Database Auditing

Jungha Woo, Sael Lee, and Carla Zoltowski
{wooj, lee399, cbz}@purdue.edu

Abstract
Government regulations and increased awareness of security issues have increased the auditing requirements of information technology systems. In this paper, we will discuss three government regulations and how they have impacted information technology systems. We classify database auditing systems by considering features of the basic components of an auditing system as proposed by Bishop: the logger, analyzer, and notifier. In addition, we will consider possible policy models that could be implemented. Finally we will survey three commercial database and third party auditing products according to the classification features, and discuss how they address the government regulations and general security needs.

1. Introduction

Auditing is the recording and analyzing of events or statistics to provide information about system use and performance in a clear and understandable manner [Bishop 2003]. The goal of an auditing system is to be able to determine if security and other policies are being violated. Those policies include general security requirements, as well as security and auditing requirements of governmental regulations.

Given an audit log of "who" did "what" to "which" data "when" and "how" and an effective means of processing the log, auditing can answer "why". By answering the question of "why", it provides one of the means to detect intrusions into the system, including privileged users. Another use of auditing is to provide periodic report of system usage and data modifications. Auditing can also be used to detect and recover database systems in case of system failure or human errors.

Malicious access to protected data can cause major damage to organizations and also problems for the public. Previously intrusion detection was focused on network tracking to prevent external threats to the system. However, according to study on insider threat by US Secret Service and CERT CC in 2005, 29% of the incidents of intrusion of database system are done by individuals inside organizations that have privileges to freely access data, or "insiders" [CERT 2005]. Therefore, auditing systems must include the ability to prevent and detect internal threats to the system.

Due to these corporate scandals, unauthorized access of patient information, distribution of customer information, and terrorism acts, the regulatory environment developed as acts such as the Sarbanes-Oxley Act (SOX), the Health Insurance Portability and Accountability Act (HIPPA), and the Payment Card Industry (PCI) Data Security Standard were implemented [LUM]. Several major venders of database systems such as Oracle, Microsoft, and Sybase have responded to these increased auditing needs. However, there are challenges to the database system's ability to satisfy all of the security and auditing needs, which have been addressed by the development of several third party auditing software solutions. In this paper we will explore the native capabilities of three common database management systems: Oracle 10g, Microsoft SQLServer 2005, and Sybase Adaptive Server Enterprise (ASE). We will also explore three third party auditing software products, namely Lumigent's AuditDB, AppSecurity's AppRadar, and Guardium's SQLGuard, to determine how they help address the increasing security and auditing needs of database systems.

2. Government Regulations

A number of recent government acts have impacted the auditing requirements of organizations. Some of these regulations have specifically impacted the auditing of the databases. We will consider three significant government acts and the impact they have had on database auditing requirements.

The Sarbanes-Oxley Act of 2002 (Pub. L. No. 107-204, 116 Stat. 745, also known as the Public Company Accounting Reform and Investor Protection Act of 2002) was proposed and passed in response to high profile corporate scandals such as Enron and WorldCom. The Act contains 11 titles and "is designed to reduce fraud and conflicts of interest within organizations, while increasing financial transparency and public confidence in the markets" [Ben-Natan 2006, p 4]. The Sarbanes-Oxley Act requires company executives to take responsibility for the security and validity of financial and auditing processes, controls, and reports. The Act requires all publicly-traded US companies and non-US companies doing in business in the US to comply with Sarbanes-Oxley [SO101]. Although SOX does not have specific requirements for Information Technology (IT) systems, because most organizations' data are maintained

electronically, SOX has had a significant impact on the security and auditing requirements within an organization. To be in compliance, organizations must have regular external audits, and "compliance to SOX has historically come down to the opinions within external auditor statements attesting that the proper controls are in place to ensure that financial data moves through the organization unaltered and is only exposed to the correct people in the organization" [SecInn 2006, p. 2]. Thus, organizations must assure that data cannot be accessed or altered by unauthorized individuals, while insuring that authorized individuals can access the data when needed.

Another governmental act that has had an impact on the auditing requirements of database systems has been the Health Insurance Portability and Accountability Act (HIPAA). Title I of HIPAA provides for the portability of health insurance coverage when workers change jobs, and limits restrictions that insurance plans can place on pre-existing conditions. Title II of HIPAA, which includes the "Administrative Simplification" (AS) provisions, requires entities that collect, store, and transmit protected medical information to adhere to a standard format and provide protection of that information from unauthorized entities [HIPAA1]. The security provisions require that organizations:

- "Ensure confidentiality, integrity and availability of all electronic protected health information (ePHI) that the health care entity creates, receives, transmits, or maintains.
- Prevent disclosure of any ePHI information that is not permitted or required.
- Ensure that system information is available for auditing trails.
- Ensure that authentication is in place so that specific workers or entities are who they say they are" [SecInn 2006, p4].

A third regulation that has impacted database auditing requirements has been the Payment Card Industry (PCI) Data Security Standard. The PCI Data Security Standard was developed by major credit card companies to safeguard credit card data. It applies to all members, merchants, and service providers that store, process, or transmit any credit card data. The standard contains many items, but those that apply most directly to databases are those discussing how to protect stored data, and to log and monitor all accesses to credit card data [Ben-Natan 2006]. Periodic audits are required to achieve compliance.

There are a number of techniques and technologies that can be used to address the security and auditing needs of the government acts. See Table 1 in Appendix A for a summary of those technologies.

3. Policy modeling in auditing systems

Auditing policies required by the government and private regulations can be represented in a formal manner. General policy models such as Bell-LaPadula, Clark-Wilson, and Chinese Wall Policy have been used in computer systems. These models can be applied to formally define auditing of database systems.

Bishop [Bishop 2003] represents constraints on a system as "*actions => condition*." The implication in the constraint requires that the *action* be true before the valid conclusion about the *condition* can be deduced. In the context of auditing events in computer systems, an *action* would be true if the event occurred, such as a read of an object. Bishop states that although the representation is unusual, it does allow a user to "simply list constraints against which the records can be audited." For constraints implementing security and governmental regulations, if the action is true, the result of the operation should match the satisfaction of the constraint. For a given action, if the condition is false, the operation should result in a failure. If it results in success instead, a security or regulatory violation has occurred.

