Towards Side-Effects-free Database Penetration Testing

Que Nguyet Tran Thi and Tran Khanh Dang
HCM University of Technology
Ho Chi Minh City, Vietnam
{*ttqnguyet, khanh*}*@cse.hcmut.edu.vn*

Abstract

Penetration testing is one of the most traditional and widely used techniques to detect security flaws in systems by conducting simulated-attacks to the target systems. Organizations can develop a tool based on this technique to assess their own security systems or use third party softwares. However, besides its advantages in exploring real security vulnerabilities without false results, this technique might leave side effects for the target systems such as incomplete testing, time consuming, disclosed sensitive information, etc. if it is used unwarily. Therefore, the penetration testers or the testing providers need a methodology in order for the test to be carried out more effectively in the security environment, and more importantly, make trust for the organizations as their systems will be verified. In this paper, we propose an extended and specific methodology for side-effects-free penetration testing in detection of database security flaws. In addition, based on this methodology, the proposed system architecture for a penetration testing tool to detect database security flaws in the secure environment, which is implemented in Oracle Database Server 10g/11g, will consolidate the applicability and effectiveness of our proposed methodology.

1 Introduction

Penetration testing is properly defined as the simulation of attacks like real hackers against the target systems to identify security vulnerabilities without false positive results. It provides database administrators or managers with the list of database security flaws that could be exploited by malicious users or external hackers to make preventive plans timely. Moreover, this is also a suitable means to evaluate the effectiveness of security measures in database systems such as access control policies, database auditing policies, user management policies, and so on.

However, there is no such thing as perfect security testing absolutely. Penetration testing is considered as a snap-shot of a system's security in time. It cannot ensure that a system which is safe by using penetration testing is safe completely because it only detects predefined vulnerabilities. More importantly, penetration testing can result in side effects for the target systems if it is performed without a well-controlled process or complies with no methodology. For example, in the case of incomplete testing, it can leave garbage or a vulnerable testing account which has gained higher privilege successfully after simulating attacks. Another example is that during testing, penetration testers can exceed their scopes, take advantage of their power to exploit the sensitive information of system. Therefore, we propose a methodology which requires the concrete steps to perform the penetration testing process without side-effect for the database systems in the security environment. The idea behind the methodology is that every activity during testing should be verified, monitored and audited based on policies prescribed by organizations and penetration testers should follow the pre-scripted format as dictated by the methodology. Based on this methodology, we build the detailed architecture for automated testing tool and implement a prototype for this architecture.

In this paper, firstly we review research results related to the penetration testing in the section 2. After that, we take a brief overview about the common penetration testing methodology and analyze the problems that can happen during testing in detection of database security flaws if it is not managed

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, volume: 1, number: 1, pp. 72-85

well. In the section 3, the methodology is proposed and described in details. Besides that, we present the architecture of automated tool and implement the prototype to illustrate our proposed methodology in the section 5. Finally, some concluding remarks and future works are mentioned in the last section.

2 Related Work

Penetration testing technique has been concerned widely for a long time. There are many papers presented about the penetration testing-based technique which is used to detect security flaws, vulnerabilities in network systems, operating systems, web applications and recently, database systems [20], [15], [13], [16]. Besides that, commercial and free tools such as NGSS, Nessus, Shadow database scanner, Scuba, Red-database-security products, etc., have been developing in order to help administrators to manage well their systems [1], [2]. More and more penetration tests are discovered by famous security experts in the penetration testing field such as Pete Finnigan, Alexander Kornbrust [1], and other white hat organizations, etc. to detect security flaws of the systems and announced broadly on the websites [1], [2].

We has also built the extensible framework based on using the penetration testing technique for detecting database security flaws which can be adapted to explore security flaws in any database systems [17]. In addition, the system for detecting the database security flaws and monitoring database activities of users has been researched and developed [18].

However, the problem about the security as well as the trustworthiness of the penetration testing–based tools or penetration tests has not been concerned really. Several papers and reports have discussed about its benefits and limitations as well as given some guidelines to do before performing penetration tests to explore database security flaws [9], [5], [14], [7], [6]. Some popular methodologies in the fields of network security, web application security, etc. propose solutions to audit the testing process and make the standard reports such as STAR in OSTMM methodology or OWASP [4]. Besides that, policies about testing scopes, responsibilities of testers, the agreements between two parties must be prescribed clearly before testing [9], [5]. However, they do not focus on the technical aspects which should be performed before, during and after testing in order to make sure the penetration testing process in the secure environment, and especially in the field of database security.

In this paper, we offer the methodology for the penetration testing process to be carried out in the well-controlled environment. Our methodology is extended and concretized from the common methodology [9], [5] supplemented with the extra security requirements for penetration testing in detection of database security flaws.

3 Penetration Testing and Security Problems of Usage

Penetration testing or pentesting is also a popular way to identify vulnerabilities in network systems, computer systems, software testing, or web applications [20], [15], [13], [16]. Basically, the process of penetration testing can divide into four core phrases (Fig.1): information gathering, simulated-attack generation, result analysis and cleaning up. In the information gathering phrase, pentesters start to gather as much information as possible about the target database system such as scanning TCP port, database version, server name, IP address, system identifier, etc. by using available tools or scripts [11], [8], [1], [2]. Based on this information, pentesters can make a diagnosis about the state of the target system, list the most likely flaws and then issue pentests or test scripts to attack in the simulated-attack generation phrase. The result analysis phrase will check whether that simulated-attack has succeeded or not. If it is successful, it means that a flaw identified by the penetration test exists in the system. Otherwise, it is safe from this flaw. All explored flaws are logged and summarized in the final reports to prevent and repair.

