Journal of Information Security, 2012, 3, 113-121
http://dx.doi.org/10.4236/jis.2012.32014 Published Online April 2012 (http://www.SciRP.org/journal/jis)

Digital Evidence for Database Tamper Detection

Shweta Tripathi[1], Bandu Baburao Meshram[2]
[1]Department of Computer Engineering, Fr. Agnel Institute of Technology, Navi Mumbai, India
[2]Head Department of Computer Technology, Veermata Jijabai Technological Institute, Mumbai, India
Email: shwetripathi@rediffmail.com, bbmeshram@vjti.org.in

Received December 28, 2011; revised February 5, 2012; accepted March 4, 2012

ABSTRACT

Most secure database is the one you know the most. Tamper detection compares the past and present status of the system and produces digital evidence for forensic analysis. Our focus is on different methods or identification of different locations in an oracle database for collecting the digital evidence for database tamper detection. Starting with the basics of oracle architecture, continuing with the basic steps of forensic analysis the paper elaborates the extraction of suspicious locations in oracle. As a forensic examiner, collecting digital evidence in a database is a key factor. Planned and a modelled way of examination will lead to a valid detection. Based on the literature survey conducted on different aspects of collecting digital evidence for database tamper detection, the paper proposes a block diagram which may guide a database forensic examiner to obtain the evidences.

Keywords: Tamper Detection; Log Files; Forensics; Oracle Database

1. Introduction

Database security is not a new buzz. But identifying the reasons of security violation in a database is a recent point of discussion. To understand how, when and what was tampered in a database the thorough knowledge of the architecture of database is required. To bind the vast information about architectures of different databases the architecture of oracle 10g is considered.

Database Forensics is a branch of digital forensic science relating to the forensic study of databases and their related metadata. For the forensic examination of a database, it has to be related to the timestamps that apply to the update time of a row in a relational table being inspected and tested for validity in order to verify the actions of a database user. Alternatively, a forensic examination may focus on identifying transactions within a database system or application that indicate evidence of wrong doing, such as fraud. Hence forth identifying who, when and how modified or tampered the data.

There are many approaches towards forensics of database defined by different researchers. A structured model to collect the evidence is a need of database forensics which has been proposed in this paper as the initial step of examining the database after the tamper.

The organisation of rest of the paper is as follows: Section 2 gives the literature survey. Based on basic concepts, section 3 is devoted for process to collect evidence. Section 4 gives the conclusion.

2. Literature Survey

The literature survey explores from the basics of architecture of oracle 10g to the vulnerabilities in oracle, digital forensics analysis and database tamper detection.

2.1. Basics of Oracle Architecture

In this section the basics of oracle 10g is explained highlighting few **Figure 1** [1]. Oracle physical storage and memory structure is explained in brief. Since these locations may guide us to locate and analyse the tamper in the database.

2.1.1. Oracle Physical Storage Structures

The Oracle database uses a number of physical storage structures on disk to hold and manage the data from user transactions. Some of these storage structures, such as the datafiles, redo log files, and archived redo log files, hold actual user data; other structures, such as control files, maintain the state of the database objects, and text-based alert and trace files contain logging information for both routine events and error conditions in the database. **Figure 1** [1] shows the relationship between these physical structures and the logical storage structures.

2.1.2. Datafiles

One Oracle datafile corresponds to one physical operating system file on disk [1].

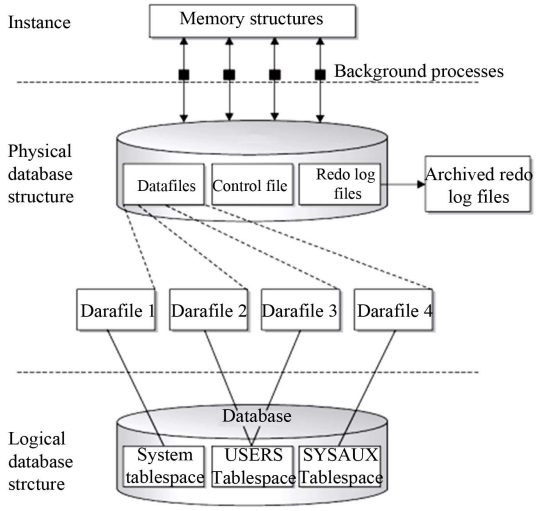

Figure 1. Oracle physical storage structures.

2.1.3. Redo Log Files

Whenever data is added, removed, or changed in a table, index, or other Oracle object, an entry is written to the current redo log file [1].

2.1.4. Control Files

Every Oracle database has at least one control file that maintains the metadata of the database (in other words, data about the physical structure of the database itself) [1].

