Teaching Database Security and Auditing

Li Yang
Department of Computer Science and Engineering
University of Tennessee at Chattanooga
[Chattanooga, TN 37403
Li-Yang@utc.edu

ABSTRACT
Hands-on laboratory experiences are essential critical for students to understand concepts and gain real-world insights in database security and auditing. We are developing a set of hands-on labs to integrate theories of database security into practices. Our designed labs do not require purchasing any commercial software or pre-configuration. Each lab includes objectives, results, and resources to help students to understand database security concepts including access control, virtual private database, and database auditing etc. We use two major database products (Microsoft SQL Server and Oracle 10g) to design and implement our labs.

Categories and Subject Descriptors
K.3.2 [**Computer and Information Science Education**]: *Computer science education, Curriculum*

H.2.7 [**Database Administration**]: *Security, integrity, and protection*

General Terms
Human Factors, Management, Security

Keywords
Database Security, Auditing, Curriculum

1. INTRODUCTION
There is a growing number of database hacking incidents and sensitive data loss. In 2001, Bibliofind, a division of Amazon.com that specialized in rare and out-of-print books, was attacked and details for almost 100,000 credit cards were stolen. In March 2001, the FBI reported that almost 50 bank and retail Web sites were attacked and compromised by Russian and Ukrainian hackers [5]. Therefore, knowledge and skills about how to manage a secure database system are very important for students in information security discipline. Database security and auditing require knowledge from both database management and computer security.

SIGCSE'09, March 4–7, 2009, Chattanooga, TN, USA.

To achieve effective instruction on database security and auditing, we developed a set of hands-on labs which engage students actively and thoughtfully in learning [9]. Our labs are designed to integrate theories of database security into hands-on labs. We provide an objective, results, and information on available resources in each lab module. Our designed labs can be implemented using both Microsoft SQL Server and Oracle 10g database. This helps student to have thorough insights into two popular database systems.

2. CHALLENGES IN TEACHING DATABASE SECURITY AND AUDITING
Database security and auditing course is taught in some universities through extensive readings on research papers and essays such as "Information Security: An Integrated Collection of Essays" by M. Abrams and S. Jajodia [4]. However, students in a hands-on program remember the material better, feel a sense of accomplishment when the task is completed, and are able to transfer that experience easier to other learning situations [10]. These benefits are achieved because "more than one method of learning is accessed" in hands-on learning and "the information has a better chance of being stored in the memory for useful retrieval" [10]. Naturally, the best way to teach database security and auditing is to instruct theories of database security and then apply theories into laboratorial exercises using popular commercial database systems. Currently limited books and pedagogical materials are available to serve this purpose. This paper shows how to integrate theories of database security into practices through hands-on labs.

3. TOPICS OF DATABASE SECURITY AND AUDITING
Topics in database security and auditing include database basics, access control theories, application security, virtual private database (VPD), and database auditing. We designed seven lab modules to cover above topics. Each lab module consists of an objective, results, and available resources. The resources include books and websites that we found very useful to assist students with lab assignments.

3.1 Database Basics
The concept of database systems especially SQL query language of relational databases is revisited in the first week. Students in our class install two database systems from scratch: Oracle 10g database and Microsoft SQL Server 2003/2005. Oracle 10g database is free to download from Oracle website which provides a step-by-step installation guide. Microsoft SQL Sever can be downloaded and installed for free with MSDN license. The first lab module helps students to create a database by both writing

SQL scripts and using graphical interfaces. The tasks include creating a database schema, adding primary and foreign key constraints, and manipulating instances in the databases.

Laboratory 1: Build a Database

Objective: to let students know how to create and manipulate a database using both scripts and graphical interfaces.

Results: Students are able to

- create a database scheme with given database design
- create primary and foreign keys for relations
- instantiate the database with instances
- insert a new instance to the created database
- delete or update an existing instance
- manipulate three options (RESTRICT, CASCADE, or SET NULL) in referential integrity

Resources:

- Afyouni's book [8] provides the examples to create database schema and database instances.
- Sunderraman's book provides scripts to create database schema and instances.
- The ADbC tool [2] shows how to maintain referential integrity.

3.2 Access Control

To ensure high-level overview and meet business needs, we base database security on implementing security policies. Security policies can be implemented through access control rules. Access Control is one of the major components of databases security. Access control policies can be grouped into three major classes: Discretionary access control (DAC), Mandatory access control (MAC), and Role-based access control (RBAC). DAC policies control accesses based on the identity of the requestor and on access rules stating what requestors are (or are not) allowed. MAC policies control accesses based on mandated regulations determined by a central authority. RBAC policies control accesses depending on the roles that users have within the system and on rules stating what accesses are allowed to users in given roles.