We will consider three policy models that commonly provide the basis for the security models implemented in computer systems and databases: Bell-LaPadula, Clark-Wilson, and the Chinese Wall Policy.

The Bell-LaPadula policy model is a multi-level access control model developed primarily to formally model access issues related to government applications. It characterizes subjects and objects according to their security level, and does not allow subjects to read entities that are at a higher security level (more secure), or allow subjects to write to entities that are at a lower security level. The constraints that would express the above access policy are as follows:

S reads O => L(S) >= L(O)
S writes O => L(S) <= L(O)

where S = subject, O = object, and L(S) and L(O) express the security level of the subject and object, respectively. This is applicable to government regulations where access to data is defined by roles and corresponding access levels.

In order to implement this model, systems need to log the following information. For reads and writes, the system needs to log the subject's security level and compartments, and the object, its security level and compartments, and the result of the action. For variable security levels, the system needs to log the subject or object, its old and new security level and compartments, the security level and compartments of the subject changing the object, and the result.

The Clark-Wilson policy model is described by Bishop, Wee and Frank [BWF] as a commercial analogue to the governmental policies represented by Bell-LaPadula model. According the Bishop, Wee and Frank, the Clark-Wilson model was the "first to provide a basis for assessing integrity in a realistic commercial environment, and the first not to be based on the access control matrix model. Integrity assurance requires a certification process and enforcement process. " This process must first certify that system is in a secure state, and then that programs only move the system from valid (secure) state to valid state. For this model, the system must log enough information to reconstruct the operation, and ensure that they operate only on objects on which they have been certified. The system must also "log all actions which violate the enforcement rules." Enforcement rules include the requirements to control users, and that users must be authenticated. In addition, it supports the separation of duties [Clifton 2004]. To be able to determine if the policy has been violated, the system must maintain:

i. For each transformation procedure, $TP_{i,}$ a set of constrained data items, or CDIs, upon which it has been certified.
ii. For each TP_i and each user u_j, a list of CDIs that the TP_i may manipulate on u_j's behalf.
iii. Set of authenticated users.
iv. A list of the sets of users that can certify entities.

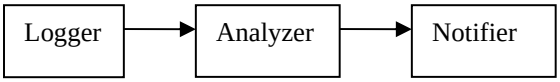
In addition to the sets described above, the system must log the user, the TP, the CDIs, and the result for each transaction. The transition-based Clark-Wilson policy model is well-suited where data integrity is a priority [Clifton 2004].

The third policy model that has been implemented in computer systems is the Chinese Wall Policy. According to Bishop, Wee, and Frank, the Chinese Wall policy "presents a more complex commercial policy in which both integrity and confidentiality are essential; it mimics the rules that stock brokerage houses use and is, in fact, patterned on the requirements of English law." To implement this model, all subjects and objects are partitioned into "conflict of interest" classes (called COIs). Each COI is partitioned into "company datasets" (called CDs).

Under this policy model, the system must log the subject and object names, when an object is sanitized, and sufficient information to enable an auditor to reconstruct membership of relevant COIs and CDs. Because of its partitioning of subjects and users into COIs, the Chinese Wall Policy model is well-suited for applications where adherence to corporate anti-trust regulations is a priority [Clifton 2004], such as applications demonstrating compliance with SOX.

4.0 Basic architecture of auditing system

The basic architecture of an auditing system is:



Anatomy of Auditing System (Bishop 2003)

The logger records information. The information that is stored in the log is determined by the security and governmental policies and the system capabilities. The analyzer takes the log as the input and analyzes it to either determine if an event of interest or a problem has occurred, or if other information needs to be logged. The notifier receives the analysis from the analyzer, and reports the result of the audit to the analyst, auditor, or other entities. In the following sections we will describe features of each component in detail and provide a framework to classify database auditing systems.

4.1 Logger

Some important issues related to logging include what, when, where, how, and how often a database should be logged. The ways in which the databases address these issues have an impact on other aspects of the system. For example, the location of the log has an impact on the parsing and retrieval efficiency (analysis and notification portions) of the system. Depending on the location, it may affect the way audit logs are browsed. If the audit log has been stored in a database, users can easily search against the database. More customized auditing procedures can be written for table-oriented logging systems. File-based audit systems can also be searched, but it is less efficient because the system will need to do additional pattern-matching jobs to do the column-based comparison.

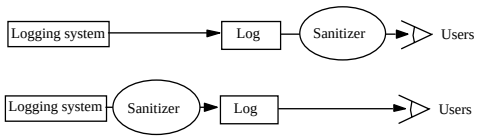
The mechanisms used to set the type of event and condition to be monitored address the "how" question of logging. In addition, systems can be classified by how they balance the need to protect sensitive data from being exposed by the audit itself, with the need to provide enough information to conduct an effective audit. A variety of sanitization and encryption techniques have been utilized by database auditing systems. Another consideration is the mechanism used to configure the database system for audit. We will get a better understanding of this configuration by studying audit rule creation semantics such as the architecture of each audit trail, and each system's logging capability.

The extent of auditing required and/or desired is the most essential factor in determining what to log

because the ability to reconstruct an event which is of interest to the analyst depends on how accurately a database has logged the data for those suspicious events. An entry in a log must contain sufficient information to find the consequence of a certain action. However, logging all data changes for all database events significantly degrades system performance without the benefit of meaningful data. Nonetheless, monitoring for critical tables and system is required.

4.2 Sanitization

As more and more attention is being paid to privacy, database solutions have to provide some ways to protect critical information such as credit card numbers and expiration dates. If a site wishes to make logs available to an external auditor, it must delete the confidential information [Bishop 2003, p. 698].



From [Bishop 2003], p. 699

This might be obtained by two different approaches. In the first approach, the sensitive information is logged, but it is sanitized before it is displayed to the users. In the second approach, the Sanitizer is deployed in between the Logger and Log system so critical information will be anonymized or encrypted so as not to reveal privacy information. Unfortunately, contemporary database do not provide any of these two solutions.

