Journal of Software Engineering and Applications, 2014, 7, 1-5
Published Online January 2014 (http://www.scirp.org/journal/jsea)
http://dx.doi.org/10.4236/jsea.2014.71001

The Current and Future of Software Securities and Vulnerabilities

Cuixue Zhang[1,2], Meijiao Zhou[1], Yalian Xie[2], Xiangli Li[1]

[1]School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai, China; [2]National Engineering Research Centre Industrial Process Automation, Shanghai Institute of Process Automation Instrumentation (SIPAI), Shanghai, China.
Email: lanyushan123@126.com

Received November 15th, 2013; revised December 13th, 2013; accepted December 21st, 2013

ABSTRACT

As it has been stepping into the e-time period, software, which is considered as the key factor of the network and computer development, has become an integral part of everyday life. Millions of people may perform transaction through internet, mobile phone, ATM, and send e-mails, handle word processing or spreadsheets for different purposes. In another word, the network and information have been related to our daily life completely. Then, by IT advancing, the awareness of software security becomes a hot and serious topic. This paper will give some comments in various aspects, such as, in the beginning of the SDLC (System Development Life Cycle), how do designers analyze the functional and non-functional requirements and choose the proper development model? And then the testing professors take which kinds of methods to test the software with white-box testing or black-box testing to discover the vulnerabilities and flaws. At the same time, the paper gives some examples to demonstrate why the security of software is pretty important and what we should do to secure that. In addition, the paper will talk something about the enterprises' actions to build a more secure network environment.

KEYWORDS

e-Time; SDLC; Software Security; White-Box Testing and Black-Box Testing

1. Introduction

These years more and more news of information leakage or systems intruded springs up, which nearly threats citizens' private data, such as ID and his like. A recent article, "Why is software so bad", indicates that the network in our daily life constructed with kinds of software is not very safe [1]. At any time, we may divulge our vital private information or enterprise resources, facing the threat of life safety or properties loss. The bad software may derive from various areas, the lack of awareness of security for programming and development, the inexperienced testers' testing with lazy algorithm and bad test cases, etc.

In the early 1970s, the concept of computer security was first studied. As the fast development of high technology, *In the Report of the Presidential Commission on Critical Infrastructure Protection*, it demonstrates that application developers with strong fundamental knowledge and logical mind are imperative and necessary to protect users' data and build safe surroundings of network [2].

In September 2013, *China Internet Security Conference* was held in Beijing. It is said that "*Prevention is futile in* 2020: *protect information via pervasive monitoring and collective intelligence*". It means that new or evolutionary and more effective method should be proposed. Just as what Heshuan Wu professor said in the ISC (China Internet Security Conference)," *Security accompanies with development, and the development of the internet similarly puts many new propositions forward for information security. Meanwhile, the progress of the information security technology also opens up new issues. The work of protecting or intruding information is racing, forever no end, and innovation is the unique permanent solution*". The following states and summarizes several

traditional development models or methods and tries its best to find innovatory ones.

2. Securities and Reliabilities

On generally, the software security includes Secure Programming, Reverse Engineering, Loopholes, Cryptography, software Protection and so on. Here we put emphasis on the intrinsic safety, arising from the automation industry, which refers to the software itself hold the abilities to handle various accidents and intrusions. Look out; it must be distinguished from software reliability, the ability to finish the specified tasks in the specified time, and functional safety [3], focusing on the control of the integral performance.

Since, there are various models to design and develop a kind of software. Generally, the more typical include incremental model, rapid prototype model, spiral model, fountain model and intelligent model as well as hybrid model. Although so many choices, we should take advantage of strengths and weaknesses of each model and make the most effective decision.

Furthermore, to meet the requirement of users, it is not only the model, but also the programming language (C/C++, VB, JAVA or Python etc.) and exploitation platform (.NET or Eclipse etc.). Additionally, it is the same important for developers to hold ample experience and sound programming habits. Here we just talk about one development model, spiral model, to demonstrate the steps in the whole SDLC.

The spiral development model consists of waterfall model and rapid prototype model. It emphasizes on the risk analysis and is particularly suitable for large complicated systems. It is an example of an iterative approach that represents the software process as a set of interleaved activities, allowing activities to be evaluated repeatedly [4]. The model was presented by Barry Boehm in his 1988 paper entitled *A Spiral Model of Software Development and Enhancement* [5]. The spiral model is shown in **Figure 1**.

It is noted that this life cycle provides more flexibility than other traditional predecessors.

Software should have to hold the following characteristics: availability, accuracy, authenticity, confidentiality, *integrity*, *possession* [2]. *In fact, some development teams* exploit several different methods concurrently to make the hybrid model of their own. The team should select the most suitable software development model, based on the currently specific product features, reducing the disadvantage of the selected model and making full use of its advantages. Hence, if to develop secure systems with such high requirement, we can consider the Aspect-Oriented Risk-Driven Development (AORDD) methodology additionally [6].

