Security Functional Testing of Distributed software

DONGHONG LI¹, SHUANGHUI YI¹
¹Beijing Institute of System and Engineering
P. O. Box 9702-19, Beijing 100101
CHINA
¹dongdong1202@126.com

Abstract: This paper describes a distributed software security testing methods and testing process, according to the characteristics of distributed software, the paper describes the security functional testing indicators, and a system of security indicators of distributed software, and the methods and means of detection of each metrics.

Key-Words: Security testing, Security metrics, Security indicators system

1 Introduction
With the rapid development of Internet, distributed software development and distributed software applications is becoming increasingly popular. This approach is conducive to reuse of resources, on the other hand it inevitably leads to more software security issues.

Software security is very important. In the previous era of stand-alone, security problem is the operating system vulnerable to viruses, security problem of stand-alone application software is not prominent. However, since the popularity of the network, the software security problem appears more prominent, making the importance of software security testing rose to an unprecedented height. Especially for distributed software, each node directly affects the security of the entire system's security. If a node's software security features is attacked, all nodes can be attacked, the security of software is at risk. Therefore, improving software security plays an important role in guaranting the normal operation and improving software efficiency.

2 Security testing methods and processes
Simply, the software security problem is divided into the following five levels of security: physical layer security, network layer security, operating system level security, application layer security and the third parties security. These levels of security are indispensable. In practice, application layer security is our focus. Application layer security problems are mainly caused by software vulnerabilities. These vulnerabilities can be design flaws or programming issues, and even the developer reserved the back door. In particular, many distributed software development and maintenance are of the same department. They know more about the system and have a certain authority, more easily invade and attack. Therefore, software security testing is essential for the user to ensure the security of software and the user may have a comprehensive understanding, and have more methods to improve software reliability.

2.1 Software security testing methods
For less demanding security software, security testing can be mixed in unit testing, integration testing and system testing. However, for software of high requirements on security, you must do special security testing, in order to prevent software attacks and identify software security problems.

Security testing is to validate the application level of security and identify potential security flaws. The purpose of security testing is to find the existence of security risks, and check the application's ability to prevent illegal invasion. For example, the security testing is to validate what methods should be taken to prevent unauthorized internal or external users to access or vandalism etc.

Currently, there are many test methods can be taken for security, the main methods are:

- Static code security testing: Through scanning the source code, comparing the program data flow, control flow, semantic information with software security rules, we can find out the code for potential security vulnerabilities. Static code testing should be taken in the early code development phase, rather than testing
- Dynamic penetration testing: Penetration testing is commonly used in security testing. It
is the use of automated tools or manual methods of analog input hackers, attacks on the application to find out the running time of the existence of security vulnerabilities.

- Data scanning process: Data scanning tool is usually a memory test, we can find many kind of vulnerability, such as buffer overflow. But memory information must require special scanning tools, it is difficult by hand.

2.2 Software security testing process
Software security testing process is divided into two areas: the reverse security testing process and the positive security testing process.

2.2.1 Reverse security testing
Most tests are based on software security flaws space. First we inspect the possible risks, then we test software for these risks. Therefore, the testing process is the reverse: starting from the defect space, establishing the threat model defect, finding out the invasion through the known vulnerability scan testing. This method has the advantage of the known defect analysis, to avoid software defects in the presence of a known type, but it is powerless for the unknown means and methods of attack. Reverse the process of safety testing includes four steps:
1. The establishment of the threat model defect
2. Search for the invasion point
3. Known vulnerability scanning
4. Verification test matrix invasion

2.2.2 Security testing positive
Normally, security testing positive include six steps.
1. Identification test space
2. Precise definition of design space
3. Design space definition of the conversion rules
4. Identify potential safety problems
5. The establishment of the invasion matrix
6. Validation invasion matrix

3 Security functional testing strategy
3.1 Security functional testing requirements
For distributed software, the focus of software security features is identification of security indicators. These indicators are Identification and authentication, user data protection, password support, security audits, communication security and security management. To identify and distinguish, we use Identification and authentication as an example to descriptr the testing strategy.

