Analysis of Software Vulnerability

CHUNGUANG KUANG, QING MIAO, HUA CHEN
Department of Software
Beijing Institute of System Engineering
P.O. Box 9702, Beijing
CHINA

Abstract: Software vulnerability is the fault that can be viciously used to harm security of software system. In order to decrease the harm, vulnerability analysis can be used to find security problems of software system as early as possible, and related measures, such as correctness, avoidance, may be planed. Vulnerability analysis is divided into two types, one is static analysis of vulnerability, the other is dynamic analysis of vulnerability. In this paper, we will introduce libFunction, which is a kind of dynamic analysis of vulnerability. LibFunction runs on redhat linux. It analyzes vulnerabilities related to library function. The main functions of libFunction are testing the behavior of application program when the return value of the called function is abnormal, and assessing the behavior.

Key-Words: Vulnerability, Vulnerability Analysis, Library Function, Software, Security, Static Analysis, Dynamic Analysis

1. Introduction

Software vulnerability is the fault that can be viciously used to harm security of software system. The faults are introduced when software system is designed, developed, and used. Although the research of software project has provided many methods to insure the quality of software system, there are still some faults in software system due to the complexity of design, development, and application environment. consequently, there are still some vulnerabilities in software system.

In order to decrease the harm that vulnerability does to security of software system, on one hand, new technologies should be found to decrease faults in software system, on the other hand, vulnerability analysis can be used to find security problems of software system as early as possible, and related measures, such as correctness, avoidance, may be planed to increase the security level of software system.

Vulnerability analysis is divided into two types, one is static analysis of vulnerability, the other is dynamic analysis of vulnerability. Static analysis of vulnerability refers to analysis of vulnerability that applied to source code. The theory of static analysis of vulnerability is simple, by researching, static analysis techniques try to find the vulnerability patterns of source code level, then check the source code to know whether the source code contains known vulnerability patterns or not. The principle is simple, but the realization is complex. Further more, one method can only deal with one problem of one language, a lot of methods must be integrated to develop a powerful tool. Another defect of static analysis of vulnerability is that it can only be applied to source code. It can do nothing if there is no related source code. Dynamic analysis of vulnerability is superior to static analysis of vulnerability in the way, because dynamic analysis is not applied to source code, but applied to object code that is running. There are many kinds of dynamic analysis of vulnerability. In this paper, we will introduce libFunction, which is a kind of dynamic analysis of

vulnerability. LibFunction runs on redhat linux. It analyzes vulnerabilities related to library function. The main functions of libFunction are testing the behavior of application program when the return value of the called function is abnormal, and assessing the behavior. In order to be practical, the original library function must be modified to return abnormal value with possibility of 1 when test is on and to execute normally when test is off, consequently, libFunction has the functions of searching library function, modifying library function, and compiling library function.

2. Detailed Design of libFunction

2.1 Introduction

We usually call library functions when we develop application programs. There are two kinds of return value for library functions, one is normal, and the other is abnormal. The return value is normal when the called library function is executed successfully, and the return value is abnormal when the called library function is executed unsuccessfully. Take the function (void *malloc(size_t size)) as an example, if a piece of required memory with length of size is allocated, the begin address of the piece of memory is returned. If a piece of required memory with length of size has not been allocated because of insufficient memory or other reasons, NULL (0) is returned. There may be several kinds of abnormal return values that represent different abnormal cases. All possible return values of the called library function should be considered when an application program calls a library function, which is not the reality. The reality is as follows, only the normal case is considered when an application program calls a library function, or all the abnormal cases are not considered when an application program calls a library function. The faults that are introduced when the software system is developed are harm. For example, if only the normal case is considered when the function (void *malloc(size_t size)) is called, i.e. a piece of required memory with length of size is allocated successfully, the begin address of the piece of memory is returned, and use the piece of memory afterwards. It is fine when the reality is good, but how about the bad reality that the return value is NULL. It is obviously dangerous that an application program uses the memory with address of 0. At the same time, the faults that are introduced when the software system is developed can't be avoided. We should find the sort of faults by analysis of vulnerability. The sort of faults is exposed with low possibility when the computer system is running normally, because the library function returns abnormal value with low possibility. In order to be practical, the original library function must be modified to return abnormal value with possibility of 1 when test is on and to execute normally when test is off. LibFunction does as follows, It gets the source code that implements the function of library functions, find the accurate position that implements the function of the tested library function, modify the part of the source code, compile the source code and generate a dynamically linked library glibc, replace the old dynamically linked library glibc with the new dynamically linked library glibc. A dynamically generated reference file will be generated according to the test requirements when an application program is tested. If the tested application program calls a tested library function, the new dynamically linked library glibc will decide whether to return abnormal value or to execute normally according to the reference file. Further more, the new dynamically linked library glibc will decide which abnormal value will be returned when an abnormal value should be returned. LibFunction will assess the behavior of the tested application program after the test is over.

figure 1 is the structure graph of libFunction.

