing. In practice, white customers or security experts around the world usually check security vulnerabilities and possible vulnerabilities through vulnerability assessment and penetration testing. Vulnerability assessment and penetration testing provide protection of network assets. According to reliable statistics, there are more than 72 million hosts on the Internet. In other words, this may mean that there may be billions of people in the network neighborhood, which makes it difficult to guarantee the security of network and web application users.

With the entry of the Web3.0 era, website functions have become more and more complex, especially the widespread popularity and application of the Internet, such as the emergence of e-government, e-commerce, online office, online media, and virtual communities, which are profoundly affecting human life and work. At the same time, the importance of Web security is constantly increasing. The more complex the functions of the website means that more vulnerabilities are exposed to the Internet. The Web security problems faced by various organizations of governments and enterprises are becoming more and more diversified and complicated. The threat of hackers is growing rapidly, causing more vulnerabilities to the information network of enterprises. Serious damage or even irreparable loss. Therefore, improving the security of Web applications is the top priority in solving Web security problems.

The web server of mimic defense structure is the first available mimic defense system developed from theory to practice in mimic structure system. At the beginning of 2016, the mimic web server passed the national test; in April 2018, in accordance with the unified deployment of the Ministry of Industry and Information Technology, the mimic web server was officially launched in Henan to provide online services; in May 2019, the mimic web server participated in the second mimicry international elite challenge. It has withstood more than 2.9 million attack tests conducted by 29 “white hat” hacker teams around the world. In June 2020, the mimic web server was still untouched in the 3rd Mimic International Elite Challenge. So far, the mimic web server has carried out pilot cooperation with the Industrial and Commercial Bank of China, China Telecom, Henan Jingan Network and other units.

In the evaluation of the mimic Web server's ability to resist unknown vulnerabilities and backdoors, a mimic Web server security test method based on fuzzing technology is adopted, in this method For the test cases of fuzzing, there are still problems such as single use cases and poor diversity.
2. Related work

Web application vulnerabilities are one of the most serious cyberspace security threats\cite{6-7}, against XSS in Web applications. Vulnerabilities have attracted wide attention from scholars at home and abroad since they were discovered in 1996. Many research organizations or institutions have done a lot of research on their detection work and proposed related methods, which mainly include three categories: static analysis methods and dynamic analysis. Methods and analysis methods combining dynamics and statics\cite{8}. Zheng et al.\cite{9} proposed a path sensitive static analysis of web applications for remote code execution vulnerability detection. A path- and context-sensitive inter-process analysis is proposed to detect Remote Control Equipment (RCE) vulnerabilities. Its characteristic is to analyze the string and non-string behavior of the Web application in a path-sensitive way. Ibéria et al.\cite{10} proposed the use of static analysis and data mining detection to remove Web application vulnerabilities. Because the static analysis method depends on the source code of the application, it is very difficult to obtain the source code of the application during the detection process, and the static detection method cannot verify the authenticity of the vulnerability, so there are certain limitations, so, More people tend to use dynamic or static and dynamic methods to detect vulnerabilities. Duchene et al.\cite{11} proposed a black box fuzzer for detecting cross-site scripting vulnerabilities in web applications. The fuzzer uses genetic algorithms to generate malicious input, using attack syntax as algorithm parameters. The attack syntax can reduce the search space and simulate attack behaviors by restricting crossover and mutation operations, and the original string matching method may not be able to accurately infer Taint location, so it uses double taint inference to obtain accurate results. Although this method uses control flow dependence and data flow dependence to guide fuzzing and solves the problem that most black box detectors cannot detect reflective XSS, this method Ajax program is not supported, and the scope of application is limited. Cheng Cheng et al.\cite{12} proposed an XSS vulnerability detection method based on fuzzing and genetic algorithm. This method mainly defines the grammatical completion rules of the attack vector by analyzing the filtering mechanism of the server, and then uses the genetic algorithm to continuously analyze and optimize the attack load to generate the optimal vector. However, the attack code used in this method requires manual input, which requires a large amount of manual operation and is not automated. Based on the traditional XSS vulnerability detection technology, in terms of mining vulnerability injection points, the collection of user injection points is not comprehensive because the anti-crawling mechanism of some websites is not considered, or the XSS vulnerability detection method based on fuzzing and genetic algorithm supports Ajax programs, etc. And, because of a series of problems such as a single type of XSS attack payload or insufficient detection coverage, the system's detection performance is low, and vulnerabilities cannot be found comprehensively and effectively. Ni Ping et al.\cite{13} proposed a reflective XSS vulnerability detection method based on fuzzing. This method uses web crawlers to extract more potential user injection points in the page and reduce In order to analyze the attack load of the website response and adjust the weight of the attack load element to generate a more efficient load to find the existing vulnerabilities, in order to use the targeted use of the attack load of the website response, the constant change of the weight in the process of generating test cases leads to the fuzzing test. Resource consumption increases, and for test cases, test cases with low weights cannot be regenerated, and there is a possibility that security vulnerabilities cannot be discovered.