3. Vulnerabilities and Flaws

The vulnerability is noted that where there are flaws about the hardware, software, the concrete implementation of agreement or the system safety strategy, hackers or crackers can access or damage the system, unauthorized. Here we put something about the software holes,

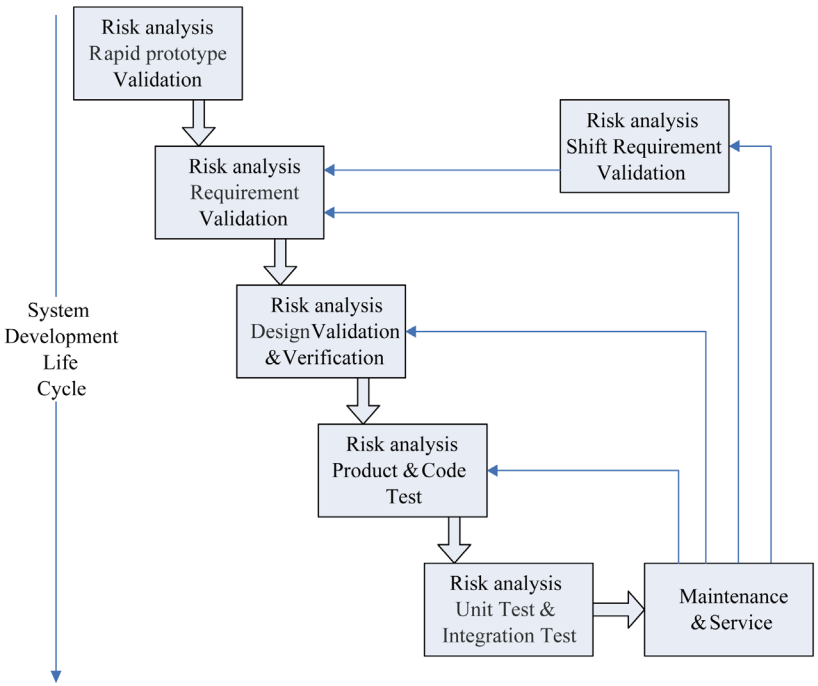

Figure 1. Spiral model.

a kind of bug. With the development and in-depth applications of information technology, software system is becoming more and more complicated and huge. No matter which kind of programming language and development model you choose, it may come across vulnerabilities or holes that lead the system to suffer intentional or unintentional intrusion.

However, vulnerabilities or flaws are various for different reasons and they may be used by malicious code and lead to separated and unexpected disasters. Especially, vulnerabilities won't come out automatically and consciously and must be discovered artificially. Mostly, they are discovered by crackers with ranged attack or internal attack. To avoid great loss, we have to distinguish vulnerabilities and give corresponding patches timely.

3.1. Vulnerabilities from Source Code

Whether the server programs, the client software or operating system, as long as it is written in code, there are different levels of bug. And it generally includes the following categories:

- *Buffer Overflow and Memory Leak*

It is inferred that when the programmer overlooks the use of extra-long string and don't restrict the boundary of buffer in a function process while users give extra-long input, it will lead to buffer overflow. In general, there are problems about the character array and function pointer. Attacks to the buffer overflow may much easier and more flexible. Just like the following instance:

```
/* vulprog */
int main (int argc, char * argv [])
{
Void (* fp) (char *) = (void (*) (char *)) &puts;
char buff [256];
strcpy (buff, argc [1]);
fp(argc [2]);
exit (1);
}
```

- *Memory leak*

Memory leak usually indicates that the applied space or applied address space are forgot to release. As procedure goes by, it tends to leak, because of memory decreasing. So, it is fatal to allocate and release memory legitimately.

3.2. Improper System Configuration

When we install systems, we often follow the default sets, easy to use. In fact, it similarly means easy to break because the default configuration mostly holds a low level of security. Sometimes, the programmers forget to close temporary and testing ports, leaving system vulnerable. In addition, in case of unsecure device in your LAN or alliance, attackers may intrude your device through other trusted partner device.

In a word, it is vital to set up a sound system configuration firstly in order to develop secure software or network conditions.

3.3. Unencrypted Data Communication of Sniffer

SNIFF is just interceptor and contains server sniffer and remote sniffer, software sniffer. If your data is transferred in the network without complicated encryption, it is easily to be captured by others.

3.4. The Defect of Design

At the beginning of the design, the defect may come from the non-logical analysis of the software requirement and the improper choice about design model. In fact, the program protocol, TCP/IP, contains holes, such as Smurf-intrusion, SYNflood, IP address spoofing and his like.