Finally, the cleaning up phrase is performed to restore the system to its form state. Before considering the

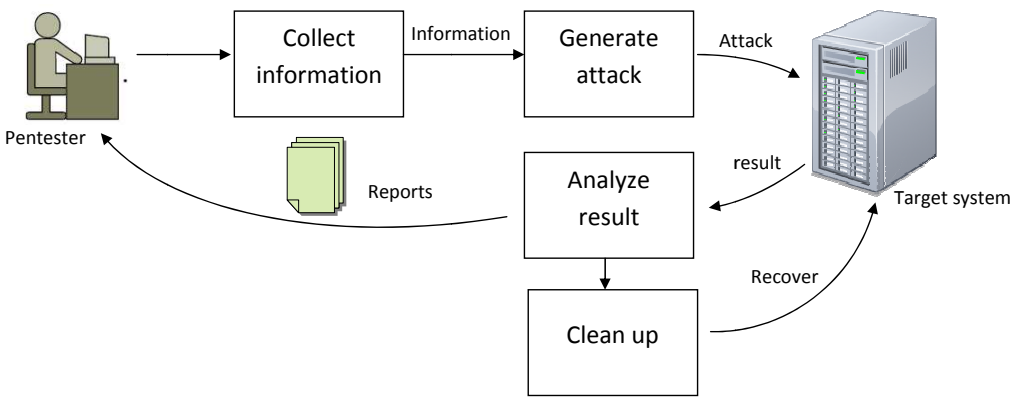

Figure 1: A common penetration testing methodology

problems of using the penetration testing, we write a simple script [18], [19] to detect a flaw of DBMS_METADATA.GET_DDL function in Oracle 10g as follows:

```
Step 1: Execute the script
-- Step1.1 Preparation
create user NORMAL_USER identified by normaluser;
grant CREATE SESSION to NORMAL_USER;
grant EXECUTE ON SYS.DBMS_METADATA.GET_DDL to
NORMAL_USER;
-- Step1.2 Penetration attempt
Connect NORMAL_USER/normaluser@targetDB;
CREATE OR REPLACE FUNCTION NORMAL_USER.ATTACK_FUNC
return varchar2
authid current_user as
pragma autonomous_transaction;
BEGIN
EXECUTE IMMEDIATE 'GRANT DBA TO NORMAL_USER';
COMMIT;
RETURN '';
END;
/
-- Inject the function in the vulnerable procedure
BEGIN
SELECT
SYS.DBMS_METADATA.GET_DDL('''||NORMAL_USER.ATTACK_FUNC
()||''','') FROM dual;
END;
/
-- the penetration attempt is finished.
-- Step 1.3 check if NORMAL_USER has dba role
```

```
connect system/PassTargetDB@targetDB;
declare
num NUMBER;
begin
select count(*) into num from DBA_ROLE_PRIVS where
granted_role = 'DBA' and grantee = 'NORMAL_USER';
WRITE_RESULT(num);
end;
/
Step 2: Clean up
drop user NORMAL_USER CASCADE;
exit;
```

The above script includes several steps to attack into database targetDB by using the simulated user NORMAL_USER. NORMAL_USER creates a function ATTACK_FUNC which tries to grant the DBA privilege to him and then injects into GET_DLL, the system function of Oracle DBMS. After that, penetration tester checks whether NORMAL_USER has the DBA privilege or not. Finally, he drops NORMAL_USER to restore the database system to its form state.

The problems of using penetration testing will be analyzed according to four basic requirements violated [17] for the security system, described as follows:

3.1 Confidentiality

Pentesting is a simulated-attack technique. It collects sensitive information of the target database systems in the first phrase, for example, the information about database name, port, IP Address, as well as user account which the pentesting process would like to test, and so on. Besides that, the way to attack the database system also needs to be protected from unauthorized users. For this reason, the organization must have the access control mechanism in the target database systems for users and data objects that are involved in the penetration testing process.

3.2 Integrity

During the pentesting process, simulated users and/or temporary objects as means to attack are created or modified (for example, user NORMAL_USER, function ATTACK_FUNC, function GET_DLL in the above example). However, these users and/or objects such as tables, procedures, functions, etc. can be modified by other sessions at the same time with the pentesting process. For example, after injecting SQL into a vulnerable procedure, the simulated user may gain higher privileges successfully. At that time, the user who is using pentesting technique to test their database systems can take advantage of this account to exploit the secret data of organization. Conflicts as well as data disclosure can happen if the current objects in the pentesting process are also used accidently in the business operations. For example, the pentesting process is inserting new records into the existing tables and other users will read these tables before the system recovery of the pentest is performed.

If the penetration testing process terminates immediately for a certain reason or the cleaning up phrase is not done completely by the lack of experience of pentesters, the target database system will be in a mess with temporary objects and/or subjects. A real hacker or even internal employers will exploit this negligence to perform malicious behavior afterwards.

Therefore, the pentesting process needs to prevent such objects and users from being modified or connected in other sessions. Moreover, it is necessary to check the system recovery after the termination of testing to restore the system to its form state.