One method of privacy protection is the use of disk encryption. If a database supports privilege-based database table encryption, user information will be secured even when a malicious user steals a database disk or backup tape because of the amount of time it would require to decrypt the data. This approach may relieve database administrator's burden to keep the data secure. But we do not believe this provides a satisfactory solution since most encryption solutions in the market do not take care of audit logs. Unless the log tables are encrypted, the information in the logs is unsecured.

4.3 Analyzer

In analyzing there are two major issues: what to analyze for and when to analyze. The question, “what to analyze” considers the goals of auditing, which are usually related to the detection of security violations. Regarding “when to analyze”, the analysis can be periodic, based on transaction counting, and/or occur in real-time. There are four major goals in the analysis process:

- Detection of the need to modify policy of the system.
- Detection of defined problems or anomalies.
- Determine the accountability of actions taken by the users.
- Produce periodic reporting of system usage or data changes.

There are three categories of frequency of analysis: periodic, transition counting, and real-time [Orman 1997]. More frequent analysis can detect violations faster and allow prompt settlement. However, as analysis gets more frequent, it becomes more costly. The challenge is to find the balance between the correctness and timeliness, both of which are important. Real-time analyses analyze and validate each transaction and report immediately of the policy violation. Real-time analysis provides most frequent analysis and it can be most costly to the system and is done for policies that are critical to the system. Periodic analyses perform analysis after a given period of time. This method is most often employed to give insight of the overall system transitions or to provide periodic reporting. The transition counting method performs analysis after every n number of transitions has been made. Real-time is a special case of transition counting method where n equals one. The transition counting method is most effective in situations where the load of transition varies independent of the time frame. Most database systems and third party software products provide all three categories of frequency of analysis. Most often the frequency used is dependent on the policy of auditing and what the analyzer is looking for. It also depends on the overall architecture of the auditing system. For database systems, analysis is part of the database system, which means it will cause overhead since the log sizes are larger than the data themselves. Most often real-time analysis is very costly. For third party auditing software that has a separate system to log, analyze and report, there often is very little overhead. Thus, real-time analysis is affordable.

4.4 Notifier

There are two issues in the notification stage that are of primary interest. The first is the audit browsing techniques available. The second is the ability of the notification system to sanitize the data based on the policy and the viewer’s privilege level.

The goal of the audit browsing tools is to be able to present information in a way that it is easy for the security administrators and auditors to identify potential threats or violations of policy. Bishop

identifies six basic browsing techniques: text display, hypertext display, relational database browsing, replay, graphing, and slicing. This is a current research area.

Sanitization considered in notification phase deals with correlation of audit information with the user. Depending on the user, the level and section of viewable audit results must be altered. Before a user is allowed to view the log or result of analysis, information that is not relevant or information level that is higher than user's allowed access, is removed. There are two basic underlying principles in sanitization: separation of privilege and principle of least privilege. According to Jarry 1997 in "The protection of information in computer systems", separation of privilege is protection mechanism that requires more then one key or authentications to obtain the data or access to the system. Principle of least privilege refers to granting the least possible amount of privileges in order to enhance protection of data and functionality from faults and malicious behavior. In the context of database auditing, these principles require that users be restricted to search and view entries of a narrow scope so no user is allowed to access all entries. In addition, when accessing critical data logs, more than one privileged user must agree.

4.5 Application to Database Systems

The auditing system that Bishop describes is in relation to file or network systems. When applying this auditing architecture to databases, there are a number of differences to keep in mind [Ram 2006]. First of all, databases are data-rich by nature, and thus more objects must be protected in a database. This can impact the amount of information that must be logged. Secondly, the lifetime of the data is normally longer in a database than for an operating system, so again maintaining auditing records for the lifetime of the data can also impact the system. Third, database security is concerned with differing levels of granularity, such as file, record, or field, whereas operating systems protect real resources. That requires more flexibility in access control mechanisms, and also in auditing constraints. Fourth, in database systems the objects can be complex logical structures, a number of which can map to the same physical data objects. The auditing and security features must be able to accommodate this many-to-one mapping. Fifth, the different architectural levels – internal, conceptual, and external – have different security requirements which must be represented in the constraints and the logs. Last, database security is concerned with the semantics of data, as well as with its physical representation.

5.0 Survey of auditing systems in current database systems

The next two sections survey both commercial database systems and third party products related to the classification of issues discussed in Section 4. In this section we will explore the native auditing features of three commercial database systems: Oracle 10g, Sybase Adaptive Server Enterprise (ASE), and Microsoft SQL Server 2005. This exploration will reveal how the native auditing features of these databases are insufficient to address all of the security and auditing requirements for these systems. In response to these needs, several third party software products have been developed. In Section 6, we will then look at how three different third party auditing software products address the security gap between native database capabilities and auditing requirements.

5.1 Native Database Systems: Logging

When + how often to log
All three databases monitor every command to detect qualified events. If a particular event is met with the predefined condition by users, it logs necessary information to the vault. This is the fundamental principle of the database audit so similar for all database products. In addition, Oracle does log by default during database startup or shutdown and connections to an instance with administrator privileges. It also logs all activities when users log on as system administrator privilege like SYSDBA or SYSOPER.

What to log
Sybase logs event type, event mode (success or fail), database name, event time, transaction id, table name, object id, server user id, database userid , and extrainfo (other information) of DDL or DML events such as "create", "alter", "delete", "drop", "select" for "table", "view", "procedure", respectively, or system events such as error or disk init, disk mirror, etc. The "extrainfo" column contains additional information about the event that is specific to the event number. First, it contains the executed SQL command so users can look up whole commands. If a user had executed multiple commands, each command is preserved by separating them by a colon. Second, it provides simple transition table that has "previous value" and "current value". This allows for rollback when the audit is found to be malicious and needs to be revoked.

Oracle 10g database stores user name, instance number, process identifier, session identifier, terminal identifier, name of the schema object accessed, operation performed or attempted, completion code of the operation, date and time stamp, and system

privileges used. The instance number helps to identify which instance the user was connected to when making changes in Oracle Real Application Clusters (RAC) environment and the terminal ID tells which terminal the user was connected.