DAC policies of a database system can be implemented by an *access matrix model* which regulates the privileges that a subject can have on an object. In order to develop an access control model we identify objects to be protected and subjects that execute activities and request access to objects, and actions that can be executed on the objects. An object can be a table, a view, a procedure or any other database object. A subject can be a user, a role, a privilege, or a module. In the access matrix model, a state of the model is defined by a triple (S, O, A), where S is a set of subjects, O is a set of objects and A is a access matrix where rows correspond to subjects and columns correspond to objects, and entry A[s, o] reports the privileges of s on o [5]. Changes to the state of a model are carried out through commands. For instance, an owner can grant to others or revoke from others, a privilege to execute an action on her files. An access matrix model can be practically implemented by an authorization table, an access control list, or a capability list generally used in DBMS systems . An access control list associates each object with a list of subjects and actions that a subject can exercise on objects. A capability list associates each user a list of objects and actions that the user is allowed to exercise on the objects. SQL Server, MySQL, Oracle Database, DB2 and Sybase support the implementation of access matrix models. One thing that database security administrator need pay attention to is the vulnerability of the discretionary policies. The fact that discretionary policies do not enforce any control on the flow of information makes it possible for processes to leak information to users not allowed to read it.

Role-based access control (RBAC) is an alternative to traditional DAC and MAC and attracts attentions from commercial applications. RBAC attempts to specify and enforce enterprise-specific security policies in a way that maps naturally to an organization's structure. Role-based policies regulate the access of users to the information based on organizational responsibilities that users have. Role-based policies break user authorization into assignment of roles to users, and assignment of roles to permission. Roles allow a user to use database systems with the least privilege required for a particular task he/she needs to perform. Users authorized to powerful roles do not need to exercise them until those privileges are actually needed. This minimizes dangers due to intruder masquerading or Trojan Horses. Separation of duties in RBAC requires that no user should be given enough privileges to misuse the system on their own. Both SQL Server and Oracle databases support concepts of roles.

One vulnerability of DAC lies in the fact that there is no control on flow of information. A Trojan Horse is one of the attacks that exploit such vulnerabilities.

Laboratory 2: Implementing DAC

This lab can be implemented in either Oracle 10g or Microsoft SQL Server. Both Oracle 10g and SQL server support concept of roles, as a result implementation of DAC can be extended to the implementation of Role-based Access Control (RBAC).

Objective: To implement database security policies using discretionary access control.

Results: Students are able to

- create users, roles, profiles, privileges.
- interpret given database security policies into an access control matrix.
- assign privileges based on users.
- assign privileges based on roles .
- understand potential vulnerabilities of DAC.

Resources: Afyouni's book provides examples about how to create users, roles, profiles, privileges and how to assign roles and privileges. They can be used as preparation of this lab.

The most common form of MAC is the multilevel security policy using classification of subjects and objects in the system. In multilevel mandatory policies, an access class is assigned to each object and subject. The access class is one element of a partially ordered set of classes. The partial order is defined by a dominance relationship. An access class consists of two components: a security level and a set of categories. The security

level is an element of a hierarchically ordered set, such as Top Secret (TS), Secret (S), Confidential (c), and Unclassified (U), where TS > S > C > U. The set of categories is a subset of an unordered set, whose elements reflect functional areas. MAC policy controls flow of information, thus preventing leakages to unauthorized subjects. The security level of access class associated with an object reflects the sensitivity of the information contained in the object, that is, the potential damage that could result from the unauthorized disclosure of the information. The security level of the access class associated with a user, also called clearance, reflects the user's trustworthiness not to disclose sensitive information to users not cleared to see it. The "Trusted Computer System Evaluation Criteria (TCSEC)", a United States Government Department of Defense (DoD) standard, has details about MAC which is also known as label security. The TCSEC, frequently referred to as the Orange Book, was replaced with the development of the "Common Criteria for Information Technology Security Evaluation (C2)" standard originally published in 2005 [2]. Most of database vendors can offer functions supporting label security through use of row-level security or fine-grained access control, which is one of advanced security features in databases. Oracle calls it Fine-Grained Access Control (FGAC). DB2 currently only supports this feature on z/OS and calls it Multi-Level Security (MLS) [1]. SQL Server only supports this feature in SQL 2005 and calls it Fine-Grained Privileges.