3.3 Availability

The penetration testing imitates a real attack to check whether the flaw exists in the database system or not. There are many attack types such as SQL Injection, Denial of Service, Buffer overflow, etc. [10]. These attacks can make the system disrupted and data lost if the penetration testing process is not controlled carefully to be able to stop timely before or right as the disruption happens, especially denial of service attacks. Also, if the penetration tester predicts what will happen as well as the risk level of the next step for the end-users to prepare plans or make a decision that whether the testing should be continued or not. By this way, the risks are reduced significantly.

3.4 Non-repudiation

Pentesting can leave unavoidable risks because of the dangerous nature of attack. Therefore, the penetration testing process needs to prove the origin of database changes such as who performed as well as what were done during testing in order to make reports, response plans as well as provide criminal evidences in the case of the target database systems damaged by pentesting.

If a penetration testing process is reckless of danger, the security requirement for a database system such above will be breached out. Therefore, a new loophole could take place right after a penetration test has been completed.

4 Towards Side Effect-free Penetration Testing Methodology for Database Security Systems

The proposed methodology of database penetration testing focuses on three aspects: people, penetration testing process and policies. It suggests the model of separated roles that take part in the penetration testing process. It also shows the difference segments of penetration testing along with the specific requirements about person and techniques that should satisfy with in each phrase of the process. Besides that, the methodology highlights the policies that enforce the penetration testing process working well under the secure environment.

4.1 Separation of duties

To prevent a single person from defrauding the organization, our methodology requires that the organization should be divided into small security groups which are involved in the penetration testing process as follows:

- Risk assessment group: has the main responsibility to identify which targets and risks should be tested, prioritize them for the penetration testing group. They also provide the essential information if any for the penetration testing group or the penetration testing program to perform a test. Normally, they reply on policies of the organization or criterion that show that which data is sensitive, critical or valuable, which regulation such as HIPAA, Sarbanes Oxley, need to be complied with, which users are suspicious, which objects have the ability to be damaged, etc. to set the scanning policies for the penetration testing group or automated testing program.

- Penetration testing group (or penetration testers): is a group of specialized testers. They has the main responsibility to collect information of systems, write scripts, try to simulate attacks into the target database systems and send reports of detected flaws to security system managers. In the case of outsourcing or using third party tools, penetration testing group is considered as developers or providers of the penetration testing program for the organizations. Risk assessment group will

play a role of penetration testing group to perform pentests which are provided by the third party organizations.

- Security system managers: are charged with taking care of individual assessments, make plans before testing, resolve the issues after testing and work as a bridge between the penetration testing group, risk assessment group and target owners. For example, they find the best time to conduct a test, set policies to define scopes for penetration testing group, manage flaws that are discovered, review the reports of detected vulnerabilities, etc.

The reason of separation of duties is that penetration testing is the sensitive process. If the organizations delegate all powers to penetration testers without a good management, they will take advantage of penetration testing techniques to exploit the system as analyzed in the section3. Therefore, our methodology defines clearly responsibilities and limitations of each team to mange them easily. Penetration tester do not have the privilege to simulate attacks into all aspects of the system . Risk assessment group does not have the privilege to define the way to attack into systems. Security system manager just have the privilege to manage the penetration testing process in the high level without understanding deeply the issues related to penetration testing techniques.

4.2 Penetration Testing Methodology

As shown in figure 2, we define the penetration testing process in the database security including phrases along with the requirements for each phrase which are more specific than the common methodology [4], [5].

There are seven phrases in this methodology, including planning & preparation, information gathering, vulnerability discovery, attack generation, result analysis & report generator, clean up, and system review.

- Planning & preparation: Before penetration testers carry out penetration tests on the database systems, a great deal and a clear plan need to be done. The organization's objectives as well as agreements about scopes, authority, time, duration of penetration testing must be clarified with penetration testers. In this phrase, security system managers set policies for penetration testers such as scopes, the permitted scanning period, the maximum duration of testing, etc. Risk assessment group set policies to scan flaws in the database systems such as choosing targets, listing the greatest security risks, and so on for penetration testers to discover vulnerabilities in the system.

- Information gathering: There are two kinds of penetration testing, black box and white box. If penetration testers perform the attack with no prior knowledge (black-box), they will use available tools or scripts to collect information such as Nmap, SQLPing, Osql, tnscmd10g.pl, etc. Otherwise, the risk assessment group provides them the necessary information to test. All information in this phrase is very sensitive. Therefore, the organization should set the limitation for the penetration testers, install the access control into the target database systems if any and require them to document what they have gathered.

- Vulnerability discovery: Using the information found in the previous step, penetration testers can make a diagnosis about the state of the target system, specify scripts to test, and list the most likely flaws. They also determine to perform which penetration tests, give the estimated time as well as possible risks and proposed solutions associated with the penetration tests. Moreover, the script content is very sensitive because it contains the algorithm to discover the vulnerability in the systems. Penetration testers can write the abnormal code to serve their malicious purpose. Therefore, these scripts should be signed or confirmed by authors and will be checked the origin when necessary.