Considering the above problems comprehensively, in Section 3 of this paper, a detailed description of the proposed method for optimization of Web fuzzing test cases based on genetic algorithm is given.

3. Web attack use case optimization based on fuzzing and genetic algorithm

This method is shown in Figure 2. By using the attack traffic detection system to summarize the attack types of the traffic with attack characteristics in the access traffic of the public network, and then manually analyze and record the composition of attack elements of different attack types; The proportion of attack elements is assigned weights; these weighted elements are brought into the genetic algorithm generating test case system, thereby generating a large number of new test cases with attack characteristics.
Figure 2. Flow chart of optimization of Web fuzzing test cases based on genetic algorithm

3.1. Web access data acquisition and classification
The architecture of the attack traffic detection system is shown in Figure 3. First, determine the name of the file to be analyzed, and pass it to the attack traffic detection system. Then the system will traverse and read each log in the file, and then according to the matched attack type element Summarize, and finally generate a detection report of the attack traffic type.

Figure 3. Attack traffic detection system architecture diagram

3.2. Attack traffic signature database
According to the data obtained by the attack traffic detection system, the element types and attributes of different attack types are analyzed. This article mainly uses XSS attacks as a case to analyze the element types and type attributes. Among them, each element type of XSS includes six categories: header tags Class, handler class, space class, payload class, tail tag class, and symbol class. The attributes of specific types include, see Table 1 below:

Table 1. XSS attack element type analysis table.

| XSS each element type | Type attribute |
|-----------------------|----------------|
| Header tag            | `<a href`, `<body`, `<form`, `<frameset`, `<iframe`, `<img`, `<input`, `<script`, `<video` |
| Handler               | `onerror=`, `onload=`, `onmouseEnter=`, `onmouseLeave=`, `onMouseOut=`, `onmouseover=`, `onpropertyChange=`, `onreadystatechange=`, `onscroll=`, `onResize=`, `src=` |
| space                 | `&nbsp;`, `\n`, `\t` |
| Payload               | `"alert(1)"`, `"javascript:alert(1)"` |
| Tail tag              | `>`, `">`, `""`, `""`, `"/>`` |
| Symbol                | `=`, `""`, `?` |
3.3. Genetic algorithm generates test cases

The genetic algorithm first uses different types of elements in the attack traffic signature database as the initial population. The fitness function in the genetic algorithm is a maximization function. The function of this function is to assign a value to each instance evaluation. The fitness function is $	ext{fitness}() = \text{numelems} + \text{numatrs} - \text{numerrors}$; including: the number of attributes (numatrs), the number of elements (numelems), and the number of errors generated (numerrors). The higher the optimization degree of the sample, the larger the value of the fitness function. Use the BeautifulSoup function to perform lxml analysis on the website response to get the number of attributes, the number of elements, and the number of errors generated in the fitness function.

The flowchart of the genetic algorithm is shown in figure 4. After determining the initial population and fitness function, the method is optimized for the selection operation. In the selection operation, those healthy seeds are selected, that is, the attack is selected in the attack traffic detection system. High-threat elements are used as healthy seeds; when the selected seeds are below a certain threshold, they are discarded. In the selection process, the roulette selection algorithm, the sort selection algorithm, and the tournament selection algorithm are used; in the crossover process, two different test cases are crossed to generate new test cases; the mutation process is divided into coding and insertion. These are all implemented by genetic algorithm modules.