2.1.5. Archived Log Files

An Oracle database can operate in one of two modes: archivelog or noarchivelog mode. When the database is in noarchivelog mode, the circular reuse of the redo log files is no longer available in case of a failure to a disk drive. Archivelog mode sends a filled redo log file to one or more specified destinations and can be available to reconstruct the database at any given point in time in the event that a database media failure occurs [1].

2.1.6. Alert and Trace Log Files

When things go wrong, Oracle can and often does write messages to the alert log and, in the case of background processes or user sessions, trace log files [1].

2.1.7. Backup Files

Backup files can originate from a number of sources, such as operating system copy commands or Oracle Recovery Manager (RMAN) [1].

2.2. Vulnerabilities in Oracle

To collect the evidences for data tamper detection one should have a brief knowledge of the suspicious location in the database. This part of the paper summarizes the vulnerabilities in oracle 10g with the [2].

The author mentions there are three areas an attacker can exploit to break in to a given system. Firstly, they can exploit the trust already given to them in the case of an inside attacker or in a social engineering attack; secondly, an attacker can exploit a weakness in the configuration of the server and third and last, an attacker can exploit a vulnerability in the software. Social engineering attacks are beyond the scope of this paper. We have highlighted few weaknesses in server configurations and software vulnerabilities. Some of the common configuration weaknesses that can be exploited to break in are listed below.

2.2.1. Exploiting Configuration Vulnerabilities

Exploiting configuration vulnerabilities is an easy task of an attacker due to following vulnerable configuration [2].

Default Usernames and Passwords Prior to 10g, Oracle was renowned for the number of default user accounts that have default passwords after an install. Some common accounts with their password include [2].

Username	**Password**
SYS	CHANGE_ON_INSTALL
SYSTEM	MANAGER
DBSNMP	DBSNMP
MDSYS	MDSYS
CTXSYS	CTXSYS
WKSYS	WKSYS
SCOTT	TIGER

***Files*:** After installing 10g, both release 1 and 2, there are a number of files that contain the password selected by the DBA during installation. Whilst the passwords are obfuscated it is trivial to recover the clear text password from these files. If an attacker can gain access to these files then they have the keys to the kingdom. These files include:

CONFIGTOOLS.XML
CONFIGTOOLSINSTALLDATE.LOG
IAS.PROPERTIES
TARGETS.XML [2]

Reflections on default passwords Many times we often hear a DBA protest that they block direct connections to the database with a firewall rule setting—this generally means blocking access to port 1521—so they don't need to worry about default user IDs and passwords. However, they often forget to block ports 2100 and (less so) 8080 which means that, if the Oracle server is running the XML Database (XDB), then attacker can log in over these ports. An FTP service in XDB listens on TCP port 2100 and a web server on port 8080 [2].

Unnecessary Services Here is a simple chain of logic: the more services that are running, the greater the size of the attack surface; the greater the attack surface; the greater the likelihood of a flaw existing, the greater the

likelihood of a successful break in. If there is no strict business requirement for something to be running, if it doesn't serve an active role in any business processes then it should be turned off [2].

The TNS Listener Prior to Oracle 10g the TNS Listener was installed without a password set for it. This means that, unless a password was set at a later time, anybody with access to port 1521, or whatever the port the server is listening on, could remotely administer the Listener. One of ways in which an attacker could exploit this was to set the listener's log file to a set location such as ~/.rhosts on a *nix system or a batch file in the Startup folder of the administrator on a Windows system. Once the location is set then the attacker can write content to the file such as "+ +" in the case of .rhosts or a command to execute in the case of a batch file in the Startup folder [2].

2.2.2. Exploiting Software Vulnerabilities

There are many different classes of software vulnerability and every so often new classes are discovered. Oracle has suffered from most at some point or continues to and these provide an attacker with a way into the server [2].

Buffer Overflow Vulnerabilities

A buffer overflow vulnerability is conceptually easy to understand. The programmer sets aside some memory to hold some data and they make some assumptions about the size of that data. Along comes an attacker however and stuffs too much data into the buffer. The buffer overflows and the data overwrites other "stuff" in memory—some of which may be crucial to the running of the program or the program's flow of execution. With this now under the control of the attacker they can get the program to do their bidding as opposed to what the program was supposed to do [2].

Format String Vulnerabilities

Format string vulnerability arises when a programmer uses one of the functions in the printf family of functions without specifying the format string. This, in effect, allows the attacker to present the format string instead [2].

PL/SQL Injection

PL/SQL injection vulnerabilities are one of the more common types of Oracle security flaw. Oracle stored procedures are known as packages. These packages contain data, procedures and functions. They also have the ability to execute dynamic SQL statements. By default, when a PL/SQL package executes it does so with the privileges of the owner and so, if the package suffers from a security vulnerability, it presents the attacker with the ability to run SQL as the owner of the package [2].