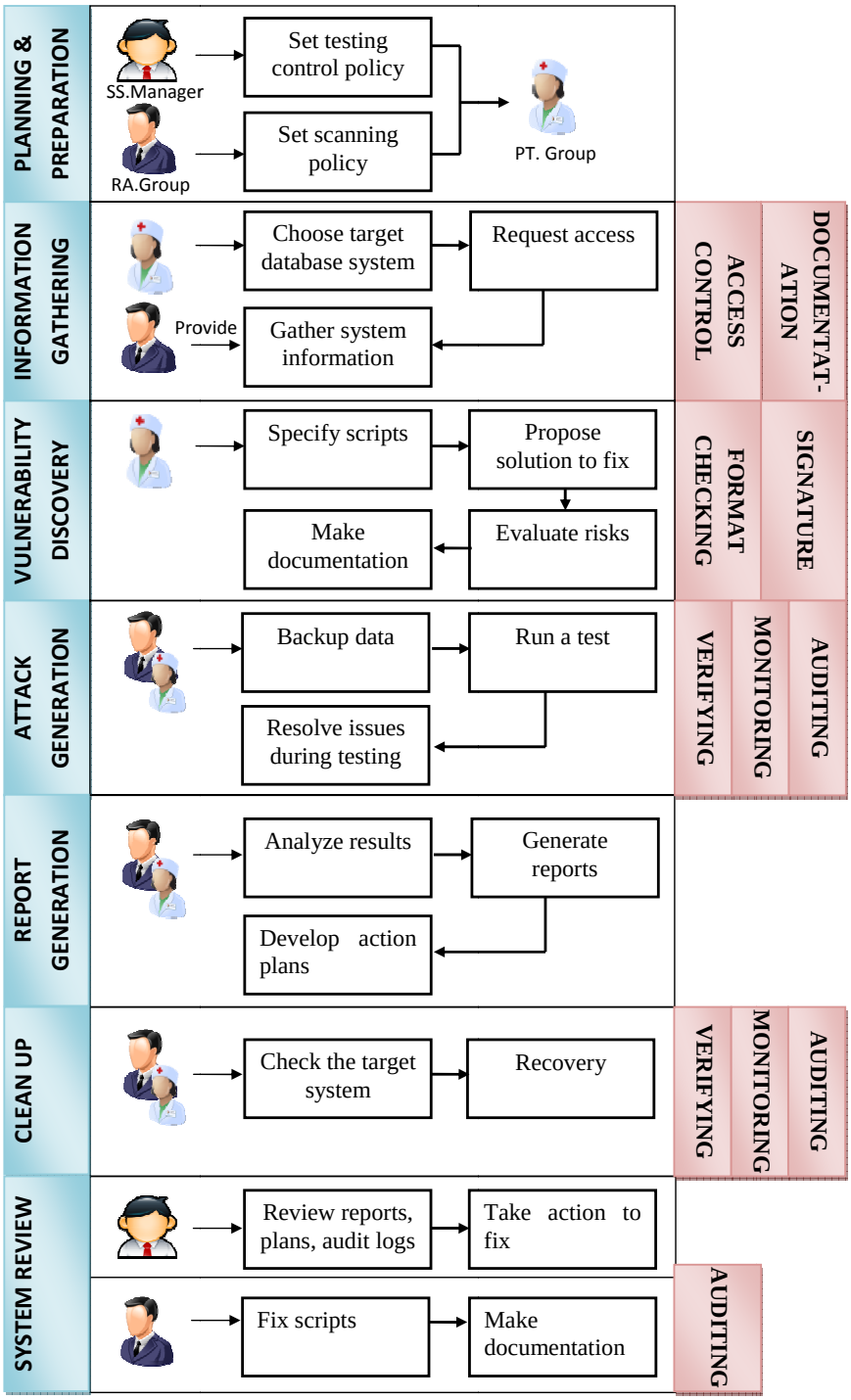

Figure 2: Extended Penetration Testing Methodology

- Attack generation: If the organizations use the third party or automated tools which are written by outsourced penetration testers, the risk assessment group will perform this task, run the program to scan the target database systems. The simulated-attack can leave problems during testing such as

make database disrupted, access sensitive information illegally, leave temporal objects, violate the predefined policies, etc. For this reason, the organization or the program should provide mechanisms to manage the penetration testing process such as verifying the script content before running, monitoring the process, auditing what the process has caused on the target database systems. If any violation of policies is found, the penetration testing process must be halted. Besides that, a mechanism to prevent conflicts between objects being tested and being used by other processes during the scanning time should be established.

- Result analysis & report generator: If a simulated attack is successful, it means that a flaw identified by such penetration test exists in the system. Otherwise, it is safe from this flaw. The list of detected flaws is summarized in reports. The final report for the vulnerabilities should also contain several suggested solutions to fix these flaws. Besides that, the information about flaws, testing processes and risk level should be attached in the final report. Based on the reports, the organization and the groups will discuss together and give countermeasures as well as plans for repairing and estimated cost.

- Clean up: During testing, objects or subjects such as procedures, tables, views, or accounts as means are created for the penetration testing process. For example, a simulated- account used to perform a test must be restored in the end of the test. The reason is that real hackers can exploit such accounts, which already gain access successfully after attacking into the system. Therefore, the cleaning up phrase must be done completely and carefully to restore the system to its form state. This phrase should be monitored and audited because the risk assessment group or penetration testers can take carelessness or deliberately as recovering the database system.

