A FRAMEWORK FOR DATABASE FORENSIC ANALYSIS

Harmeet Kaur Khanuja[1] and D.S.Adane[2]

[1]Department of Computer Engineering, Pune University, MS, India
harmeetkaurkhanuja@mmcoe.edu.in
[2]Department of IT, RKNEC, Nagpur
dattaadane@yahoo.com

ABSTRACT

Information security policy demands auditing for the high performance databases for ensuring data integrity and also to detect database tampering if any. Relational database uses auditing capabilities, which involves examination of information and operations for accuracy, legality and propriety to report risks and to make recommendations to promote sound-operating practices. Database auditing is the process to be carried out on continuous basis. This records and analyzes the database activity for reporting on some period. But the database can be tampered deliberately or accidentally by authorized or unauthorized users at any instance bypassing auditing system too. The suspected behaviour with invalid access to the database must be inspected and analyzed further with database forensics. In this research paper a framework is proposed for analyzing and reconstructing the activity of any unsuspicious behaviour within database. The purpose is to identify, collect, analyze, validate, interpret, generate forensic report and preserve the evidence for digital investigations. To prove in the concept, the database MySQL database 5.5 is studied and analyzed for this proposed framework.

KEYWORDS

MySQL, database, forensics, arifacts, logs, cache

1. INTRODUCTION

Today almost all applications use high performance databases to deal with data. So the database security community are coming up with a number of different techniques and approaches to assure data confidentiality, integrity, and availability. But it is observed that digital attacks are targeting the databases ensuing database security breaches and threats [1, 2 and 3]. The outcome is that the existing laws / regulations are specifying investigations and response to security breaches or policy violations. For example finalized HIPPA (Health Insurance Portability and Accountability Act of 1996) rules include “information security” which encompasses incident response describing the attempted or successful unauthorized access, use, disclosure, modification or destruction of information or interference with system operations in an information system. HIPPA specifies that there should be thorough analysis and reporting of security incidents [4]. This gives rise to the need for database forensics which satisfies this demand. Organizations must, therefore, consider their incidence response policies carefully, which are part of their overall security policies.

Database activity can be audited through a SQL trace in MS SQL server which is one of the security policies. This is an interface made available through extended stored procedures to

DOI : 10.5121/cseij.2012.2303

identify poorly running SQL statements, and to debug other performance problems. An application SQL Profiler collects the events. But only SQL Trace cannot be relied to monitor database. It is said that even Microsoft discourages the use of SQL Traces on a production system [5], because when enabled it can consume memory, CPU cycles, and disk space. Also SQL Trace does not audit or monitor systems on continuous basis. The traditional auditing system does not have intelligence built into it. It does not support filtering conditions. No amendment is possible to trace to what, when, or who is being audited. It is difficult to trace malicious activity. It is said that SQL Trace is great at amassing a huge amount of data, but is inadequate in finding the "needle in the haystack" that is evidence of malicious activity. Similarly in MySQL 'Information_Schema table' provides access to database metadata. The profilerEventHandler class in MySQL implements the interface that is used to handle profiling and tracing the events [6].

Secondly, audit systems are written to a local file or table which can be insecure because the data is not well protected. If a malicious activity is carried out, the suspected person can tamper with audit logs or can delete records leaving no traces of the activity. The same will happen for a DBA who wanted to perform actions they are not authorized to carry out. But as the database systems make numerous redundant copies of sensitive data items in table storage, indexes, logs, materialized views, and temporary relations, so when data is deleted, it is not destroyed but it often persists on disk [7, 8]. It is difficult to predict whether the sensitive data is lost after deletion or it has the possibility of recovery. These remnants of past data and activities pertaining to database systems can be revealed through database forensic analysis. This analysis does the process of extracting information and data from database internals basically from logs, cache, data files, tablespace, through trigger operation, view etc. Therefore, knowing the database management system structure is a precondition for forensic analysis. To analyze the data, it is important to know and understand in detail, how the database is built along with the understanding of forensics [9, 10]. With this knowledge one can identify and can come up to useful forensic investigation report in quality time. The extensive features for MySQL database 5.5 are highlighted in this paper. The analysis would be carried out on the forensic copies of database which is obtained using MySQL utility programs in our proposed framework. The database forensic investigations will reach the databases to determine precisely when the attack occurred, which data was compromised, from where the attack took place and fairly who performed the attack. The goal here is to design a methodology to validate hypothesis about past activities in a manner that is presentable in court.