![Figure 4. Genetic algorithm flow chart](image)

The roulette selection algorithm\cite{14} in the selection operation is shown in figure 5; the size of the area corresponds to different probabilities, the larger the area, the greater the probability. Print this picture on a piece of paper, and throw a handful of millets at random. The number of millets that fall in the green area is relatively the largest.

![Figure 5. The roulette selection algorithm](image)
The tournament selection algorithm \cite{15} is shown in the figure; each time a certain number of individuals are removed from the population (returned sampling), and then the best one is selected to enter the offspring population. Repeat this operation until the new population size reaches the original population size. The multi-dollar tournament is to take out several individuals from the overall at one time, and then take the best individuals from these individuals and put them into the set that is reserved for the next generation of the population. The specific steps are as follows:

1. Determine the number of individuals selected each time N. (Binary tournament selection means selecting 2 individuals)
2. Randomly select N individuals from the population (each individual has the same probability of being selected), and according to the fitness value of each individual, select the individual with the best fitness value to enter the next-generation population.
3. Repeat step (2) several times (the number of repetitions is the size of the population) until the new population size reaches the original population size.

![Figure 6. Tournament selection algorithm](image)

4. Experiment and result analysis
In order to effectively and objectively evaluate the optimization of the Web fuzzing test cases proposed in this article, the dvwa test website is selected to test the implementation scheme, and then compared with the experimental results of the traditional Web fuzzing test, and then the final evaluation conclusion is drawn.

4.1. data collection
Analyze nearly 2 million access logs of the "3rd Mimic Elite Challenge". When counting elements, you need to pay attention to the ASCII transcoding in the logs. Common ASCII codes in the logs are shown in figure 7.

![Figure 7. Common ASCII codes](image)

Through the analysis of the attack traffic detection system, the statistics in Table 2 are obtained: Among them, there are 81 use cases of SQL injection attack type, 569 use cases of cross-site attack script (XSS) attack type, 11430 use cases of sensitive file download attack type, and directory traversal attack. There are 629 use cases of type, 695 use cases of command injection attack type, local files contain 656 use cases of attack type, 28 use cases of 0Day vulnerability attack type, and 19 use cases of Webshell attack type.
Table 2. Attack traffic detection system data statistics.

| Attack type                  | Total |
|------------------------------|-------|
| SQL injection                | 81    |
| Cross-site attack script     | 569   |
| Sensitive file download      | 11430 |
| Directory traversal          | 629   |
| Command injection            | 695   |
| File Include                 | 656   |
| Zero Day Vulnerable          | 28    |
| Webshell                     | 19    |

4.2. *lab environment*

Test machine operating system: Windows 10, 64-bit operating system; script interpreter: Python3.7; memory 32G; processor: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz; Web server operating system: CentOS Linux release 7.6; Number of CPUs: 1; Memory: 1G; Web service type: Nginx.

4.3. *Experiment and result analysis*

According to the different genetic algebra in the genetic algorithm, test statistics are performed. As shown in Table 3, the attack traffic detection system obtains statistics through 629 XSS attack logs. The test cases can be fully enriched in the genetic algorithm. It can reach 4 to 5 times the original data volume, and can be obtained through the analysis of the validity of the test cases. With the continuous increase of genetic algebra, the proportion of test cases for effective attacks continues to increase, and suitable genetic algorithm algebra control within 100 generations. However, the traditional Web fuzzing method's effective attack test data accounted for far less than the method of optimizing test cases in this paper.

Table 3. Experimental data statistics.

| Fuzzing method     | Whether the fuzzing test data is automatically generated | Number of test data | The number of test data for effective attacks |
|--------------------|---------------------------------------------------------|---------------------|---------------------------------------------|
| Traditional Web fuzzing method | No                                                      | 1297                | 717                                         |
| Method of this article | Yes                                                     |                     | 1098 1072                                   |
|                     | 20 generations                                         |                     |                                             |
|                     | 50 generations                                         |                     |                                             |
|                     | 100 generations                                        |                     |                                             |

According to the different genetic algebra in the genetic algorithm, test statistics are performed, and the attack traffic detection system obtains statistics through 629 XSS attack logs. The test cases can be fully enriched in the genetic algorithm, and the original data can be reached in the 50 generations of genetic algorithm. It can be obtained by analyzing the validity of test cases. With the continuous increase of genetic algebra, the proportion of test cases for effective attacks is increasing, and the suitable genetic algorithm algebra is controlled within 100 generations. However, the traditional Web...
fuzzing method's effective attack test data accounted for far less than the method of optimizing test cases in this paper.