Trigger Abuse

Not only are packages, procedures and functions vulnerable to PL/SQL injection but triggers can be too. A trigger is a piece of PL/SQL code that fires when a specific event occurs—for example a user performs a DML operation such as an INSERT. A number of triggers have been found to contain security weaknesses. These include the SDO_LRS_TRIG_INS, SDO_GEOM_TRIG_INS1 and SDO_DROP_USER_BEFORE triggers owned by MDSYS and the CDC_DROP_CTABLE_BEFORE trigger [2].

Attacks via Oracle Application Server

Attacks from the outside of the network, particularly where the database server is properly protected by a firewall, typically originate from the web server—more often than not in the form of SQL injection attacks in the custom JSP, PHP or ASP application [2].

2.3. Digital Forensic Analysis Methodology

The entire investigation process can be divided into four phases [3].

1) Identification: in this phase it collects the information of the compromised system. System Configuration, software loaded, User profiles etc.

2) Collection Phase: collects the evidence from the compromised system. Evidence is most commonly found in files and Databases that are stored on hard drives and storage devices and media. If file deleted, recovering data from the deleted files and also collects evidence file deleted files.

3) Analysis phase: analyse the collecting data/files and finding out the actual evidence.

4) Report phase: the audience will be able to understand the evidence data which has been acquired from the evidence collection and analysis phases. The report generation phase records the evidence data found out by each analysis component. Additionally, it records the time and provides hash values of the collected evidence for the chain-of-custody.

2.4. Database Tampering

Maintaining data integrity is an important aspect of security in a database. The basics of database tampering can be explained better with the case study.

2.4.1. Example of Data Tampering

Any business cannot afford the risk of an unauthorized user observing or changing the data in their databases. There are several types of concerns that are realized about database security. They are "unauthorized data observation, incorrect data modification, and data unavailability". An example of unauthorized data observation would be a database user accessing information that they are not authorized to view. Incorrect data modification can be intentional or unintentional. Intentional data modification could be a student changing their grade or a data entry clerk accidentally entering the price of a line item incorrectly for an order. Data unavailability exists

when "information crucial for the proper functioning of the organization is not readily available when needed." [4].

A data breach in a bank information system [5] was an unauthorized access to clients banking accounts which has resulted with money problems to a certain number of clients. Somehow, they have a minus on credit card accounts. Bank personnel are unable to identify the culprit. Persecution department send a team of digital forensics. They have a task to collect all evidence about suspicious transactions, examine them and make a report to the court of low. The team needs to follow a precisely defined procedure to provide valid court evidence (**Figure 2**).

Different kinds of log files are available in oracle 10g. To identify who had got the unauthorized access, when he had got the access and how he had tampered the data in the accounts database, a forensic examiner should collect the relative log files and perform the analysis to gain the information about the criminal.

2.4.2. Detection Methods

Different methods to detect tampering can be based on the identification of violation of the integrity of the database. The threats to integrity are [6] authentication and access control, application integration, database extensions, inherited OS vulnerabilities, privileged parties.

A novel relational hash tree [6] can be designed for efficient database processing, and evaluate the performance penalty for integrity guarantees. Such implementation does not require modification of system internals and can therefore easily be applied to any conventional DBMS.

Database provenance chronicles the history of updates and modifications to data [7]. Without additional protecttions, provenance records are vulnerable to accidental corruption, and even malicious forgery, a problem that is most pronounced in the loosely-coupled multi-user environments often found in scientific research. The problem of providing integrity and tamper-detection for database provenance can be solved by a checksum-based approach, is well-suited to the unique characteristics of database provenance, including non-linear provenance objects and provenance associated with multiple fine granularities of data [7].

The Enterprise Resource Planning system (ERP system) manages database through application on the database top layer [8]. The investigation of the corporate frauds, gives output as investigation data from application software, which cannot be trusted. The database design in the form of dual-book may be used for hiding data; hence

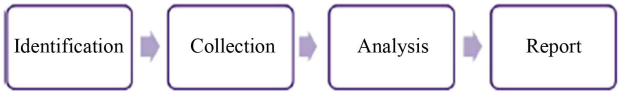

Figure 2. Investigation processes.

investigation data cannot be obtained from these systems. In such case the data is not only provided in the application software in the database, but there is additional data which is related with investigation. Therefore, the investtigators must verify what investigation data exists on the database. These fields are based on the database and application. The attribute information of these fields are stored in a database schema, through the DBRE (database reverse engineering), it is possible to determine the table relationship by extracting schema information. The DBRE largely divide two parts: data structure extraction and data structure conceptualization. Improving the DBRE process and hence analyzing the table relationship of database and data extraction method is useful from the view of the digital forensics.