A framework proposed in this paper works in two stages. To carry out the work in first stage detailed multiple logs of MySQL at the verification place are made available which is used for attribution. The logs grow into substantial sizes which hold important information. However, along with the useful information it also contains routine operational data which may not be required at the time of analysis. Extracting the useful information from logs which are needed for the target analysis is a challenging task. To do so, MySQL utility programs are used for making forensic copies of the databases and its multiple log files (text files, binary log files etc.) for analysis. These files of database are then parsed and read using Awk / Perl script written to give the relevant information as per the condition laid by the investigator. The expected outcome of the script is the metadata having traces of actions from the multiple files which helps to predict the identification and activities of unauthorized events carried out. Then based on the Inference rules laid down using expert knowledge the decisions are taken from the stored information in the derived metadata to get the relevant and filtered information for analysis. This is achieved by matching similar patterns and behaviour of the system concluded from metadata. A pre-final log analysis report is generated at this stage.

In the second stage we identify and reconstruct the activity from the various MySQL server artifacts identified and collected as described in section 5 for further analysis. The activity reconstructed has to be validated against the previously generated log analysis report to give a final forensic report.

2. RELATED WORK

Database Forensic is an essential area which must need research awareness. The lack of research is due to the inherent complexity of databases that are not fully understood in a forensic context yet. It is said that databases are inherently multidimensional from a forensic perspective [10]. The paper InnoDB Database Forensics shows how the MySQL tables in the .frm files are built and how important information is saved. The aim was to identify and name the bytes and interpret them. With that knowledge, it is possible to detect inconsistencies in the database. But there was no knowledge discovered for the multiple log files and cache for further analysis [11]. David Litchfield proposed a LogMiner tool which allows an Oracle DBA and/or Forensic analyst to reconstruct the actions taken on an Oracle database even if the auditing features have been turned off. LogMiner is a utility that can be used to analyse the redo log files that are created by an Oracle database [8, 9, and 12]. Dragoon (Database foRensic Analysis safeGuard Of arizONa) by Kyriacos E. Pavlou and R. T. Snodgrass is a prototype auditing system for tamper detection and database forensic analysis. This gives the mechanism to detect tampering of a database. It uses the methodology of cryptographically-strong hash functions and Digital Notarization paid services [13, 14, and 15]. A survey study is carried out in our previous work on "Database Security Threats and Challenges in Database Forensic" which highlights the work done by various researchers in the area of database forensics [16].

3. MYSQL INTERNALS

MySQL is the most popular Open Source SQL database management system. Figure 1 below shows main components of MySQL.

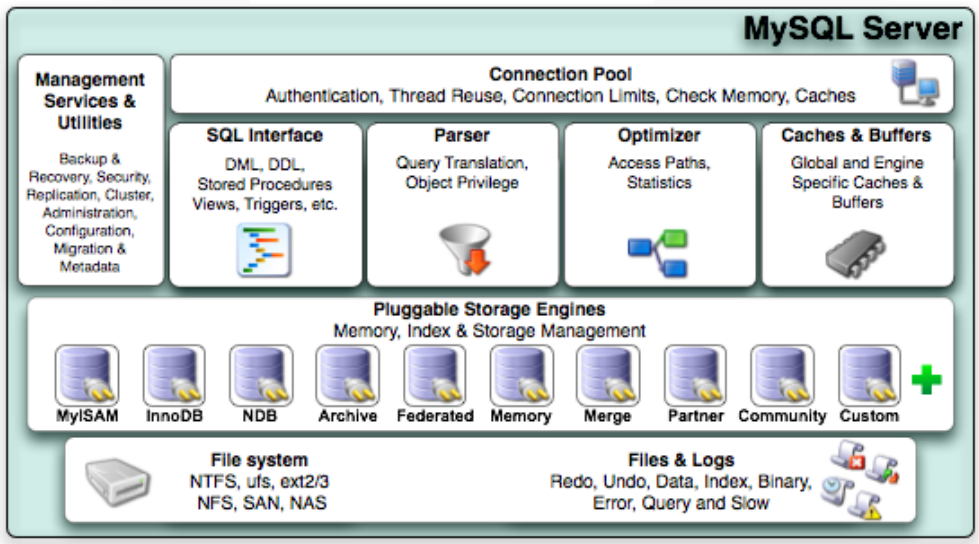

Figure 1. MySQL Architecture with Pluggable Storage Engines

The default storage engine for new tables in MySQL database 5.5 is InnoDB. [5, 20] MySQL data directory stores all the information managed by the MySQL server. It also stores all the databases, status files and logs. This includes features for transactions, foreign key support, replication, subqueries, stored procedures, views, and triggers. These capabilities take MySQL into the realm of enterprise applications. Most information can be collected from the Global and Storage engine specific caches and buffers. The storage management of these engines form the mines for database forensic investigation.

These database components should be well known for investigations. Some of the components like directory structures and some Log files from investigation point of view are highlighted below.

3.1 Structure of the Data Directory

The data directory is where the server stores its databases and status files. For Forensic investigation it is important to understand the structure and contents of the data directory so that the investigator knows how the server uses the file system to represent databases and tables, as well as where the server logs are located and what they contain.