3.5. Implementation Bug

Some common vulnerability types based upon their Common Weakness Enumeration (CWE) descriptions. Here we provide some implementation bug vulnerabilities types found in the projects. *SQL Injection* (CWE-89) vulnerabilities occur when user input is not correctly validated and the input is directly used in a database query. *Path manipulation* vulnerabilities occur when users are allowed to view files or folders outside. *Command injection vulnerability* occurs when input from the user is directly executed [7].

There are kinds of vulnerabilities. When define the dangerous level of software, we prefer leaks number and loopholes density to determine the "vulnerability factors" in the software security evaluation index, both indicating safety risk of software more clearly and completely [8].

As time goes by, more and more system vulnerabilities will be discovered, it is necessary for users to update the corresponding procedure patches. Recently, it turns to be more difficult to prevent hackers' intrusion because of the mysterious multi-platform virus and the technology of AET [9]. No matter what, the flaws exist always and the pursuit of perfection is just on the way.

4. Testing and Loophole

There are some differences between testing and loophole. In general, the common methods of testing are static test and active test, which are used in the software design, before delivered. However, loophole is a kind of technology, which is used to mine the flaws and patch them as to the in service software.

4.1. Software Testing Type

Considering whether the procedure runs or not, the approaches of software testing include static testing and active testing.

- *Static Analysis*

The static analysis method is focused on the application code, carrying out comprehensive, direct scanning and extracting the key words and grammars. Interpret their meaning and study the behaviour of the procedure. Moreover, detect system bugs strictly according to the preset features and concerned safety standards. Put it simply, the static testing can be asserted below categories: Code Inspection, Static Structural Analysis, and Code Quality Metrics.

However, the static analysis only can find the know holes with clear features.

- *Active Testing*

Compare to static testing, active testing needs to compile and run the procedure. Briefly, it includes black box and white box. Black box, named function test, refers that test Software without considering the internal structure and characteristics of external characteristic. White box, named structure test, refers that design test cases and test the path and process of program based on the program's internal structure and logic design. In the real test, specific methods contain Memory Map, Non-executive Stack, and Sand Box.

4.2. Statement about Loophole

For a high requirement of safety system, the later work of loophole and version updating is more important. Likewise, holes mining technology is considered which is based on source code or based on the target code. The following are some executed concrete approaches in vulnerability discovery.

- *The distributed demand-driven*

It leverages how end users use the software to increase the coverage of essential paths [10]. The proposed system consists of many client sites and one testing site. The software under test is installed at each client site. Whenever a new path is about to be exercised by a user input, it will be sent to the testing site for security testing. The testing site has to analyse the execution trace for vulnerabilities detection. We organize the bit sequences of tested execution paths as a binary decision tree (BDT). Each non-leaf node in the tree corresponds to a program branching point. Mark relevant signatures according to the testing site.

- *Based on the analysis of intrusion records*

Vendors can evaluate the software vulnerability by analysing the records of the attacks on the security holes or attack the software as hackers to discover unknown holes [11]. Spare some effort to detect the execution software and issue patches timely.

- *Systematic manual penetration testing*

One vulnerability discovery technique, proposed by Smith and Williams [12], suggests using the functional requirements specifications of the software system to systematically generate security tests to surface security vulnerabilities. They create these tests by breaking the systems functional requirements statements into distinct phrase types such "Action Phrase" and "Object Phrase," and using these two phrase types, propose a systematic method of generating security tests using common patterns.

5. The Loss of Software Vulnerabilities

Though, there being various flaws or holes about our software and systems, if they make no harm to our life we still needn't take so much attention to finish perfect design and take almost all tools or methods to test or maintain the software. In fact, the loss is shocking.

About in 2000, in Huawei, a small flaw, just a sentence was written if (value = null) instead of if (value == null), which leaded the user community communication outage over 40 hours, and direct or indirect loss more than 3 million dollars. This little mistake in the programming almost has broken the extension of the overseas market.

Relatively, the private information being attacked illegally is more frightening, just like the recently news "Check Hotel" and "Prism Door" as well as "Aurora Event of Google".

June 2013, the prism event was announced by Edward, the ex-stuff of CIA (Central Intelligence Agency), which astonished many countries. The national security agency and the FBI entered Microsoft servers, Google servers, Apple servers and as their like, giant IT companies, to gain American, even other peoples' private information and scanned them. Let take no comments about the USA behaviours, which similarly demonstrates that our network is so unsafe and perhaps we are monitored or intruded anytime by ulterior organization or personnel.

Faced on such many attacks and intrusions, the developers should have to spare no effort to develop firm softwares and try hard to build a safer network environment.