- System review: This is the last phrase in the penetration testing methodology. The security system managers evaluate the vulnerabilities of the target systems based on the reports, and give the action plans to resolve them. As allowed, the risk assessment group will fix flaws by using solutions offered by the penetration testers or the testing program. The activities should be also audited and the target systems should be checked again after repairing to make sure that the vulnerabilities have been resolved.

4.3 Penetration Testing Control Policy

In our methodology, we propose the testing system should have three kinds of policy at least, that is, the testing control policy, the scanning policy, and the auditing policy.

- Testing control policy: The policy is set by the security system managers to define the scopes of pentests, the limitations of penetration testers, the valid scanning period, the access control of sensitive information, and so on.

- Scanning policy: The policy is set by the risk assessment group to decide which target database systems will be checked, specify which risks should be verified, determine the time to perform the pentests, who has the responsibility for scanning, etc.

- Auditing policy: To prevent the system performance from being reduced significantly, the auditing policy is used to describe what should be audited such as which database activities, which users, which objects, etc. instead of auditing all database activities unnecessarily.

Three above policies are necessary to control the penetration testing process, prevent misuse of penetration testers, resolve conflicts during testing, and provide evidences to support finding the cause of database system disruption if any during testing.

4.4 Penetration Testing Control Mechanism

The organization should implement four mechanisms to enforce the policies working well: access control mechanism, verifying scripts, monitoring the testing process, and auditing process. Scripts should be verified to find out the abnormal code or the attack code violated policies. Each activity or each step of penetests should be monitored continuously in order to find out conflicts or risks during testing, and then send alerts to managers or administrators to solve timely.

5 Towards an Architecture of Automated Penetration Testing

In this part, we give the architecture that provides the automated penetration testing process to perform in the security environment and it is based on the proposed methodology. In which, end-users can verify, prevent conflicts, audit what happened during the penetration testing process and check the system recovery after testing. Besides that, the separation of duties is applied into our design to reduce the potential damage from intentional actions of end-users.

5.1 Architecture

With the analyzed above problems along with our proposed methodology, we build the program, which performs penetration tests in the security environment for detecting database security flaws. It satisfies with the following requirements:

- Separation of roles in the system, namely, penetration tester role (PT role), risk assessment role (RA role), and security system manager role (SM role)
- No other session can access to the same objects which are being modified by the pentesting process
- No other account can be logged into the database system if it is being used by the session of pentesting
- After testing, the database system must be restored to its former state
- Activities during the pentesting process need to be audited to comply with the non-repudiation requirement of security

The main idea is dividing a test into steps in which RA role can specify their properties as well as the risk level of them. The program will verify the script content of each step before launching and executing it. If there is any violation in that step, for example, the test script accessing to sensitive objects, the program will alert and wait for the user handler to continue or not. Besides that, triggers are created before running the test to protect objects from accessing by other database users. All of activities during testing will be audited automatically based on the auditing policy into the database of program by the popular auditing techniques.

As shown in fig.3, the high level of our architecture includes five layers: Graphical User Interface, Functions, Penetration Testing engine, Testing Control and Database.

Graphical User Interface layer provides application forms for users to interact with the system. According to the authorization mechanism, each role performs several separate functions.

Functions layer includes the services for users in the system. For example, penetration tester use flaw specification to specify flaws, write scripts, declare parameters, evaluate risk level, etc.

Penetration testing engine layer has the function to manage the penetration testing process from beginning to end. The procedure to perform a pentest includes the steps: loading the script and setting