Laboratory 3 Countering Trojan Horse Using MAC

This lab shows how to use MAC to ensure confidentiality and control information flow. It can be implemented in either Oracle 10g or SQL Server 2005.

Objective: To make student master the concepts of MAC and Trojan Horse.

Results: Students are able to

- interpret database security policies into MAC rules
- create security levels and labels
- bind security labels with objects
- bind users or roles with security levels
- students are able to justify that their implementation can counter Trojan Horse attacks

Resources: Oracle Database 10g Tutorial [6] has "Using Oracle Label Security" that provides step by step guide to implement MAC in Oracle databases. Students can practice the tutorial first before they implement their own MAC policy.

There are two prerequisites if you use Oracle 10g to implement Label Security: First, install Oracle Database, the basic one, and create a database using database configuration assistant. Second, install Oracle Label Security by selecting the option of label security, and configure the created database in first prerequisite by adding label security. The website of Oracle has very detailed tutorials and scripts to implement label security.

3.3 Virtual private database (VPD)

Virtual private database (VPD) provides row-level access control beyond the capabilities of roles and views. The VPD is a shared database schema containing data that belongs to many different users, and each user can view or update only the data he or she owns. The VPD can ensure that online banking customers see only their own accounts. The Web-hosting companies can maintain data of multiple companies in the same database, while permitting each company to see only its own data. The VPD results in lower costs of ownership in deploying applications. Security is built once in the data server, rather than in each application that accesses data. Security is stronger because it is enforced by the database no matter how a user accesses data. VPD is a key technology that enables organizations to build hosted, web-based applications.

Microsoft SQL Server uses *VIEW* database object to implement VPD. VIEW objects are created to limit what users can access in a table, that is, a view is created to hide columns or rows from users.

Laboratory 4: Virtual Private Database (VPD) using VIEW

Objective: To implement a VPD using views in either Microsoft SQL Server or Oracle 10g database.

Results: Students are able to

- create a VIEW object based on a give condition (e.g., department = 20).
- create a VIEW object to display rows that belong only to the logged on user. The VIEW object has one additional column as current user name which is returned by a built-in function USER.
- test whether a user can only display his/her owned records by SELECT.

Resources:

- The book of Afouni has examples about implementing VPD using VIEWs.
- The ADbC tool [2] supports row-level security using VIEWs.

Oracle databases use *application context* to implement VPD. Application context is a function specific to Oracle that allows you to set database application variables that can be retrieved by database sessions. Application contexts act as secure caches of data that may be used by a fine-grained access control policy. Upon logging into the database, Oracle sets up an application context in the user's section. One can create a customized application context and attributes. For example, a clerk can access only the records of the customers who lives in a region assigned to him. But a manager can access any record. The VPD is enabled by associating one or more security policies with tables or views. Direct or indirect access to a table with an attached security policy causes the database to consult a function that implements the policy. The policy function returns an access condition known as a predicate (a `WHERE` clause), which the database appends to the user's SQL statement, thus dynamically modifying the user's data access. You can implement VPD by writing a stored procedure to append a SQL predicate to each SQL statement that controls row-level access for that statement. For example, if John Doe (who belongs to Department 9) inputs the `SELECT * FROM emp` statement, then you can use VPD to add the `WHERE DEPT = 9` clause. In this way, you use query modification to restrict data access to certain rows.

The implementation of Oracle VPD requires knowledge of PL/SQL which is a special language available for developers to code stored procedures which are integrated seamlessly with the database objects via SQL. The language, however, has far more potential than the simple commands of SQL (UPDATE, SELECT, INSERT & DELETE). The language allows for modularity and complexity of standard procedural languages. Loops and variables are permitted within this language which makes if far more useful for data extraction and manipulation than the standard SQL. PL/SQL is used to enforce the logic and business rules of the database application. If students are not familiar with PL.SQL, an additional lecture is necessary to cover triggers, procedures and functions, stored procedures and functions, packages, and built-in packages. *Oracle 10g programming: A primer* [8] is a book that provides concise coverage of basic Oracle SQL programming. It has rich source codes free to download, which help students to understand PL/SQL.

Laboratory 5: Implementing a VPD Using Oracle Application Context

Objective: To make students understand concept of application context and implement VPD through application context.

Results: Students are able to

- Create a PL/SQL package that sets context.
- Create a context and associate it with the package.
- Set the context before users retrieve data (at the login).
- Use the context in a VPD function.