SQLServer logs all the same attributes as Sybase plus the origin of the request (machine name). If the SQL text includes sensitive information, for example, a password, SQLServer replaces it with '*'. This can be seen as a second type of sanitization according to Bishop's classification where the logger removes information before logging so nobody can see it. Contents of the audit logs can be browsed through a GUI program called SQLServer Profiler. Of particular interest are the EventClass column, which identifies the recorded auditable event; the TextData column, which contains the command (typically a stored procedure or SQL statement) that was executed in the database; the NTUserName and NTDomainName columns, which identify the user who ran the command; the ObjectName and DatabaseName columns, which identify the table, view, or stored procedure that was manipulated or invoked; and the StartTime column, which records when the command was issued. Of secondary interest might be the RoleName, TargetUserName, TargetLoginName, and DBUserName columns [SQLSVR2000].

SQLServer complies with the C2 requirement of U.S government. C2 auditing records include extensive information not only related to server shutdown and startup, but also success or failure of permission when accessing individual objects in a database and executing DDL and DML commands. When the system is configured for a C2 audit, the conditions that are monitored cannot be altered. This is biggest disadvantage of C2 auditing. If fine-grained auditing is desired to monitor the access of a certain table, for example, the Server-Side Trace functionality must be used.

Where to log

The storage location of the logs may be one of the most important factors in auditing system. Sybase uses a system database called 'sybsecurity'. This database can contain up to 8 audit tables, but only one table will be used at a specified time. Since Sybase uses tables for auditing, an analyst is able to query by attribute. This provides fine-grained search capability as much as 3rd party auditing software products have. In addition, column-based lookup enables 3rd party software developer to easily implement a more powerful graphical user interface because they do not need to parse logs to extract a particular attribute.

Unlike Sybase, Oracle permits the administrator to choose the location of audit trail in either an operating system file or a database. Oracle uses an audit trail database named SYS.AUD$ and DATA DICTIONARY table. The SYS.AUD$ database has several tables such as DBA_AUDIT_SESSION, DBA_AUDIT_OBJECT, and DBA_AUDIT_STATEMENT. To make data retrieval easy, several built-in views are provided to users. DBA_AUDIT_TRAIL is a view based on the AUD$ table that presents the information in a readable, text-based form. Users are also able to log to the operating system's file. That is done by use of syslog function. This option is more secure since the Database Administrator (DBA) does not have privilege to delete the OS's syslog. This means there is still evidence of malicious users' activities even if the DBA modified some tables or records on purpose and then deleted audit information from the native database. Compared to Sybase, Oracle provides additional session information from session table which does not exist in Sybase.

Auditing in Microsoft's SQLServer can be done by using the Windows Security EventLog, the SQL Profiler, and DDL triggers [Ruebush 2005]. SQLServer is distinct in that it is able to log to the Microsoft Windows file system. Secondly, the default error log for SQLServer is stored to a file in the folder "C:\Program Files\Microsoft SQL Server\MSSQL\LOG". Lastly, C2 auditing log can be saved in a file or in the database. This is biggest difference compared to other databases. When a trace is configured for a set of characteristics to be monitored, a client program named SQL Profiler allows the user to choose between "Save to file" and "Save to tables". If the user chooses to log to a file, audit logs are stored in the "Program Files\Microsoft SQL Server\Data\" folder as AuditTrace_yyyymmddhhmmss.trc, where the second part of the name indicates date and time when the log file was created [SQLSVR2000]. Users can view the contents of logs using SQL Server Profiler. In case of fine-grained auditing, which can be thought of as customized auditing for a particular table, SQLServer logs it into "C:\sqlauditfile.trc". After creating a trace, the user can configure the event type, the columns they want to audit, and bits related to other fields.

How to log

In this section, we explore how users create audit rules which generate the logs. Sybase and Oracle use grammar-based semantics to create rules whereas SQLServer uses a step-by-step wizard like a GUI. Sybase uses the sp_audit command for configuring the condition for the events that are desired to be detected and logged. The grammar of sp_audit is as follows:

"*sp_audit option, login_name, object_name [,setting]*", where *option* specifies the type of event to monitor, such as "all", "dbaccess", "select", "delete", "revoke", "security", etc. The "'security" option includes built-in conditions related to (1) system shutdown and startup; (2) activating or deactivating user roles; and (3) certificate or proxy setting. The *[,setting]* field can have the value of "on", "off", "pass", or "fail". The value "off" is used to deactivate that audit rule. If "fail" is specified, the database system will only log when the condition check failed to be true.

Oracle's audit rule creation is done by the following expression:
"audit {statement_option|privilege_option} [by user][by {session|access}] [whenever {successful|unsuccessful}]".
Additionally, users can write a trigger to save customized information. With the help of using trigger, Oracle provides the equivalent functionality of Sybase's pseudo transition table.

One very important auditing feature of Oracle 10g is Fine-Grained Auditing (FGA). Normal auditing triggers can detect who modified which table or which column. In other words, users can track any data modification on tables if they write a trigger procedure. Unfortunately, this does not hold for "SELECT" statement since "SELECT" does not change any data, no trigger can be made. If the security administrator wants to monitor the person who access a particular table, more specifically a column which contains sensitive data, he can use Fine-Grained Auditing feature (FGA). Whenever specified columns are accessed, Oracle logs that event into FGA audit trail. Then an analyst can check the table whether a security breach has occurred or not. Let's see one example how a FGA policy can be written.

```
begin
  dbms_fga.add_policy (
    object_schema=>'BANK',
    object_name=>'ACCOUNTS',
    policy_name=>'ACCOUNTS_ACCESS',
    audit_column => 'BALANCE',
    audit_condition => 'BALANCE >= 11000'
 );
end;
```

In this example, there is a database 'BANK' which contains a table named 'ACCOUNT'. That table has a column 'BALANCE'. If administrator wants to monitor when a user tries to access records whose balance is more than $11000, Oracle logs this event to the FGA audit trail. Since FGA is content-based, it gives users more capability to audit [FGAEX].

The C2 auditing feature of SQLServer does not provide much freedom for customization. Unlike Sybase and Oracle, SQLServer does not allow users to write a SQL-like procedure. Its functionality is rather limited. Users need to select the event type and the object to monitor using Graphical User Interface when he/she makes a new trace by SQL Profiler. Thus, if an event meets the user selected criteria, that event is stored in the audit file. After a trace is activated, it stores the SQL command being executed if the condition is satisfied.