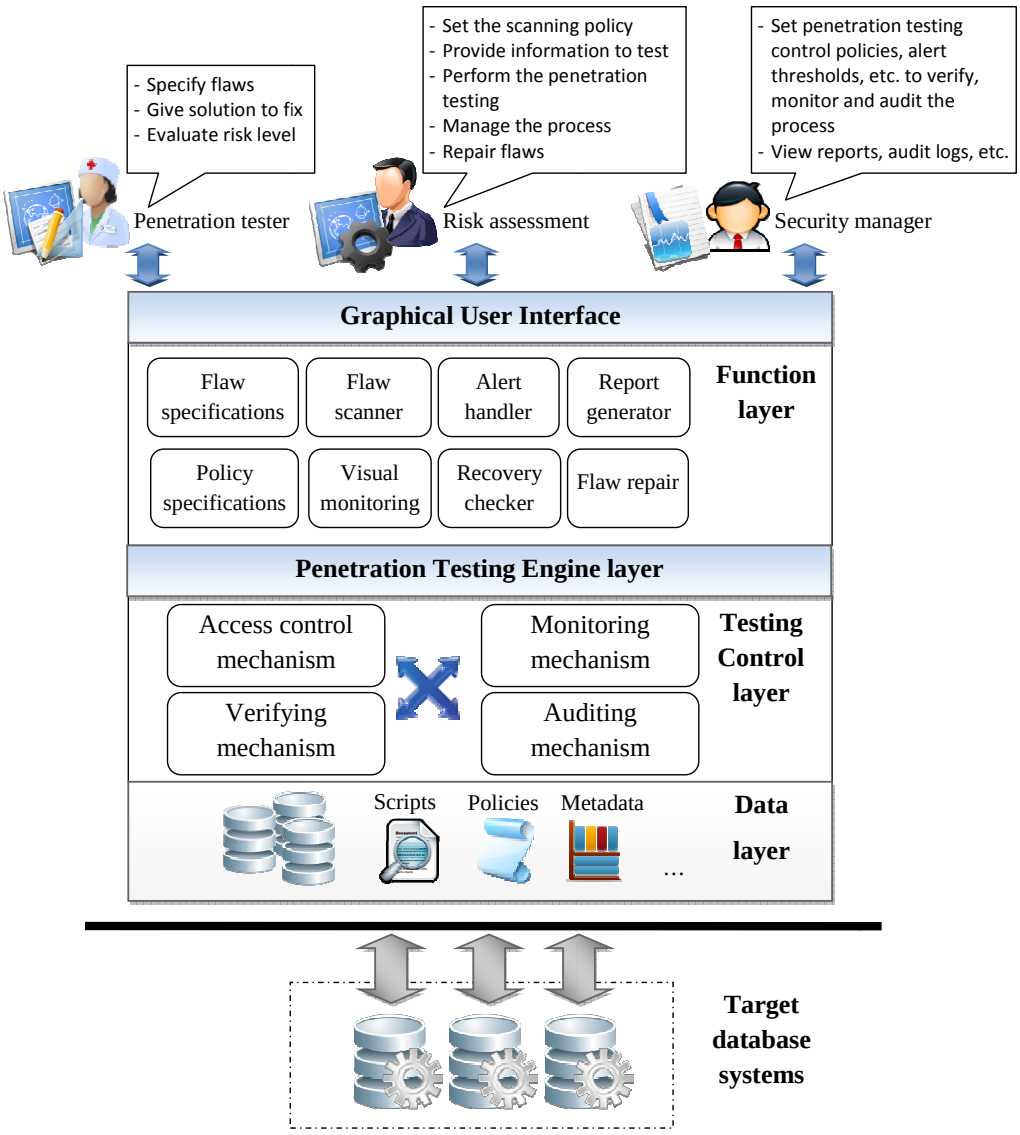

Figure 3: High-level architecture of the “safe” penetration testing process

parameter values, verifying the script content, generating the executable script, executing the script, cleaning up and generating reports. The running pentest is also monitored and audited into the program database. The penetration testing process can be halted and wait for the user handler if there has any conflict with the policies. For details, we will present in the section 5.3.

Testing control layer provides the mechanisms to authenticate the users in the system, verify the script content, monitor the penetration testing process and audit the activities during testing. Using the testing control policies, it checks the rules continuously to find out abnormal or dangerous database activities as testing, and then alerts to the penetration testing engine layer to resolve timely.

Database layer is the program database, contains the repository of flaws, policies, as well as metadata and other necessary data for the testing system.

5.2 Specification of the Penetration Testing Process

In this part, we focus mainly on the flaw specification and the definition of policies.

Flaw specification. A test process of flaw is specified into steps in the format of XML. In each step, penetration testers can define its properties such as risk level, warning, message, message format, alert type, etc. Besides that, they can add more tag elements to customize its properties. The program will display the value of these user-defined properties as running the test.// One of mandatory elements is "Content" which contains SQL code for the program to execute and attack the database system. SQL code for the test script can be retrieved from other resources such as websites about database security, published flaws, or announcement of DBMS vendors [11], [8], [1], [2] or written by pentesters. The program will prepare and create links as well as intermediate statements automatically for executing steps in succession. For the details of running a test, we will present in the next section.

Policy specification. There are many kinds of policy including the scanning policy, the testing control policy, the auditing policy section 4.3. In this part, we focus to describe the testing control policy which is used to manage the penetration testing process. Each policy includes a set of rules defined by end-users. Our architecture provides the following kinds of testing control policy.

- Context-dependent testing control policy: The rules are checked before the penetration testing runs. Information from the testing environment is taken into the checking module. Typical contextual condition including time, IP address, database instance, user privilege, etc. is defined in this policy.
- Event-based testing control policy: The rules are verified at each step of pentest before the program launches that step to execute. They include the conditions about objects, subjects, SQL command types, privileges that are allowed to grant to users in the script, as well as legal grantees as doing a certain activity to the target database system.

All conditions of rules are written according to the prescribed structure as the following examples:
IP address condition: BETWEEN <IP address> AND <IP address> for the IP range condition, and <IP address> (,<IP address>)* for the list of authorized IP addresses.
Time condition: FROM <datetime> TO <datetime> ON [Monday – Sunday] *
Object condition: <object-name> (, <object-name>)*
End-users can use the operator '!' to reverse the condition for both two kinds of security systems, the open and closed system. The program will check all user-defined rules automatically before executing SQL commands in the pentest.