6. Development of Safety-Related Software in Enterprises

Since software security has been a heating topic, especially in IT related enterprises and automation industry. The software developers and vendors have to devote more assets and talents to design and develop safer products.

Just as what Hongyi Zhou's, CEO of Qihu360, put the

lecture in the ISC (China Internet Security Conference), "*Free Security is Rebuilding and Expanding Security Industry*".

Terminal security will become more and more important in the future. In order to ensure the security, enterprises may take more attention about a new concept of security, cloud security, as to the unknown threats like ATP (Advanced Persistent Threat) and 0 Day. The coming development trend of the enterprise security more depends on cloud security and the "boundary" to implement. At the same time, he announced the mysterious product, 360 Eye. He also addressed that a Generic Security may become a trend, because" *it is Impossible to achieve the real and forever security of network. Just as it had been the safest shield and the sharpest spear*." (More information on http://isc.360.cn/index.html).

As to the network safety and mobile security as well as big data period, IBM also provides much measurement or new technologies and release a series of security products, like "QRadar Risk Manager," Network Activity Collectors," and "IBM InfoSphereGuardium," etc. (More information on http://www.cbinews.com/topic/2013/05/IBM_fenghui/).

7. Conclusion and Prediction

At such a network age, information security has to be the most important factors and software securities must be the related and key part. It is said that *in* 2020, *enterprise IT departments will not own the device, and in the case of cloud-based services, they may or may not control the network, server, OS or application* [13]. As the coming of the age of big data and smart-cloud, information must become the focal point in such a war of information security strategies. Someday, the way of traditional office work may turn to BYOD (bring your own devices). It is on the way that People-Centric security instead of Control-Centric approaches to information safety. Additionally, rapid detection and response about security program will be emphasized rather than traditional prevention.

Nowadays, MT (mobile terminal) is becoming more intelligent and portable and it has been the tendency. Crank calls and junk massages turn to be new unsafety, disturbing citizens' life. One day, the software and information security may be equal to national safety and personal safety, then corresponding national and international laws will be more considerable and comprehensive.

This work is particularly directed by Dr. Xie, a senior engineer. And 360's engineers supplied much help by technology exchanging platform.

REFERENCES

[1] C. Banerjee and S. K. Pandey, "Software Security Rules: SDLC Perspective," (*IJCSIS*) *International Journal of Computer Science and Information Security*, Vol. 6, No. 1, 2009.

[2] C. Y. Lester, "A Practical Application of Software Security in an Undergraduate Software Engineering Course," *IJCSI International Journal of Computer Science Issues*, Vol. 7, No. 3, 2010.

[3] H.-Y. Sun and X.-C. Shi, "The Relationship Research between Reliability, Safety and Functional Security," 2010.

[4] A. Sumithra and Dr E. Ramraj, "A Checklist Based Framework for Software Security Risk Management," *International Journal of Computing Technologies and Applications*, Vol. 2, No. 2, pp. 304-308.

[5] B. Boehm, "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, Vol. 21, No. 5, 1988, pp. 61-72. http://dx.doi.org/10.1109/2.59

[6] R. S. Gaykar and D. S. Joshi, "Enhancement of Software Security Through Design Phase," *Résumé S. Gaykar et al./International Journal of Engineering Science and Technology* (*IJEST*), Vol. 3, No. 4, 2011.

[7] A. Austin, C. Holmgren and L. Williams, "A Comparison of the Efficiency and Effectiveness of Vulnerability Discovery Techniques," *Information and Software Technology*, Vol. 55, No. 1, 2013, pp. 1279-1288. http://dx.doi.org/10.1016/j.infsof.2012.11.007

[8] R. Wang, "Research on Comprehensive Evaluation Method of Application Software Security," Dalian University of Technology, Dalian, 2013.

[9] China Internet Security Conferences, CISC 360, 2013.

[10] D. Z. Zhang, D. G. Liu, C. Csallner, D. Kung and Y. Lei, "A Distributed Framework for Demand-Driven Software Vulnerability Detection," *The Journal of Systems and Software*, G Model, JSS-9220.

[11] M. Kimura, "Software Vulnerability: Definition, Modeling, and Practical Evaluation for E-Mail Transfer Software," *International Journal of Pressure Vessels and Piping*, Vol. 83, 2006, pp. 256-261. http://dx.doi.org/10.1016/j.ijpvp.2006.02.003

[12] B. Smith and L. Williams, "Systematizing Security Test Planning Using Functional Requirements Phrases," Technical Report TR-2011-5, North Carolina State University, Raleigh, 2011.

[13] 360 Internet Security Centre, Featuring Research from Gartner, "Development Trend of Enterprise Security in the Internet Age," 2013.