5.2 Sanitization

According to the Oracle security guide, encryption does not solve the access control problem because it requires key sharing between multiple users if more than one person wants to access a particular table [ORASEC2005]. Although it might cause overhead to access information that requires decryption, encryption can still be used to prevent unprivileged users from looking up sensitive data. Sanitization itself does not bring any additional access control advantage either, but it is meaningful for at least obfuscating plain text.

In this sense, encryption is used as a sanitization method, or method of hiding information. Because Oracle has encrypt and decrypt functions that can encrypt/decrypt data in tables, and the audit log is stored as a table, it is our belief that it is possible to encrypt the audit log tables. Assuming this is possible, and finding no information to the contrary, this situation poses a third way to sanitize the audit logs than the two methods presented by Bishop. This method differs from first approach in the sense that the data is not strictly removed from the logs before viewing, but instead, is encrypted. It still allows for access if the decrypt key is known, but does provide for significant protection of the data. It differs from the second approach in the sense that although data is logged in both situations, the Bishop sanitization model uses pseudo look up tables to substitute attributes, whereas in Oracle, the logged data is encrypted.

Sybase Encryption Option does not pose restrictions on the table names, allowing any table to be encrypted. The sanitization functionality can be obtained by designating sybsecurity audit tables. Administrators cannot see the contents of logs unless he/she is in the qualified decrypt group. In addition, this approach can be called a pseudonymizing sanitizer because it deletes information when it shows information to the users, but the originator of the logs can reconstruct the deleted information [Bishop 2003, p698]. Sybase Encryption supports column-based encryption and role-based encryption, saving the decryption key in internal tables. One concern arises under this architecture. If the DBA

is not removed from the allowed user lists which describe accessible roles to the internal key tables, all efforts to protect privacy will be negated. In addition, the Sybase Encryption Option provides advanced protection. In the normal pseudonymizing process, the system looks up a mapping table to reconstruct the original logs. If the original plain text is very simple such like "Yes" and "No", a smart attacker accessing the log can regenerate it without knowledge of decryption key because the range of encrypted value is very limited. To prevent this attack, Sybase ASE Encryption option uses random encryption keys. This will improve the resistance to key analysis attacks.

Oracle also supports table encryption. To use encryption, Oracle users have to write a PL/SQL program to encrypt and decrypt tables. With DBMS_CRYPTO package, users are able to encrypt only selected columns. One difference from Sybase Encryption is key management. In Oracle, key management must be done by the application, so an application user or software that encrypts columns or tables must store the decryption key somewhere that is not in the table.

SQLServer has similar features as the others. One distinctive feature of SQLServer is the use of certificates to distribute keys. Encryption and decryption keys are managed internally. Therefore if users take advantage of the built-in crypto function, they can obfuscate critical information. However, users should remember that this approach is only meaningful for table-based logging. If the default audit log is stored in a file, encryption does not provide privacy. Users have to come up with their own solution not to reveal sensitive data.

5.3 Analyzing and Notification

The ability to analyze the data in the audit logs contributes to the effectiveness of the auditing system to meet the four goals of auditing stated in the previous section. Since Oracle and Sybase use tables as a location of logs, users can easily access and investigate them using standard SQL command. More customized auditing procedures can also be written and saved for later reference. The output of log analysis can be stored or displayed by browsing tools, and so the browsing techniques can impact the effectiveness of the analysis.

In order to analyze the log data, an auditing tool must be able to understand the format of audit log and, at a minimum, be able to parse the log generated by database unambiguously. Atallah proposed to verify the soundness of log format by using a verification tool to check the completeness of the logging grammar [Atallah2000]. This approach might be also useful in database auditing area. Since third party audit software products have to deal with diverse log formats, they can benefit from complete and grammatically correct log entry. Ideally, every database system vendor would use a standard format of log with commonly agreed attributes. However, this assumption is too optimistic. Nonetheless, if log entries were grammatically sound, that would facilitate the integration of logs more easily by log consumers.

Atallah's approach cannot be effectively applied to Sybase and Oracle since they use tables to store the audit log, thus parsing has already been done. However, since SQLServer uses a file for audit log and saves the whole SQL statement as one attribute even when it uses a table for auditing, a query by columns requires searching throughout whole log file. If Atallah's grammar was used to fragment SQL command to each category, the analyzing job will be greatly improved.

The SQL Profiler is a GUI that can be used to not only configure events, but also to monitor and analyze them too. The platform provides a variety of analysis and reporting features including online analytical processing (OLAP) capabilities. Notifications can be sent using SQLServer Notification Service after DDL and SQL Trace events or on a predetermined schedule specified by the subscriber. Notifications can be sent to a wide range of devices, including to a cell phone, PDA, or e-mail account [MSDN Notification]. Similarly, Oracle 10g includes analysis tools like OLAP, and data-mining, and also supports notification to cell phone, pager or email. Focused at the data level, Sybase Data Auditing monitors and audits DML and DDL to provide an understanding of which data has been accessed or changed, when, and by whom. Its included reports allow auditors and others to view, report, and analyze database activity [Sybase Auditing 2005].

The following table contains a summary of the auditing features for the three database products we explored.

function\database	SQLServer	Oracle	Sybase
Support division of duties	Y	Y	Y
Sanitization	Can be implemented using Encryption	Can be implemented using Encryption	Can be implemented using Encryption
Encryption	Y	Y	Y
What to log	all objects, tables, views, procedure, trigger	the same as SQLServer plus successful/not successful	all object, all table, all views, procedure, trigger
When to log	database startup, shutdown; when EC (Event-condition) is satisfied	EC is satisfied	EC is satisfied
Where to log	operating system	database or operating system or XML	database only
Fine-grained auditing	Y	Y	N
Result storage type (table or file)	File or Table*	Table	Table
Audit trail database name	NA	SYS.AUD$ table	sybsecurity table
Column based search possible	Y	Y	Y
Can limit # of returned rows for sensitive data	N	Y	N
Mechanism	composing rules by GUI tool	Selected user database action using audit features, define triggers	composing rules by grammar based audit tools
Frequency	Realtime	Realtime	Realtime
Support command capture	Y	Y	Y
Support command regeneration	N**	Y	Y
Support Rollback (for suspicious transaction)	Y	Y	Y

* SQLServer can import file log into table.
** SQLServer does not support built-in recovery tools but you can use third party product to recover records from log

6.0 The need for third party auditing products

Although major database system venders support various functions to users such as generating new audit rules, managing the log, and recovery of the original database if any corruption has been found due to malicious users' activities, simply turning audit function on does not guarantee the completeness of database audit because more and more threats are coming from within. The need to address increasing security breaches from within requires more effort to secure the data than is provided by the native database's auditing tools. For example, it is necessary to restrict the privilege to modify sensitive data even if the user is a DBA. In addition, privileged user must not be allowed to modify audit logs in any case because the audit log is the key component for the investigation and system recovery.