5.3 Implementation of Penetration Testing Engine

This part includes the steps to perform a pentest as mentioned in the section 5.1. The details of this process are shown in the fig.4 and described logically as follows:

First, the program bases on the context-dependent testing control policy to check the context of scanning the system such as the IP address condition, the time condition, and the database instance condition. If not violated, the program will load XML specifications of chosen flaws, fill with value of parameters configured in the scanning policy. For each flaw, it will be parsed into fragments or steps. After that, the risk level of each step will be checked. If it is violated with the predefined rules, the program will notify to the risk assessment group. Otherwise, the SQL code of that step will be parsed and verified through the event-based testing control policy for the test script. If it is valid, the program will do

some things such as retrieve the previous connections, value of parameters, etc. and then generate into the executable files including the scanning file, the trigger file, and the untrigger file. The scanning file will be executed by the execute SQL engine of DBMS [17], [18]. Moreover, to comply with the data integrity and data confidentiality, the program will create triggers on objects to prevent access from other users before executing the scanning file. In the case that triggers of objects have already existed in the database, the program will get the content and merge them with the code of trigger of program if any. The untrigger file will be executed in the clean up phrase. If any violation happens after verifying, the program will

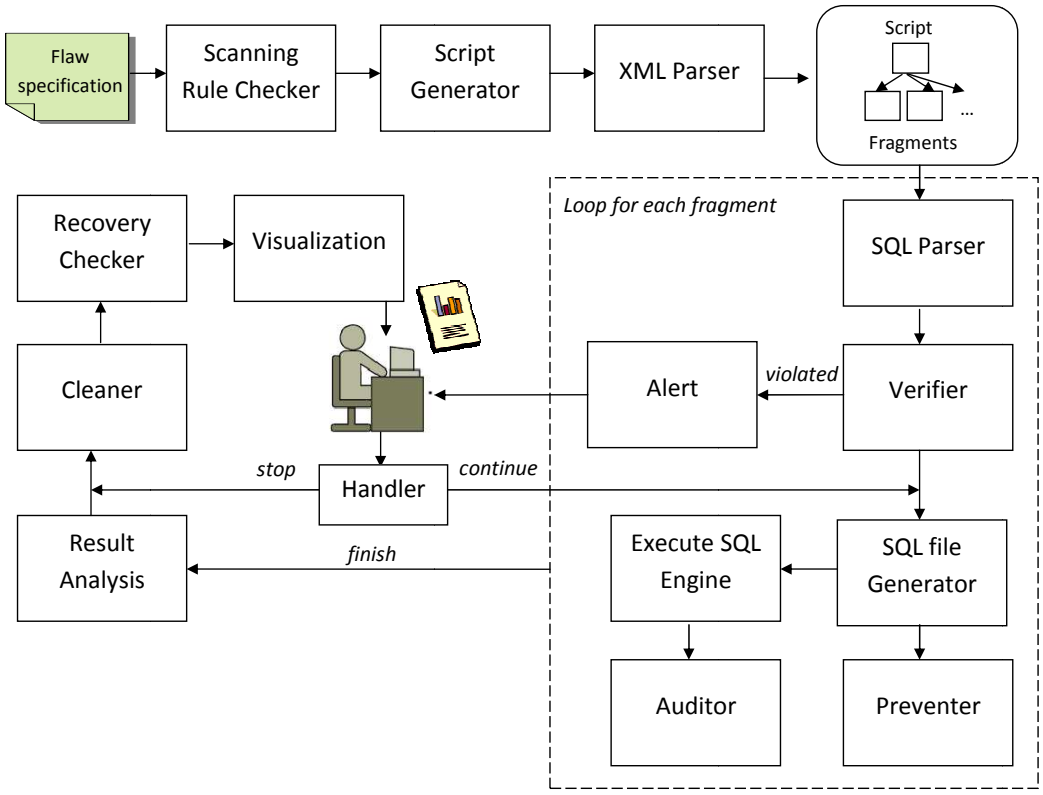

Figure 4: Implementation of the Penetration Testing Process

send a message to risk assessment group and offer several options to handle or do automatically based on the predefined rules such as stop, continue, or skip the current test, stop the scanning process or send an email. All activities during testing are audited and stored in the program database by the basic auditing techniques such as standard auditing, triggers, DBMS auditing technology (FGA in Oracle for example), etc. [12]. Therefore, whenever the pentesting process finishes or stops suddenly, the program will check database changes (recovery checker) and report the state of system recovery to users. Besides that, the reports will be summarized and processed automatically by the report generator and then sent to them.

In summary, the solution will provide the security environment for end-users to perform the pentests precisely and safely on their database systems. The system allows users to write policies for verifying the risks of testing before simulated-attacking the database system. The separation of duties is applied to manage roles and prevent a single person from defrauding the organization. During testing, other people cannot access to the testing objects to eliminate conflicts. All of database activities happened in the pentest are also audited so that the program can check the system recovery after testing as well as make reports and provide evidences to support finding the cause of database system disruption due to

testing if possible. This architecture resolved the security problems analyzed in the section3.

5.4 Evaluation

The architecture provides an approach to build the automated testing tools which can perform pentests in the security environment. The end-users can control the penetration testing process, customize their policies to monitor and audit, review the audit trails and check the recovery ability of program after testing. Moreover, conflicts between objects being tested and being used by other operations are also detected and prevented. To some extent, the given architecture for an automated testing program also meets with the regulations such SOX, HIPPAA and the basic security requirements. However, the solution has not focused on the scanning engine to maximize the performance of system yet. It proposes the simple scanning solution to perform pentests, that is, scripts are written according to the predefined structure (in this case, XML) with parameters. The scanning engine will load the scripts, set the value of parameters through the input user interface and then generate the executable scripts, and then execute them by calling the SQL console of DBMS. Moreover, the verifying algorithm is based on matching the predefined alert rules; therefore, it is limited by the scope of knowledge of people. It cannot find out the illegal access in the scripts if penetration testers use the complicated dynamic SQL commands such as the execute command combined with several variables.