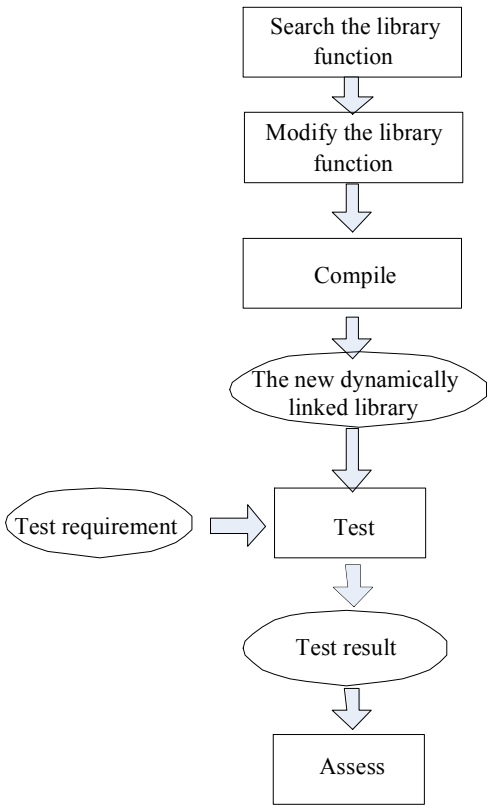

Fig.1

LibFunction has two interfaces with the outside world. They are

1. User interface: used to collect requirement information of user, such as the definition form of library function, search path, search pattern, the accurate position that the library function will be modified, the tested application program, the tested library function, the type of abnormal return value of the tested library function, the position where an abnormal value will be returned, position pattern, the IP address of the computer where the tested application program locates at, and so on.

2. Output interface: used to output assess report

LibFunction has four inside interfaces, they are

1. The interface between the module of searching library function and the module of modifying library function: the output of the module of searching library function, i.e. the file that the searched library function is implemented and the position where the searched library function locates at in the file, is the input of the module of modifying library function.

2. The interface between the module of modifying library function and the module of compiling library function: the output of the module of modifying library function, i.e. the file that has been modified, is the input of the module of compiling library function.

3. The interface between the module of compiling library function and the module of testing application program: the output of the module of compiling library function, i.e. The new dynamically linked library glibc, is the input of the module of testing application program.

4. The interface between the module of testing application program and the module of assessing: the output of the module of testing application program, i.e. the result of the test, is the input of the module of assessing.

The detailed information of all the modules is given as follows.

2.2 The module of searching library function

The main function of the module is searching the file that the searched library function is implemented in and the position where the searched library function locates at in the file. Its control flow and data flow are as follows:

1. Generate the graph interface of user. The definition form of the searched library function can be formed manually or selected from a list. The searched path has been initialized, but the user can modify it. The search pattern must be selected from a list.

2. Collect the user information.

3. Judge whether the user information conforms to the standard or not after user confirms to search. If the user information does not conform to the standard, some related messages are displayed, otherwise, execute the following steps.

4. Decompose the definition form of the

searched library function, for example, "void *malloc(size_t size)" will be decomposed into "void", "*", "malloc", "(", "size_t", "size", and ")".

5. Extract the name of the searched library function.

6. Generate the required regular expression for the first search, search the possible files that implement the function of the searched library function according to the regular expression, the search is limited to the searched path.

7. Generate the required regular expression for the second search, search the possible position that implements the function of the searched library function according to the regular expression, the search is limited to the files found in the previous step. If some possible positions are found, the related file name and the accurate position are displayed.

2.3 The module of modifying library function

The main function of the module is modifying the file that the searched library function is implemented according to the related reference file. Its control flow and data flow are as follows:

1. Accept the search result of the module of searching library function, and display it in the graph interface of user.

2. It is possible that there exist several results. user can select the exact file and the exact position after exampling more detailed information.

3. The exact file and the exact position are accepted. If the file has not been modified, a backup of the file is generated, and some required codes for modifying are added in the front of the file. If the file has been modified, no backup of the file is generated, and no code is added in the front of the file.

4. Modify the file in the exact position according to the reference file named return_type.h, All the abnormal return values of library function are contained in return_type.h.

2.4 The module of compiling library function

The main function of the module is compiling the modified files. A new dynamically linked library glibc is generated. Its control flow and data flow are as follows:

1. Get the required path for compiling.

2. Compile the modified files. If there exist some compiling errors, related messages are displayed in the graph interface of user, otherwise, the user is noticed that compiling finished successfully.