Audit in native database is mostly done by triggers. Using triggers has a few advantages. First, it is easy to implement. Second, it is flexible in the sense that one who writes a trigger can freely modify it. However, even though a trigger is easy to write, it is also hard to maintain. Triggers require modification whenever any change is made to the database schema. Triggers are easily disabled and enabled later. So database systems that rely primarily on triggers for auditing are not very effective in preventing or detecting an inside threat from privileged user who can freely manipulate the trigger. In addition, internally developed tools using triggers do not capture all activity that it is desirable to monitor. For example, native database capabilities may not be able to detect attacks which are done by embedding SQL command in applications. Furthermore, triggers might interfere with runtime performance so may lead to performance degradation. If a large volume of audit logs have to be stored in a table in real time, it causes undesirable overhead. In addition, all internal activity can not be absolutely independent from DBA.

To overcome these limitations, many users are using third party audit software to address their auditing needs. In this paper, we will explore the capabilities of three third party audit software products: Lumigent's Audit DB, AppSecurity's AppRadar, and Guardium's SQLGuard. Since we have already addressed the general mechanism of how database auditing works,

we will only focus on the difference and improvement capabilities compared to native databases.

6.1 Working mode of 3rd party products

Since 3rd party auditing software products deal with multiple databases instances, its architecture has to have a server-client structure. The term 'Agent' is a client program that functions as a sensor to monitor SQL command in the network. The location of the agent, or monitoring method, can be characterized by one of five methods. This method has an impact on how the agent reads SQL commands and its ability to block those commands.

Monitoring methods deal with approaches to collecting data or logging [Conry-Murray 2005]. The first method consists of passive server or device that taps into the network and monitors traffic flowing into and out of the database system. AppRadar is an example of software that uses the passive monitoring method. Passive monitoring will not slow down the network and it will cause no overhead in database system. Although passive monitoring does not cause any performance delays, it is not able to monitor encrypted traffic.

The second method puts devices directly inline with the network so that all packets going in and out are intercepted and passed by the device. Guardium's SQL Guard is a system that is able to work as both passive and inline modes. Inline monitoring methods can block attacks or misuse. It is also able to monitor encrypted packets. Decryption of packets and verifying of each packet may cause network latency.

Passive and inline methods are similar in that they can both monitor network traffic. For this reason, they share common characteristics. Both methods are not able to monitor activities of users who are directly logged into the database. This means that they are not able to protect that database against insider threats. Even with the downfall, both methods are frequently used because they do not impact the database system performance. Also, network traffic does not consist solely of database network packets, so it is important that they have ability to detect attack signatures specific to database platforms. AppRadar, Guardium's SQL Guard and most other database auditing systems provide this ability.

The fourth method is agent based monitoring where the extraction of logs is done directly from the database. Example of agent based monitoring is Lumigent's Audit DB. Audit DB distributes individual agents that extracts log from each of the database(s) and reports to the central auditing system where analysis and reporting are done. Agent based monitoring is able to validate all database activity not affected by encryption. Although this method provides complete logging, it may affect the performance of database systems. Also with multiple agents, complexity of system increases.

The final method of monitoring is a system integrated method. System integrated monitoring methods act as part of the database system. This method consists of all monitoring methods provided by Oracle, SQLServer, Sybase, and all other database system that provide auditing methods. System integrated monitoring give complete logging. However, like any other programs that run in the database system, it can highly degrade the database performance.

Lumigent's Audit DB simply collects logs from native database products so they purely rely on the native database's audit log. Agents are installed to the system where native database resides in. Agents monitor, alert and collect all database transactions from the SQL Server transaction log, Oracle redo log or the native database audit feature (Trace or Oracle Audit) [LUMIFAQ]. This approach is the simplest solution for monitoring multiple heterogeneous databases. However, this is not fundamentally different from native database's approach. We think Audit DB is still weak to the attack from inside because if audit log in native database is modified on purpose, the result directly affects the analysis of Audit DB. Their contribution is to transform different log format to their own format.

Application Security's AppRadar gets SQL commands from the network. Agents for Sybase, Oracle and IBM DB2 databases tap into network and make a copy of every packet through network switching equipment using a monitor or Switched Port Analyzer (SPAN) port. This does not cause transaction latency on the database or network traffic because it does not block any packet. For SQLServer, an agent is deployed in the system where SQLServer is running.

Guardium's SQL Guard can operate as both inline or passive server and can migrate from monitoring to prevention modes. If SQL Guard is deployed as inline, it sits between user terminal and database and checks every incoming packet to see if it is related to an attack or unprivileged activity. If it thinks a packet is a dangerous one, it will block the packet so the user issued command cannot reach to the database, thus operating in prevention mode in this case. Blocking is powerful function and may be useful in the sense that it has prevention capability. It also has the potential to have a false positive response. In worst case, a legal user's safe SQL command can be blocked. Some

products provide pattern learning functionality to lower the false-positive. AppRadar and Audit DB do not have pattern-learning, or profiling, feature. However, Guardium's solution automatically suggests customized policies via a "learning mode" that establishes a baseline of normal activity and looks for anomalous behaviors. (Policies are generated or derived from baseline of normal activity.) For better profiling, machine learning technique might be utilized, but that is beyond the scope of this paper.

6.2 Logger

All events from agents will be transferred to server's database. Users can query central manager's log for forensic purposes. In this section, we see what kinds of attributes each auditing system provides to the users.