Resources:

- Oracle Database 10g Tutorial [6] has step-by-step tutorial on "Restricting Data Access Using Virtual Private Database."
- The book of Afyouni also has a good example of implementing VPD through application context.

3.4 Application Security

Database applications are considered the new major problem when increasing bugs arising from programming errors in applications. SQL injection is a typical application attack as a result of insecure code. For instance, web application allows users to input text to build a query which will be executed against a database. A malicious user enters malformed text data to have commands included in the data sent to an interpreter. The malicious user can either gain access to information that he/she was not authorized, or delete or alter data in the back-end database. For example, a hacker can fill the username with *user* and password with *guess'*; **delete from table** *users* **where** *username* **like** *'%*. The database executes two SQL statements:

> **select** *user* **from** *users* **where** username='user' **and** *password* = guess';
>
> **delete from table** *users* **where** username **like** '%'

The *users* table that stores authentication information with mapping between username and password will be altered by unauthorized hackers who do not need have the knowledge of the legal username and password.

Relational databases including SQL Server, Oracle, MySQL, DB2, and Sybase are vulnerable to SQL injection attacks. Litwin [7] discussed five measures that the developers can take to prevent SQL injection attacks.

Laboratory 6: Experiencing SQL Injection

Objective: To make students understand the mechanism of SQL injection, how to find SQL injection bugs and the possible measures to prevent SQL injection exploits.

Results:

- Students are able to launch SQL injection attacks. We use Animated Database Courseware tool developed at Kennesaw State University available at http://coffee.kenesaw.edu.
- Students read extensively on SQL injection and countermeasures. A copy of reading report is required.

Resources: The ADbC tool [2] has components helping understand SQL injection attacks.

3.5 Database Auditing

There is no security without auditing, therefore security and auditing should be implemented in an integrated fashion. Auditing database activity and access can help identify security issues and resolve them quickly. Auditing as a function needs to play a central role in ensuring compliance because an audit examines the documentation of actions, practices and conduct of a business or individual. It then measures their compliance to policies, procedure, process and law. Two types of data are required to ensure compliance of the database environment. The first category includes audit trails and other logs, which is called auditing information. The second audit category involves security audits. They are sometimes called assessments, penetration tests, or vulnerability scans, and focus on the current state of a database environment rather than auditing data.

Many database auditing trails can be produced for a database environment, so we will discuss the categories of auditing. The first category of auditing that is required in most environments is a full audit trail of logon and logoff, and record all failed login attempts. The second category is to audit Data Control Language (DCL) of the database. The DCL covers changes to privileges, user/login definition and other security attributes. A complete audit trail of any changes made to the security and privilege model must be audited. The target of auditing in this category includes addition and deletion of users, logins, and roles, changes to the mappings between logins and users or role, password changes, and changes to security attributes at a server, database, statement, or object level. The third category is to audit Data Definition Language (DDL) which changes database schema. Some stealing information activities (e.g. Trojan Horse) may often involve DDL command. For example, data can be copied to additionally created table. The forth category is to audit changes to sensitive data via Data Manipulation Language (DML) activities. This auditing is particularly useful for a Sarbanes-Oxley project where accuracy of financial information is important. Through auditing DML activity we can record old and new value of sensitive records such as salaries. The fifth category is to audit changes to sources of stored procedures and triggers where malicious codes can easily hide. The sixth category is to audit database errors because attackers will make many attempts before they get it right. The last but not the least is to audit any changes made to the definition of what to audit.

An auditing solution has two indispensable parts: collecting information and using information. Auditing does not enhance security unless its trail is used. It is impossible to rely on a manual

process to ensure that all the audit reports are checked and assessed. Automation and oversight becomes important parts of a sustainable solution. A built-in oversight defines the review workflow and order, ensuring that the audit tasks are continuously activated and reviewers do not hold up the processes. Moreover, adopting an independent audit trail created by a third-party solution is a good practice to collect and use audit information because independent audit immunes to bugs and vulnerabilities that the database may have. The third party solution also helps to archived and secure audit information, and audit the audit system.

Laboratory 7: Auditing DML Action

Objective: To learn and implement DML (SELECT, DELETE and UPDATE) action auditing.
Results: Students are able to

- Create a table called AUDIT_DATA to contain an audit trail of all data change operations on the target table.
- Create a trigger on the target table. The trigger will be fired when operations (SELECT, DELETE and UPDATE) are performed against the target table.
- View contents of auditing table AUDIT_DATA.