There are different mechanisms within a database management system (DBMS), based on cryptographically strong one-way hash functions, which prevent an intruder, including an auditor or an employee or even an unknown bug within the DBMS itself, from silently corrupting the audit log [9]. DBMS transparently store the audit log as a transaction-time database, so that it is available to the application if needed. The DBMS should also store a small amount of addition information in the database to enable a separate *audit log validator* (to be referred to simply as the *validator*) to examine the database along with this extra information and state conclusively whether the audit log has been compromised. The DBMS periodically send a short document (a hash value) to an off-site *digital notarization service*, to bind when changes were made to a database [10].

The above methods have discussed about the models to identify the threat and hence detect the evidences in the vulnerable database. But our problem as a database forensic examiner to analyse the data to identify who, where and how tampered the data, is still not solved. With above methods we cannot gather the data for analysis. The later part of the paper summarizers the work of the researchers in the field of identifying locations for collecting the evidences.

2.4.3. Sources for Evidences

The author of [11] has listed the main source of evidence as follows:

1) Listener log—logs connections to the listener, use lsnrctl to administrate it. It Can be found in /u01/app/oracle/oracle/product/10.2.0/db_4/network/listener.log.

2) Alert log—system alerts important to DB e.g. processes starting and stopping. It Can be found in /u01/app/oracle/admin/orcl/bdump

3) Sqlnet.log—some failed connection attempts such as "Fatal NI connect error 12170".

4) Redo logs—current changes that have not been checkpointed into the datafiles (.dbf).

/u01/app/oracle/oradata/orcl/redo02.log
/u01/app/oracle/oradata/orcl/redo01.log
/u01/app/oracle/oradata/orcl/redo03.log

5) Archived redo logs—previous redo logs that can be applied to bring back the data in the db to a previous state using SCN as the main sequential identifier. This can be mapped to timestamp.

6) Fine-Grained Auditing audit logs viewable from FGA_LOG$ and DBA_FGA_AUDIT_TRAIL VIEW.

7) Oracle database audit SYS.AUD$ table and DBA_AUDIT_TRAIL VIEW.

8) Oracle mandatory and OS audit /u01/app/oracle/admin/orcl/adump

9) Home-made trigger audit trails—bespoke to the system.

10) Agntsrvc.log—contains logs about the Oracle Intelligent agent.

11) IDS, Web server and firewall logs should also be integrated to the incident handling timeline. This will rely heavily on well synchronised time in the network as previously mentioned.

Different locations [12] to find forensics data is listed below:

- TNS listener log
- Many types of trace files
- Sqlnet logs (server and clients)
- Sysdba audit logs
- Datafiles for deleted data
- Redo (and archive) logs
- SGA (v$sql etc)
- Apache access logs
- v$db_object_cache
- Wrh$%% views
- Wri$ views
- Statspack views
- col_usage$
- Audit trails –
  - AUD$, FGA_LOG$
  - Application audit (who/when, triggers, other)
- Flashback, recycle bin

The capabilities of LogMiners can be evaluated as a Forensics investigation tool. Its general applicability can be assessed for its forensics by testing how well it can create a timeline and copy of past database actions. LogMiner proves itself to be very useful for this purpose. The interpretation of the TIMESTAMP data types is also done by The LogMiners. Databases are excellent reporting mechanisms and LogMiner allows the analyst to use a database SQL interface to the redo logs of Oracle which are separate from the database itself.

At minimum this can provide verification of information found through normal database Auditing and at maximum could be the main source of information in an investigation and allow subsequent recovery of lost data [13].

3. Proposed System

Our proposed system is based on two basic steps in identifying the location, that is summarizes the flow of processes for collecting the evidences and we have also designed the block diagram summarizing the vulnerable locations in the database.

3.1. Identifying Locations for Evidence

A series of papers on performing a forensic analysis of a compromised Oracle database server is published by Mr. Litchfield. Based on these papers we have identified the locations where we may obtain the evidences for tamper detection.

Redo Logs

A Redo Entry, otherwise known as a Redo Record, contains all changes for a given SCN [14]. The entry has a header and one or more "change vectors". There may be one or more change vectors for a given event. For example, if a user performs an INSERT on a table that has an index then several change vectors are created. There will be a redo and undo vector for the INSERT and then an insert leaf row for the index and a commit. Each change vector has its own operation code that can be used to differentiate between change vectors. The table below lists some of the more common ones:

5.1 Undo Record
5.4 Commit
11.2 INSERT on single row
11.3 DELETE
11.5 UPDATE single row
11.11 INSERT multiple rows
11.19 UPDATE multiple rows
10.2 INSERT LEAF ROW
10.4 DELETE LEAF ROW
13.1 Allocate space [e.g. after CREATE TABLE]
24.1 DDL

The forensic examiner must go through each redo entry and work out what has happened and attempt to separate those which are "normal" and those which are part of an attack.