Audit DB records Session ID, Date & Time, the user (SQL username and NT username), Client Host Name, Login Name, Application Name, Server, Database, Tables & Views, OS user and the actual text of the SELECT command. AppRadar stores Agent IP, Title (usually the type of event, e.g., SELECT, UPDATE, DELETE), Date & Time, Source of Attack (corresponding Client Host Name in Audit DB), and SQL Text executed to the SQLServer. If an administrator of AppRadar finds normal events were reported as attack, he can separate that event and can make an Exception. That kind of event will not be regarded as an attack from then on. SQL Guard has the ability to monitor information about the user and application trying to access the database including account names, IP addresses, and session information. It also can log both DML and DDL commands. It can also log access to sensitive data fields in the database [Lindstrom 2005].

6.3 Sanitization

One of the primary purposes of third party auditing is to monitor suspicious activities in real time. To be most effective, the subject who does the monitoring must be different from normal database administrators. This is necessary because segregation of duty is an essential component of a security plan. For that reason, it is important that audit logs provide sufficient information for the auditor to detect malicious access to the database. This must be done while still protecting sensitive data as much as possible. Sanitization techniques in third party software products must be implemented so as to balance the needs of the auditor and the protection of sensitive information.

6.4 Policy generation & deployment

With third party auditing software, it would be very convenient if users needed to configure only one common rule for all databases. Every database supports predefined policies that users can choose. In this sense, users can have the same policy over all managed databases. However, none of the three 3rd party auditing programs support creating customized rules. It may because of the schema of each database instance are likely to be different, and fine-grained auditing requires specific table name or column names to monitor.

AppRadar cannot manage multiple heterogeneous databases with one AppRadar Server. If a company has Oracle 10g, Sybase ASE, and SQL Servers, one server for each database needs to be installed. AppRadar has policy manager and filter manager interfaces. Policy manager enables users to customize audit policy (select among built-in policy) whereas filter manager enables users to create a new audit rule. Unlike AppRadar, SQL Guard can manage heterogeneous databases with one centralized server. SQL Guard supports automated policy generation for database access. Unlike other systems that require the user to perform a detailed analysis of the environment and manually define policies, Guardium's solution automatically suggests customized policies via a "learning mode" that establishes a baseline of normal activity and looks for anomalous behaviors. It is important to note that SQLGuard V5.0 now supports fine-grained auditing. In addition to managing multiple databases like SQL Guard, Audit DB also allow users to define acceptable access and use of data. Audit DB creates baselines of “normal” behavior from which to measure drift from policy. If anomalous activity is found from the log, agent alerts to the central server [AUDDS].

6.5 Analyzer and Notifier

Analyzing process of most vendor database systems such as Oracle, MS SQLServer, and Sybase are done by SQL querying. They do not provide predefined analysis tools or programs. Third party auditing software provides predefine automatic analysis tools. In AppRadar, FISMA, HIPAA, PCI, and SOX regulations have been analyzed as selectable policies and each of these policies can be modified to an organization's specification. In addition, new rules/signatures can be created, although AppRadar does not have any pattern learning feature. Audit DB does not provide analysis based specifically on the government regulations discussed, but does provide analysis of compliance using the atomic regulations. It also does not have learning ability but outputs compatible forms that can be used as an input to other

data mining software. SQL Guard has policy learning mechanisms of defining normal activity by using real-time linguistic analysis which is not a complete learning technique, but does use contextual information. All three products offer real-time alerts for automatic notification whenever policies are violated.

6.6 Handling full log

When the database system or the operating system that runs database products do not have enough space for saving audit logs, they have to decide whether to process the newly created logs. They can simply ignore if the log size exceeds the limit or they may choose to delete the oldest entry in the log archival system and then insert most recent log entry.

In Oracle, if logs stored in database audit trails become full, it is no longer able to accept new records and audit action cannot be completed. This causes error message to be reported. If logs stored in operating system audit trail becomes full only certain actions that are always audited are allowed to execute and other actions are discarded [Oracle2005].

If log storage of SQL Server becomes full, the database system will not restart until there is free storage. If the storage is not available, it is possible to modify the arguments of Service Control Manager (SCM) service setting. An alternative method is to start a minimal SQL Server configuration from the command line and prevent clients from connecting to the database and perform transactions while auditing is disabled [SQLSVR2000]. Also, maximum size of storage and action when the maximum size is reached can be specified. Three options are available for the action: overwrite events as needed, overwrite events older than specified number of days, and do not overwrite events [MSSQL2000].

Feature\Audit software	**AppRadar**	**Audit DB**	**SQL Guard**
Working mode	passive	pulling information from database	inline & passive
Profiling	N	N	Y
Monitoring objects	DDL,DML	DDL,DML	DDL,DML
Detect OS resource use	Y	Y	N
Supporting Databases	Oracle, SQLServer, Sybase,DB2	Oracle, SQLServer, Sybase	Oracle, SQLServer, Sybase,DB2
Provide User defined rule	Y	Y	Y
Unified policy* deploy	Y	Y	Y
Fine-grained auditing	N	N	Y
Log Browsing	Can export reports to TEXT, HTML, DHTML, RTF only	TEXT, PDF	Crystal reports (almost every format)
Regulation			
FISMA compliance	Y	N	N
HIPAA compliance	Y	Y	Y
PCI compliance	Y	Y	Y
SOX compliance	Y	Y	Y

* Here policy means a policy for database auditing not for access control policy for database.

7.0 Conclusion

We explored the auditing system as described by Bishop as it applies to databases in general, and surveyed how it specifically has been deployed to three major commercial database systems. We explored the impact of recent governmental regulations on organization's auditing requirements. We determined that as the auditing and security needs increase, commercial databases are increasingly finding it a challenge to meet these needs with just native capabilities. Finally, we surveyed three third party software products that have been developed to help bridge the gap in auditing needs of database systems.

References

[Atallah2000] Chapman Flack, Mikhail J. Atallah, Better Logging through formality , Springer-Verlag Lecture notes in Computer science; Vol 1907, 2000.

[AUDDS] http://www.lumigent.com/files/AuditDBTechnicalDataSheet.pdf

[Axelsson 2005] Axelsson, S. Intrusion Detection Systems: A Survey and Taxonomy, March 2005.