5. Conclusions
In the process of Web fuzzing, the genetic algorithm is used to make the test cases of different types of Web attacks in the fuzzing have independent domains, and the probability management is used to propose new solutions to Web security.

This paper proposes an optimization method for fuzzy test cases based on genetic algorithm. First, use the attack traffic detection system to classify and analyze the attack types of the public network access logs, pass the generated various attack elements to the genetic algorithm module, set a suitable fitness function, and perform xml analysis on the website response content to obtain each The value of the fitness function of the test case, and finally the effect statistics of the genetic algorithm are obtained by analyzing the high proportion of the fitness function. Compared with traditional fuzzing detection methods, the method used in this article can comprehensively and efficiently detect the security of Web programs.

Acknowledgments
This work was funded by the National Natural Science Foundation of China (61521003) and the National Key Research and Development Project (Grant No.2018YF0804003, and Grant No.2017YFB0803204), and thanks to the National Digital Exchange System Engineering Technology Research Center for funding.

References
[1] 2019 China Internet Network Security Report / National Computer Network Emergency Technology Coordination Center. – Beijing: People's Posts and Telecommunications Publishing House, 2020.7
[2] Saxena P, Hanna S, Poosankam P, et al. FLAX: Systematic Discovery of Client-side Validation Vulnerabilities in Rich Web Applications[C]/NDss. 2010.
[3] Garn B, Kapsalis I, Simos D E, et al. On the applicability of combinatorial testing to web application security testing: a case study[C]/Proceedings of the 2014 Workshop on Joining AcademiA and Industry Contributions to Test Automation and Model-Based Testing. 2014: 16-21.
[4] Najera-Gutierrez G, Ansari J A. Web Penetration Testing with Kali Linux: Explore the methods and tools of ethical hacking with Kali Linux[M]. Packt Publishing Ltd, 2018.
[5] Qu S, Zhang Z, et al. Security Testing Method of Mimic Web Server Based on Fuzzy Testing Technology[C]/Application Research of Computers. 2021.11.
[6] Simos D E, Garn B, Zivanovic J, et al. Practical Combinatorial Testing for XSS Detection using Locally Optimized Attack Models[C]/2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). IEEE, 2019: 122-130.
[7] Liu M, Zhang B, Chen W, et al. A survey of exploitation and detection methods of XSS vulnerabilities[J]. IEEE Access, 2019, 7: 182004-182016.
[8] Sarmah U, Bhattacharyya D K, Kalita J K. A survey of detection methods for XSS attacks[J]. Journal of Network and Computer Applications, 2018, 118: 113-143.
[9] Zheng Y, Zhang X. Path sensitive static analysis of web applications for remote code execution vulnerability detection[C]/2013 35th International Conference on Software Engineering (ICSE). IEEE, 2013: 652-661.
[10] Medeiros I, Neves N, Correia M. Detecting and removing web application vulnerabilities with static analysis and data mining[J]. IEEE Transactions on Reliability, 2015, 65(1): 54-69.
[11] Duchene F, Rawat S, Richier J L, et al. KameleonFuzz: evolutionary fuzzing for black-box XSS detection[C]/Proceedings of the 4th ACM conference on Data and application security and privacy. 2014: 37-48.
[12] Cheng C, ZHOU Y. Finnding XSS Vulnerabilities Based on Fuzzing Test and Genetic Algorithm[J]. Computer science, 2016: S1.

[13] Hydara I, Sultan A B M, Zulzalil H, et al. Cross-site scripting detection based on an enhanced genetic algorithm[J]. Indian Journal of Science and Technology, 2015, 8(30): 1-7.

[14] Słowik A, Białko M. Modified version of roulette selection for evolution algorithms—the fan selection[C]//International Conference on Artificial Intelligence and Soft Computing. Springer, Berlin, Heidelberg, 2004: 474-479.

[15] Blickle T. Tournament selection[J]. Evolutionary computation, 2000, 1: 181-186.