Data Blocks

The second paper [15] in this series is dedicated on Locating dropped objects. When the block is filled up, the server starts filling a new block. Each row in the block has a three byte header. The first byte is a marker and contains a set of flags to indicate the row's state. For example, if the row has been deleted the 5th bit of the byte is set. For example, a common set of flags value for a marker is 0 × 2C—which becomes 0 × 3C when the "deleted" flag is set. This is an important point to remember as it is a key indicator when looking for dropped objects. The second byte of the row header is used to determine lock status and the third byte indicates the total amount

of data in the row. If the total amount is greater than 255 bytes then the row header is four bytes allowing for up to 65,536 bytes. After the row header is the data itself. Each column of the row data is preceded with a byte indicating the size. If there is no data for a given column, in other words it is null, then it is represented with a 0 × FF.

TNS Listener's log file and the audit trail

To be able to log into the RDBMS an attacker [16] needs to know the Service Identifier or SID for the database. Before Oracle 10g this could be extracted from the TNS Listener with the SERVICES or STATUS command.

Here's something to be careful of with the audit trail. When a user successfully logs on a row is created in the audit trail. This has an ACTION# number of 100 (LOGON) and the TIMESTAMP# column reflects when the logon occurred.

In building a timeline of events this is important. This effectively hides when the user actually logged on. However, if we describe the AUD$ table we can see a LOGOFF$TIME column. If we then query this column, too, we can reconcile the logon and logoff times:

Live Response

When the database is shutdown cleanly this would wipe the audit trail making the task of the forensic examiner that little bit harder [17]. Of course, the attacker could do more than just wiped the audit trail in such a trigger. Due to issues like this and the loss of volatile information, some organizations prefer to perform an analysis on the system whilst it's still powered on and connected to the network. This is called a Live Response. Live Response is all about recovering and safely storing volatile data for later analysis, in other words, all the information that will disappear when the machine is disconnected from the network and switched off. Further, Live Response gives the forensic examiner the chance to collect non-volatile evidence in a "humanreadable" format that's easier to peruse than its stored binary version—for example event logs.

Views

There are a number of virtual tables and views that Oracle maintains for performance purposes [18]. These views are accessible to DBAs and can often contain evidence of attacks. Two of these views are of particular interest—V$SQL and V$DB_OBJECT_CACHE. The V$SQL fixed view contains a list of recently executed SQL. Evidence of an attacker's activities may be found in this fixed view and careful examination of the SQL_TEXT should reveal this. It must be stressed that if an attacker can find a way to execute arbitrary SQL as DBA, of which there are many, then they can clear the SQL from this view by executing "ALTER SYSTEM FLUSH SHARED_POOL". V$DB_OBJECT_CACHE contains details about objects in the library cache.

Oracle Recycle Bin

Whenever a table is dropped, the table and any [19] dependent objects such as indexes and triggers are moved to the Recycle Bin. This way, if it is decided that the table has been dropped in error, it can be recovered from the Recycle Bin using the UNDROP statement.

System Change Number

During a forensic examination of a compromised [20] Oracle database server the SCN and its timestamp can help tell the investigator whether a block of data has been changed. This is especially useful in those cases where there is an absence of other evidence such as the redo logs or audit trail. As with all forensic examinations it's critical not to change any evidence so any investigation should take place on a cold data file and not a live data file.

3.2. Steps to Collect the Evidence

With the help of the above study we have identified the steps which are useful in collecting the evidences [17].

1) Setup the evidence collection server by the following ways:

- firstly by mapping a drive if the system is running on Windows or has Samba and then using file redirection:
  D:\>listdlls.exe > z:\case-0001-listdlls.txt
  Using file redirection can be prone to error—for example the incident responder could type C instead of Z—which would be disastrous.
- The second method is to pipe output over the network using netcat or cryptcat.

2) Perform the following general steps to get basic information like [17].

***System time and date*:**

The incident responder should first record the system time and date of system that they're investigating.

***Logged on users*:**

The list of users that are currently logged on to the system and from where and for how long is extremely useful.

***List all users and groups*:**

Obtain a list of all users, gathering details on when they last logged in, and groups on the server and group membership.

***List open ports and connections* :**

All open and connected TCP ports should be collected as well as listening UDP ports.

***List running processes*:**

A list of all running processes should be obtained. Close attention should be paid to suspicious looking entries and also any shells such as cmd.exe or /bin/sh—indeed keep an eye out for //bin/sh (note two slashes) as this may indicate an overflow or format string exploit has been launched. The forensic examiner should also get a list of each process's parent process.