Identification and authentication include authentication fails, definition of user attributes, user authentication, user identification etc, the testing requirements are shown in Table 1.

<table>
<thead>
<tr>
<th>Security function</th>
<th>Description of function</th>
<th>Testing requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication failure</td>
<td>System should define how many times authentication attempts, as well as what actions should be taken when authentication failed. Authentication attempt includes the attempting times and the waiting gate time.</td>
<td>1. Using the correct user name and password to login 2. Using the wrong username or password or login many times, until the attempting times and the waiting gate time; 3. Testing system whether taking appropriate action (such as: termination of dialogue, suggesting that re-enter, etc.) and so on</td>
</tr>
<tr>
<td>Definition of user attributes</td>
<td>System is able to grant different users of different security attributes set, so that users can use the corresponding security capabilities and maintain security attributes</td>
<td>1. Login in as administrator, and set different user-level security attributes for different level user; 2. Login in as administrator, and modify security properties for different users; 3. Login in as normal user, and test whether users can use the security properties; and so on</td>
</tr>
</tbody>
</table>

| User authentication | User authentication provide a variety of authentication mechanisms for user, including identification timing, self-identification | 1. Login as different user, before user authentication, testing whether system can provide some help information, such as "how to deal with forgetting..." |
### Security Function

<table>
<thead>
<tr>
<th>Security function</th>
<th>Description of function</th>
<th>Testing requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>User identification</td>
<td>before action, non-faking identification, only one-time authentication mechanism, multiple authentication mechanisms and feedback of protected identification etc</td>
<td>password, etc., to test identification timing; 2 Login as a user group, use the security features which is not set for the group, test self-identification; and so on</td>
</tr>
</tbody>
</table>

The security testing methods of user data protection, password support, security audits, communication security and security management can be carried out by such a table. The third column "testing requirements" can be expanded according to specific needs.

#### 3.2 Functional testing of distributed software security strategy

According to various security standards combined with the characteristics of distributed applications, we have established the security indicator system see the graph 1. In this system, the second level security indicators include authentication, access control, object reuse, marking, auditing and data protection. The third level metrics include the action of authentication failure, the definition of user attributes and so on, shown in Figure 1. From the figure, we note two level indicators is not a one to many relationship, but many to many relationship, that is, the second level indicators have multiple the third level metrics, a third level metrics can be measured a number of the second level indicators. For example, the definition of user attributes is a metrics of authentication, access control and marking indicators.

![Security indicators system](image)

**Figure 1 Security indicators system**

According to each the third level indicator, we provide several testing methods and detection means. The following describe the testing methods of authentication failure action as an example.

- **Test methods:** when it fails in authentication, we test what action the system should be operated, such as stopping the session, limiting illegal login, and logout automatically etc. We can login the system directly to test whether the system provide authentication measures, whether the system have some restrictions on the number of illegal login; or use a password cracking tool to attack the system in order to observe the system response.

- **Detect means:** when it fails in authentication, we detect the system response. It include three means: ending the login process; making the login account invalid; recording audit events. We use the brute force password tool to test the system response and check the action of the above three means manually.

#### 4 Conclusion

More and more attention is pay to software security as an important sub-field of the software field. Any a small software security vulnerability of any node of distributed software is likely to be exploited, so the entire distributed software is the target of attacking. Therefore, software security testing is
playing an increasingly important role in software
development, procurement and use. The security
indicators system we have established is an initial
exploration of distributed application security issues.
During the practice of application security testing,
we will continue to enrich the test methods and
detect means to make software security testing as an
important safeguard to ensure information security
means.

References:
[1] GB 17859-1999 Classified criteria for security
protection of computer information system.
[2] GB/T 18336.3-2001 Information technology:
Security techniques Evaluation criteria for IT
Applications》 Software Testing Conference –
2005