Resources: The book of H. Afyouni [3] provides more examples on fine-grained auditing (FGA) with Oracle, implementation of history auditing and auditing application errors.

3.6 More Topics

More topics reflecting current trends in database security can also be discusses, such as database privacy [11], privacy and linking to external databases [12, 13], encrypted database, and XML database encryption and security [14, 15].

4. EVALUATION

Hand-on labs of database security help understand the material and because of that they help retain the knowledge better. For example, students comment that the audit process was taught in several classes before database security course, but the knowledge becomes useful only after they saw how the system is actually implemented. There are several important issues that we need pay attention to based on instructor and student evaluation. Firstly, being familiar with the terminology and capabilities of a database system is very important. Therefore, hands-on labs should be designed to cover most of terminology and capabilities of database security. Secondly, following the steps and doing exactly what the lab tells are easy but not very educational. It would be much better to make step-by-step labs as preparation for separate assignments which are similar but not identical to the step-by-step labs. Finally, it is important to stay up-to-date and teach about the newest releases, because the employers are constantly upgrading their software.

5. CONCLUSION

We have reported our experiences in teaching theory and practices in database security and auditing, using hands-on labs. Our designed hands-on labs aim to serve as a model repeatable in other universities. No special configuration or purchase fee is required to use our hands-on labs.

Acknowledgements: This work is supported by grant of Tennessee Higher Education Commission's Center of Excellence in Applied Computational Science and Engineering R04-1302-085, and Odor Wheeler foundation R04-1024-032.

6. REFERENCES

[1] Guimaraes, M. New challenges in teaching database security. In *Proceedings of the 3rd Annual Conference on information Security Curriculum Development*, Kennesaw, Georgia, September, 2006.

[2] Guimaraes, M. and Murray, M. Using animation courseware in the teaching of database security. In *Proceedings of the 8th ACM SIGITE Conference on information Technology Education*, Destin, Florida, USA, October, 2007.

[3] Hassan A. Afyouni. Database security and Auditing: Protecting data integrity and accessibility. Course technology, ISBN 0-619-21559-3, 2006.

[4] Marshall D. Abrams, Sushil Jajodia, and Harold J. Podell, eds. *Information Security: An Integrated Collection of Essays*, IEEE Computer Society Press, 1995. http://www.acsac.org/secshelf/book001/book001.html.

[5] Ron Ben-Natan. Implementing Database Security and Auditing, Elsevier Digital Press, 2005.

[6] Oracle by Example Series: Oracle Database 10g Tutorial: http://www.oracle.com/technology/obe/10gr2_db_single/index.htm

[7] Paul Litwin. Data Security: Stop SQL Injection Attacks Before They Stop You. MSDN. Magazine, 19(9). http://msdn.microsoft.com/en-us/magazine/cc163917.aspx

[8] Rajshekhar Sunderraman. Oracle 10g programming: A primer. Addison Wesley. ISBN: 0-321-46304-8, 2007.

[9] RUTHERFOR, F. J., Hands-on: A means to an end. 2061 Today 3(5), 1993.

[10] Haury, D. L., and P. Rillero. Perspectives of Hands-On Science Teaching. *ERIC Clearinghouse for Science, Mathematics, and Environmental Education*, Columbus, OH, 1994.

[11] Agrawal, Rakesh, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu, Hippocratic Databases, *Proceedings of VLDB Conference*, 2002.

[12] Sweeney,Latanya. K-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10 (5), 2002; 557-570.

[13] P. Samarati, Protecting respondents' identities in microdata release," *IEEE Transaction On Knowledge and Data Engineering*, Vol. 13, No. 6, 2001, pages 1010-1027.

[14] Hacigumus, Hakan, Bala Iyer, Chen Li, Sharad Mehrotra. Executing SQL over Encrypted Data in the Database-Service-Provider Model. *ACM SIGMOD*. June 4-6, 2002. pp 216-227.

[15] Hacigumus, Hakan, Bala Iyer, Sharad Mehrotra. Efficient Execution of Aggregation Queries over Encrypted Relational Databases. Database Systems for Advanced Applications (DASFAA). 2004. Lecture Notes in Computer Science (LNCS) 2973, pp. 125–136. Springer-Verlag. 2004.

[16] Wang, Hui, and Laks Lakshmanan. "Efficient Secure Query Evaluation over Encrypted XML Databases." ACM Very Large Database. Seoul, Korea. pp 127- 138, 2006.