***List of DLLs or shared objects*:**

A list of the DLLs or shared objects that are loaded by each process should be obtained. Keep an eye out for odd looking names; on Windows look out for DLLs that are loaded via a UNC path across the network.

***List of open handles*:**

As well as what file handles a process has open a list of other handles should be obtained as well. Whilst this can reveal what an attacker may have been doing it can also help identify "parentless" processes.

***Perform memory dumps*:**

Memory dumps of all running process should be gathered even in what appear to be "normal" looking processes. The reason for this is to catch cloaking attacks—an attacker may launch a benign process like "notepad" and using CreateRemoteThread() load code into its address space.

***Perform system memory dump* :**

A dump of all system memory should be performed. This will cover those bit of memory not dumped when dumping each process.

***Get file names and MACTimes*:**

The incident responder should perform a full recursive directory list of every disk and get file and directory names as well as their creation, access and modification times. They should also gather information about each file's owner and any special attributes such as whether the read only, system or hidden attributes are set.

***Dump registry information*:**

On Windows all registry information should be dumped.

***Locate and take copies of log files and message logs*:**

All of the servers log files and event and message logs should be copied to the collection server for analysis. These logs will vary from system to system depending upon what services are running.

3) Collect the Oracle files of Interest

The Oracle specific log, trace and control files can be located in various places [17]. Firstly we need to know where each instance of Oracle is installed this can be extracted from the ORACLE_HOME environment variable if set. On Windows the HKEY_LOCAL_MACHINE\Software\Oracle Registry key stores information about each Oracle home For each Oracle home the incident responder should locate the server's start up parameter file. This will be found in the "database" directory on a Windows system or the "dbs" directory on a *nix system. Generally the filename is "spfilesid.ora" where "sid" is the database service identifier. This file contains information about where log and trace files etc are written to:

- audit_file_dest
- background_dump_dest
- core_dump_dest
- db_recovery_file_dest
- user_dump_dest
- utl_file_dir
- control_files
- db_create_file_dest
- db_create_online_log_dest_n
- log_archive_dest
- log_archive_dest_n

The incident responder should also be aware that what is listed in the start up file may not actually be what settings the Oracle server is actually using

4) Get the previously executed SQL [17].

- Get a copy of the most recently executed SQL. This can be retrieved from the V$SQL fixed view.
  On Oracle 10g the query should be:
  SQL> SELECT LAST_ACTIVE_TIME, PARSING_USER_ID, SQL_TEXT FROM V$SQL ORDER BY LAST_ACTIVE_TIME ASC;
  This will list the SQL that was executed by who and when from the V$SQL fixed view.
- Next in line should be the audit log. Everything should be selected from this table for later consumption and analysis.
  SQL > SELECT * FROM AUD$;

5) Getting a list of users and roles [17].

The incident responder should get a complete listing of all users on the system.

SQL > SELECT USER#, NAME, ASTATUS, PASS WORD, CTIME, PTIME, LTIME FROM

SYS.USER$ WHERE TYPE# = 1;

6) Getting a list of dropped tables [17].

In 10g, if a user has dropped any tables and they have not been purged from the recyclebin then a list of dropped tables should be present. This may indicate evidence of an attack:

SQL > SELECT U.NAME, R.ORIGINAL_NAME, R.OBJ#, R.DROPTIME, R.DROPSCN

FROM SYS.RECYCLEBIN$ R, SYS.USER$ U WHERE R.OWNER# = U.USER#;

7) Getting information about PL/SQL objects [17].

The source of PL/SQL objects should be retrieved and analyzed. Much of the source is encrypted or "wrapped" to use the Oracle term. The incident responder should obtain an "unwrapper" to examine the clear text as an attacker can modify a PL/SQL object and re-encrypt it to hide their attack.

8) Finishing up [17].

Once all queries have been executed the spool file should be closed and sqlplus can be closed.

SQL > SPOOL OFF

SQL > QUIT

Disconnected from Oracle Database 10g Enterprise Edition Release 10.2.0.2.0—Production with the Partitioning, OLAP and Data Mining options

C:\oracle\product\10.2.0\db_1\BIN>

Once disconnected from the server an md5 checksum should be made of the spool file and recorded with a witness present.