2.5 The module of testing application program

The module is consisted of two parts, one is named server part, the other is named client part. The main function of the module is testing the behavior of application program when the return value of the called function is abnormal. Its control flow and data flow are as follows:

1. Generate the graph interface of user of the server part.
3. Start the server part.
4. The server part is querying for a request from the client part after a ServerSocket is initialized by the server part.
5. Generate the graph interface of user of the client part.
6. Some parameters can be selected from lists, such as the tested application program, the tested library function, the type of abnormal return value of the tested library function, the position where an abnormal value will be returned (the tested application program may call the tested library function many times in one execution, the position where an abnormal value will be returned refers to the number of one call or numbers of several calls), position pattern (position pattern is divided into two types, one is single, the other is consecutive. The single one means that only one call gets abnormal return value in one execution of the tested application program, and the other calls are executed normally. The consecutive one means that several consecutive calls get abnormal return value in one execution of the tested application program, and the other calls are executed normally), and so on, the position where an

abnormal value will be returned can be formed manually, the IP address of the computer where the tested application program locates at has been initialized, but the user can modify it.

7. Collect the user information of the client part.
8. Initialize a Socket in the client part.
9. The client part sends the name of the tested application program to the server part.
10. The server part decides weather need to do some preparation work or not when the name of the tested application program has been detected. Do some preparation work if needed. For example, if the tested application program is wu-ftpd, the server part will check weather the server process of wu-ftpd is on or is not, the server process of wu-ftpd will be started if it is off. The server part will notice the client part that the server part is ready when the server part is ready.
11. The client part send the name of the tested library function, the type of abnormal return value of the tested library function, the position where an abnormal value will be returned and position pattern to the server part after the client part has been noticed that the server part is ready.
12. After knowing the name of the tested library function, the type of abnormal return value of the tested library function, the position where an abnormal value will be returned and position pattern that sent by the client part, the server part generates a reference file named config.h and clears a file named call_times.txt. Call_times.txt is used to record data related to the tested library function that called by one execution of the tested application program. Config.h contains the tested library function, the type of abnormal return value of the tested library function and the position where an abnormal value will be returned.
13. The tested application program is started as soon as the client part knows that the server part has generated config.h and call_times.txt has been cleared.
14. The client part tries to know the behavior of the tested application program, and produce the test result.
15. If another test is needed, repeat the steps from (5) to (13).

2.6 The module of assessing

The main function of the module is assessing the performance of the tested application program according to the result produced by the module of testing of application program. Its control flow and data flow are as follows:

1. Get the result produced by the module of testing of application program.

2. Assess the performance of the tested application program according to the result produced by the module of testing of application program.

3. Test result

Authors have tested several application programs using libFunction. Take wu-ftpd 2.6.1 as an example, part of the test result is shown below:

1. May 26 06:58:41 kcgnew ftpd[2033]: exiting on signal 11: Segmentation fault:::ftp CALLOC NULL 27

2. May 26 06:59:03 kcgnew ftpd[2143]: exiting on signal 11: Segmentation fault:::ftp CALLOC NULL 38

3. May 26 07:03:09 kcgnew ftpd[3328]: exiting on signal 11: Segmentation fault:::ftp MALLOC NULL 78

4. May 26 07:03:19 kcgnew ftpd[3378]: exiting on signal 11: Segmentation fault:::ftp MALLOC NULL 83

5. May 26 07:13:01 kcgnew ftpd[6105]: exiting on signal 11: Segmentation fault:::ftp SETEUID -1 1

6. May 26 07:13:03 kcgnew ftpd[6115]: exiting on signal 11: Segmentation fault:::ftp SETEUID -1 2

7. May 26 07:24:49 kcgnew ftpd[8422]: exiting on signal 11: Segmentation fault:::ftp GETMNTENT NULL 1

Kcgnew is the name of the computer where the

tested application program locates. 2033 is the process number. The tested application program may call the tested library function many times in one execution, 27 refers to the number of one call. It is shown that there are some faults related to library function in wu-ftpd 2.6.1. Correctness is needed.

4. Conclusion

There are some faults in software system due to the complexity of design, development, and application environment. Consequently, there are some vulnerabilities in software system. Vulnerability analysis can be used to find security problems of software system as early as possible, and related measures, such as correctness, avoidance, may be planed to increase the security level of software system. The method of vulnerability analysis is a kind of effective method to increase the security of software system. the search of vulnerability is useful. LibFunction described in this paper is only an attempt of the authors in research about vulnerability, hope to make a deeper research afterwards, and develop some more practical tools.

References

[1] Giovanni Vigna, University of California Santa Barbara, Testing and analysis,

[2] Ivan Krsul, Mahesh Tripunitara , Eugene Spafford, COAST Laboratory Purdue University West Lafayette, IN 47907-1398, Computer Vulnerability Analysis

[3] James Newsome, Carnegie Mellon University, Dawn Song, Carnegie Mellon University, Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software