6 Conclusion and Future Work

Penetration testing is the popular method to detect vulnerabilities of almost systems such as network systems, operating systems, web applications, and database systems. In this paper, we reviewed the penetration testing procedure, and deeply analyzed security problems in the use of penetration testing-based techniques against the target database system. We also proposed the penetration testing methodology to provide the guideline for developers of testing program or the organizations to detect database security flaws in the secure environment. We also offer the architecture for the automated testing tool to illustrate our methodology. The prototype of the architecture has also been implemented with .NET 2008, ANTLR 3.1 parser [3], etc. to detect security flaws in the database systems of Oracle 10g/11g.

In the future, we will improve the performance of the current system. The relationship between steps of flaws will be defined and checked before running the test. Take an example that if the result of checking flaw X is passed then a certain step of other flaws which has the relation to X will be discarded. By that way, the scanning performance of program will be increased much more. Besides that, we will extend more kinds of alert rule as well as enhance the verifying algorithm to be able to detect violations that are more complicated.

References

[1] Some database security sites. Available: `http://www.red-database-security.com`, `http://www.petefinnigan.com`, `http://www.securityfocus.com`.

[2] Some database testing tools. Available: `http://www.imperva.com`,`http://www.ngssoftware.com`, `http://www.nessus.org`.

[3] Antlr parser generator. Available: `www.antlr.org`, 2006.

[4] Osstmm 2.2. Technical report, `http://www.isecom.org/osstmm`, 2006.

[5] Y. F. Sattarova A. F. Alisherov and K. Tai-hoon. Methodology for penetration testing. *International Journal of Grid and Distributed Computing*, (ISSN: 2005-4262):43–50, 2009.

[6] J. Braden. Penetration testing - is it right for you?, 2002.

[7] National Infrastructure Security Coordination Centre. Best practice guide: Commercially available penetration testing, 2006.

[8] L. David. *The Oracle Hacker's Handbook: Hacking and Defending Oracle*. Wiley Publishing, 2007.

[9] Federal Office for Information Security (BSI). Study: A penetration testing model. Available: `https://ssl.bsi.bund.de/english/publications/studies/penetration.pdf`, 2003.

[10] S. Hansman. A taxonomy of network and computer attack methodologies. Master's thesis, Department of Computer Science and Software Engineering, University of Canterbury, New Zealand, 2003.

[11] H. John L. David, A. Chris and G. Bill. *The Database Hacker's Handbook: Defending Database Servers*. Wiley Publishing, 2005.

[12] R. B. Natan. *How to Secure and Audit Oracle 10g*. Auerbach Publications, 2009.

[13] A. Petukhov and D. Kozlov. Detecting security vulnerabilities in web applications using dynamic analysis with penetration testing. Proceedings of the Application Security Conference, 2008.

[14] M. T. Raggo. Pentesting databases. The ISSA Charlotte conference, 2008.

[15] J. Shewmaker. Introduction to network penetration testing. The 7th Annual IT Security Awareness Fair, 2008.

[16] J. S. Tiller. *A Framework For Business Value Penetration Testing*, pages 60–67. Auerbatch publications, 2005.

[17] P.H. Cu-Nguyen T.K. Dang, Q.C. Truong and T.Q.N. Tran. An extensible framework for detecting database security flaws. Proceedings of the International Workshop on Advance Computing and Applications, Ho Chi Minh City, Vietnam, 2007.

[18] T.Q.N. Tran T.K. Dang and Q.C. Truong. Security issues in housing service outsourcing model with database systems. Technical report, `http://www.cse.hcmut.edu.vn/~asis`, 2008.

[19] T.Q.N Tran. Problems of using penetration testing for detecting database security flaws. Proceedings of the 11th Conference on Science and Technology (CST2009) of HCMUT, 2009.

[20] S.R.Choudhary W. G. J. Halfond and A.Orso. Penetration testing with improved input vector identification. pages 346–355. Proceedings of the International Conference on Software Testing Verification and Validation, IEEE Computer Society, 2009.

Que Nguyet Tran Thi received the BEng degree of the faculty of CSE/HCMUT (Vietnam) in 2008. After graduation, she works at this faculty as a teaching assistant. She is also studying the master course majored in the information systems at her university. Last year, she received the scholarship for internship at University of Applied Science, Fribourg, Switzerland in 5 months. Afterwards, she came back to Vietnam, has continued working in HCMUT and finished her thesis. During the student time, she also received some scholarships such as Honda-Yes Award, Kitagawa, etc. which is for the excellent students in Vietnam.

Tran Khanh Dang received his BEng degree from the faculty of CSE/HCMUT (Vietnam) in 1998. He achieved the medal awarded for the best graduation student. From 1998-2000, he had been working as a lecturer and researcher in the same faculty. Then, he got a PhD scholarship of the Austrian Exchange Service from 2000-2003, and finished his PhD degree in May 2003 at FAW-Institute, Johannes Kepler University of Linz (Austria). Afterwards, he had been working as a lecturer and researcher at the School of Computing Science, Middlesex University in London (UK) since August 2003. In October 2005, he returned home and has continued working in HCMUT. Dr. Dang's research interests include database/information security, modern database applications, and MIS. He has published more than 60 scientific papers in international journals conferences. Dr. Dang has also participated in and managed many research/commercial projects. Currently, he is the head of the IS Department and director of Advances in Security Information Systems Lab at CSE/HCMUT.