3.3. Block Diagram

The proposed process flow has been shown in **Figure 3**. This diagram summarizes the different methods explained above. We have considered a database server which has been tampered by an unauthorized user. To detect the tamper the forensic analyzer has to focus and collect information from the specified locations such as redo logs, data blocks, audit trails, live response, views, oracle recycle bin, and system change number. Different sql commands and tools are available to retrieve the information. The obtained information should be stored in a server and a comparative analysis should be performed on the basis of different kinds of users and their grant roles, role membership, object privileges, system privileges, authentication and authorizations of each and every user of the database. The analysis should be performed in the presence of an in charge and reliable authority of the organisation or the database. A graph should be produced to show the variations in the expected performance. The graph can be considered as the summarized output of the digital evidences collected for detecting the tampering in the database. Hence using this output forensic analysis of the database server can be performed which can be used to identify who, when and where tampered the data.

The locations defined in the proposed block diagram shown in **Figure 4** can be useful to get information about the system and its user and hence will give guidance to process the flow shown in **Figure 4**. Analyzing the flow may give us the desired information.

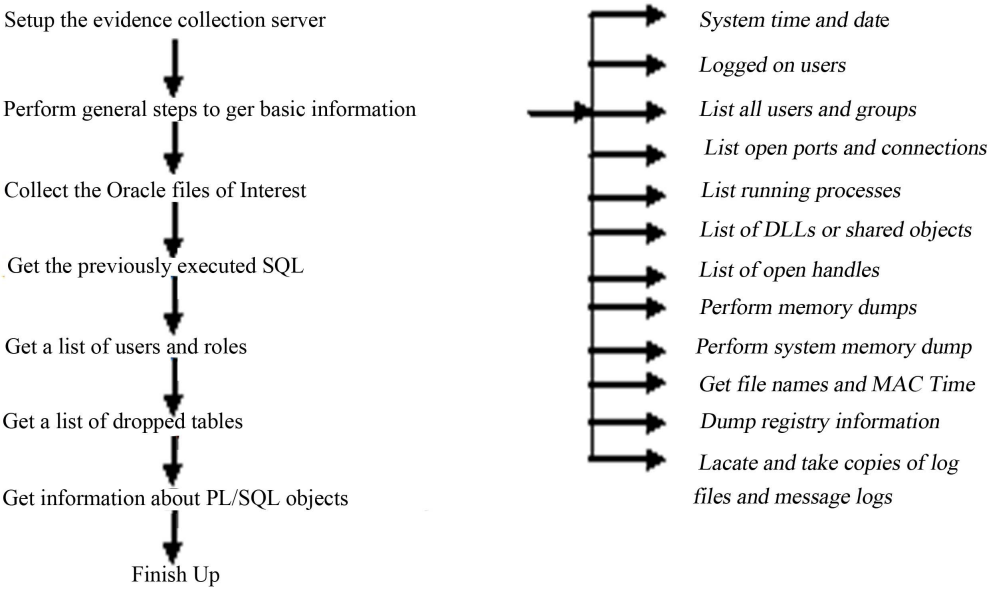

Figure 3. Process Flow.

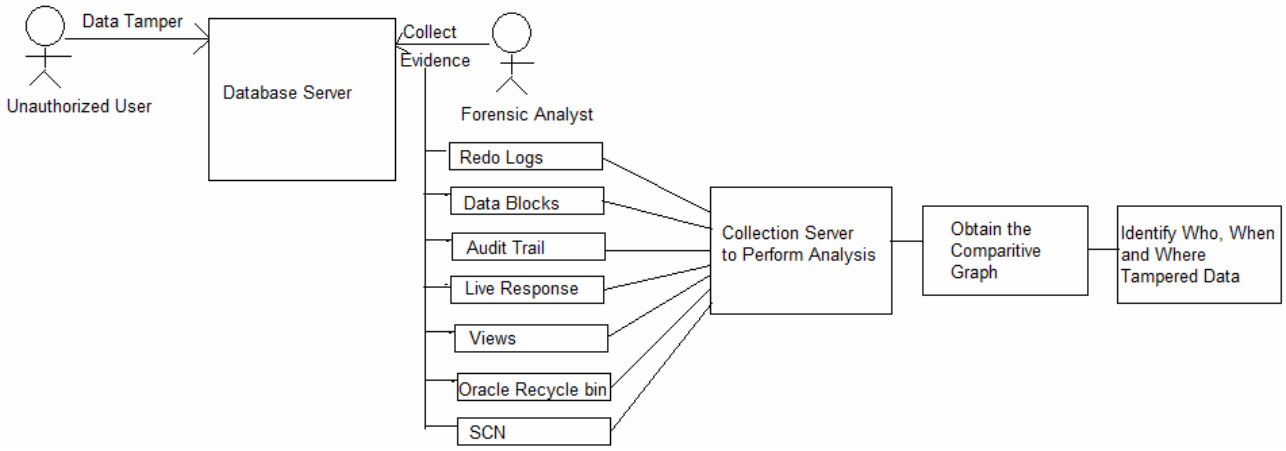

Figure 4. Proposed block diagram.