[Ben-Natan 2005] Ben-natan, Ron. Beyond instrusion detection: The next frontier in safeguarding corporate assets. Guardium White paper, 2005.

[Ben-Natan 2006] Ben-Natan, Ron. Data Privacy: Protecting the Core of your Business. Guardium White paper, 2006.

[Bishop 2003] Bishop, Matt. Computer Security: Art and Science. Boston, MA: Addison-Wesley, 2003.

[BWF] Bishop, Wee and Frank. "Goal-Oriented Auditing and Logging" from http://seclab.cs.ucdavis.edu/projects/awb/index.html, Accessed 11/26/06.

[CERT 2005] Insider Threat Study: Computer System Sabotage in Critical Infrastructure Sectors, U.S. Secret Service, CERT, May 2005.

[Clifton 2004] Clifton, Chris. Course notes from CS526; Information Security. Accessed 11/26/06.

[Conry-Murray 2005] Conry-Murray A., The Threat From Within, URL: http://www.itarchitectmag.com/shared/article/showAriticle.jhtml?articleid=166400792 08/01/2005.

[Dumlar 2005] Microsoft MSSQL Server 2006 Product Guide. September 2005.

[FGAEX 2003] Arup Nanda, http://www.oracle.com/technology/oramag/webcolumns/2003/techarticles/nanda_fga.html

[HIPAA1] http://en.wikipedia.org/wiki/Health_Insurance_Portability_and_Accountability_Act

[Lindstrom 2005] Lindstrom, Pete. Data-Centric Security: A Spire Research Report. April 2005. www.spiresecurity.com.

[LUM] http://www.lumigent.com/solutions/subject_other.html.

[LUM_triggers] http://www.lumigent.com/solutions/triggers.html

[MSDN EventLog] http://msdn2.microsoft.com/en-US/library/dhacse28.aspx . Accessed 12/4/06.

[MSDN Notification] http://msdn2.microsoft.com/en-us/library/ms166495.aspx . Accessed 12/5/06.

[MSSQL2000] - SQL Server 2000 Auditing , http://www.microsoft.com/technet/security/prodtech/sqlserver/sql2kaud.mspx By John Howie, August 2002

[ORASEC2005] - Oracle @Database security guide 10g release 2(10.2) B14266-01, June 2005

[Orman 1997] Orman, L. V. 1997. Database auditing. In Proceedings of the Eighteenth international Conference on information Systems (Atlanta, Georgia, United States, December 14 - 17, 1997). International Conference on Information Systems. Association for Information Systems, Atlanta, GA, 297-314.

[Ram 2006] Ramakrishna, M. V. http://www.csse.monash.edu.au/courseware/cse5200/ddb-l-security.pdf . Accessed 11/27/06.

[Ruebush 2005] Ruebush, Mitch. Microsoft SQL Server 2005: SQL Server 2005 and Oracle 10g Security Comparison.

[SalSch 1975] Saltzer, Jerry H. and Schroeder, Mike D. The protection of information in computer systems, Proceedings of the IEEE, vol. 63 (no. 9), pp. 1278-1308, Sept 1975.

[Sec Inn 2006] Security Innovations, Inc. "Regulatory Compliance Demystified: An Introduction to Compliance for Developers", March 2006, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsecure/html/regcompliance_demystified.asp, accessed 11/20/06.

[SO101] http://www.sarbanes-oxley-101.com/sarbanes-oxley-faq.htm. Accessed 11/1/06Conry-Murray A., The Threat From Within, URL: http://www.itarchitectmag.com/shared/article/showAriticle.jhtml?articleid=166400792 08/01/2005.

[SQLSVR2000] - SQL Server 2000 Security - Part 10 - Auditing By Marcin Policht, http://www.databasejournal.com/features/mssql/article.php/3399241, August 27, 2004.

[Sybase Auditing 2005] http://www.sybase.com/content/1034871/Data_Auditing_Overview_ds.pdf . Accessed 12/5/06.

Appendix A.

Table 1. Summary of technologies used to address governmental regulations [Sec Inn 2006]

Need	SOX		HIPAA		PCI	
	Requirement	Technology/ Technique	Requirement	Technology/ Technique	Requirement	Technology/ Technique
Confidentiality	Confidential information cannot be exposed to unauthorized entities.	Use of encryption techniques and algorithms to ensure data is only divulged to authorized individuals.	All ePHI must be kept confidential to prevent unauthorized access.	Use of strong encryption when storing and transmitting confidential data in database	Protect consumer credit card information.	Encryption of data. Only authorized systems or agents have access to sensitive account information.
Integrity	Software needs to support evidence that data has not been modified.	Cryptographic hashes and robust integrity checks.	Records should not be modifiable by unauthorized people or entities.	Use integrity checking mechanisms that apply principles of least privilege and minimize risk of privilege escalation.		
Availability	Availability of financial data to authorized individuals.	General code reliability, resistance to denial of service attacks, reliable data storage, recovery in case of system failure.	Record owners are guaranteed the right to access their own records.	Designed to properly handle errors and withstand denial of service attacks. Maintain event logs to be able to reconstruct.		
Access Controls	Support role-based access and revocation of accounts.	Integrating into larger identity management framework like LDAP. Review access rights.				
Auditing and Logging	Auditing and logging of events in systems that process sensitive data. Logs must not reveal information that system is trying to protect.	Logging of system events such as shutdowns, restarts, or unusual events. Avoidance of logging of sensitive data.	Any action that might need to be traced must be documented.	Log information to be able to construct a clear audit trail of how a user or entity attempts to access and utilize resources. Logs should be backed up.	Comply with periodic audits to demonstrate standards are adequately being met.	Logging all pertinent account transactions and accesses.
Change Management	Must notify SEC of any material changes to process that governs flow of financial data.	Log system changes in a way that they are resistant to tampering and accessible to only privileged users (separate protected system).				
Authentication			To secure ePHI, it is necessary to know the entity or person that is working with the data is legitimate and authorized to access said information.	Authentication systems should clearly map roles to permissions. Strong passwords should be enforced.	Protect consumer credit card information.	Only authorized systems or agents have access to sensitive account information.