4. Conclusion

There are many ways of securing the database. The attackers have the methods to violate the security. Then comes the role of forensic analyst who should have a thorough knowledge of the basics of a database and also the information about the database on which he is going to perform the analysis. The forensic analyst should also be able to think from the attacker's point of view. Based on different cases, the digital evidences can be collected from the specified locations. If the intensions of the attacker are known identifying the attacked location may be easier. Thinking from the attacker's point of view this paper gives a contribution towards the identification of the general locations in a database for collecting the digital evidences.

REFERENCES

[1] K. Loney and B. Bryla, "Oracle Database 10g DBA Handbook," McGraw-Hill, New York, 2005.

[2] D. Litchfield, "Book on 'Oracle Forensics'," Wiley, Hoboken, 2008.

[3] O. L. Carroll, S. K. Brannon and T. Song, "Computer Forensics: Digital Forensic Analysis Methodology," *Computer*, Vol. 56, 2008, pp. 1-8.

[4] N. Aaron, "Oracle Database Security," *ICTN* 4040, Spring, 2006. doi:10.1.1.94.4146

[5] J. Azemović and D. Mušić, "Efficient Model for Detection Data and Data Scheme Tempering with Purpose of Valid Forensic Analysis," *Proceedings of the* 2009 *International Conference on Computer Engineering and Applications*, Manila, 6-8 June 2009.

[6] G. Miklau1 and D. Suciu, "Implementing a Tamper-Evident Database System," University of Massachusetts & University of Washington, Amherst & Washington DC, 2005.

[7] J. Zhang, A. Chapman and K. LeFevre, "Do You Know-Where Your Data's Been?—Tamper-Evident Database Provenance," *Proceedings of the* 6*th VLDB Workshop on Secure Data Management*, Lyon, 28 August 2009. doi:10.1007/978-3-642-04219-5_2

[8] D. C. Lee, J. M. Choi and S. J. Lee, "Database Forensic Investigation Based on Table Relationship Analysis Techniques," *Proceedings of the* 2*nd International Conference on Computer Science and Its Applications of the IEEE SCA*, Jeju, 10-12 December 2009, pp. 1-5. doi:10.1109/CSA2009.5404235

[9] M. J. Malmgren, "An Infrastructure for Database Tamper Detection and Forensic Analysis," Bachelor's Thesis, University of Arizona, Tucson, 2007.

[10] R. T. Snodgrass, S. S. Yao and C. Collberg, "Tamper Detection in Audit Logs," *Proceedings of the* 30*th International Conference on Very Large Data Bases*, Toronto, 31 August-3 September 2004.

[11] "Oracle Forensics In a Nutshell 25/03/2007," 2007.

[12] P. Finnigan, "Oracle Forensics," OUG Scotland, DBA SIG, 30 April 2008.

[13] P. M. Wright, "Oracle Database Forensics Using LogMiner Option 3—Perform Forensic Tool Validation," *Proceedings of the GCFA Assignment—GSEC*, *GCFW*, *and GCIH*, London, 10 January 2005.

[14] D. Litchfield, "Oracle Forensics Part 1: Dissecting the Redo Logs," *NGSSoftware Insight Security Research* (*NISR*), Next Generation Security Software Ltd., Sutton, 2007.

[15] D. Litchfield, "Oracle Forensics Part 2: Locating Dropped Objects," *NGSSoftware Insight Security Research* (*NISR*), Next Generation Security Software Ltd., Sutton, 2007.

[16] D. Litchfield, "Oracle Forensics Part 3: Isolating Evidence of Attacks against the Authentication Mechanism," *NGSSoftware Insight Security Research* (*NISR*), Next Generation Security Software Ltd., Sutton, 2007.

[17] D. Litchfield, "Oracle Forensics Part 4: Live Response," *NGSSoftware Insight Security Research* (*NISR*), Next Generation Security Software Ltd., Sutton, 2007.

[18] D. Litchfield, "Oracle Forensics Part 5: Finding Evidence of Data Theft in the Absence of Auditing," *NGSSoftware Insight Security Research* (*NISR*), Next Generation Security Software Ltd., Sutton, 2007.

[19] D. Litchfield "Oracle Forensics Part 6: Examining Undo Segments, Flashback and the Oracle Recycle Bin," *NGSSoftware Insight Security Research* (*NISR*), Next Generation Security Software Ltd., Sutton, 2007.

[20] D. Litchfield, "Oracle Forensics Part 7: Using the Oracle System Change Number in Forensic Investigations," *NGSSoftware Insight Security Research* (*NISR*), Next Generation Security Software Ltd., Sutton, 2008.