Under Windows, the default data directory location often is
C:\ Program Files \ MySQL \ MySQL Server 5.0\ data or C: \ mysql \ data.
All the databases managed by the server are contained in the MySQL data directory. These are organized as shown in Figure 2 below.

- Each database has a database directory located under the data directory.
- Tables, views, and triggers within a database correspond to files in the database directory.
- The storage structure of storage engines can vary from the general hierarchical implementation of databases. For example, the InnoDB storage engine stores its tables within a single common tablespace. It then stores tablespace files in the data directory.

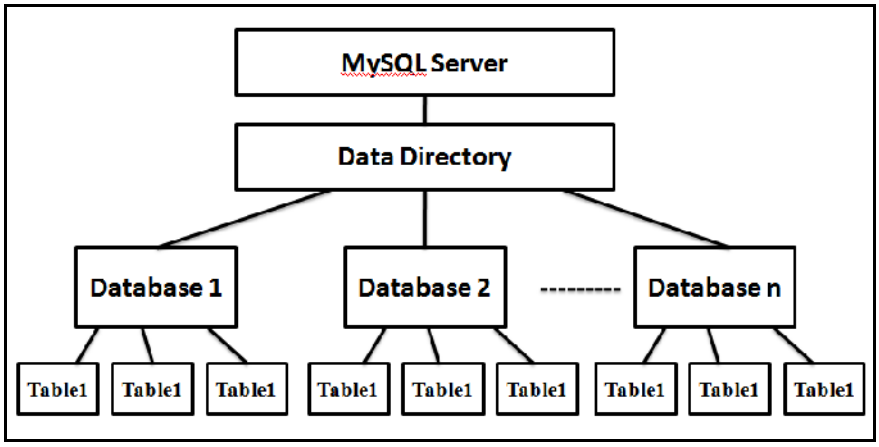

Figure 2. MySQL Data Directory Structure

The data directory also may contain other files:

- The server's process ID (PID) file. On start, the server writes its process ID to the file. This makes the other programs to discover the value for further processing with it.

- Server generated status and log files. These files provide important information about the server's operation and are valuable for administrators, especially when something goes wrong and you're trying to determine the cause of the problem. If some particular statement crashes the server, for example, you may be able to identify the offending statement by examining the logs. If the server is configured to log to database tables rather than to files, then the log tables are in the mysql database.

It also contains Server-related files, such as the DES key file or the server's SSL certificate and key files. It's common for administrators to use the data directory as the location for these files.

3.2 Information Schema

Information Schema gives access to the database metadata, information about the MySQL server such as the database or table name, column data type, or access privileges. The information about all the databases in MySQL server is stored in Information schema along with read-only tables. Example of a statement that retrieves information from INFORMATION_SCHEMA is shown below:

```
mysql> SELECT table_name, table_type,
engine
-> FROM information_schema.tables
-> WHERE table_schema = 'db5'
-> ORDER BY table_name;
+------------+------------+--------+
| table_name | table_type | engine |
+------------+------------+--------+
| fk | BASE TABLE | InnoDB |
| fk2 | BASE TABLE | InnoDB |
| goto | BASE TABLE | MyISAM |
| into | BASE TABLE | MyISAM |
| k | BASE TABLE | MyISAM |
| kurs | BASE TABLE | MyISAM |
| loop | BASE TABLE | MyISAM |
| pk | BASE TABLE | InnoDB |
| t | BASE TABLE | MyISAM |
| t2 | BASE TABLE | MyISAM |
| t3 | BASE TABLE | MyISAM |
| t7 | BASE TABLE | MyISAM |
| tables | BASE TABLE | MyISAM |
| v | VIEW | NULL |
| v2 | VIEW | NULL |
| v3 | VIEW | NULL |
| v56 | VIEW | NULL |
+------------+------------+--------+
17 rows in set (0.01 sec)
```

The query above requests a list of all the tables in database db5, retrieving information such as table name, its type, and its storage engine. Each MySQL user has the right to access these tables, but can see only the rows in the tables that correspond to objects for which the user has the proper access privileges. Users who have insufficient privileges will see NULL.

3.3 MySQL Status and Log Files

Logs generate substantial amounts of information almost loading the disks. MySQL data directory contains a number of status and log files, as summarized in Table 1 shown below. MySQL Server maintains logical logs and storage engine maintains physical or logical logs (additionally). The server's data directory is the default location for each file with its default name (with HOSTNAME). The table lists only the server-level status and log files. Individual storage engines may have their own logs or other files.

File Type	Default Name	Facts and File Contents for investigation
Process ID file	HOSTNAME.pid	The server process ID
Error log	HOSTNAME.err	Information of Startup and shutdown events and error conditions
General query log	HOSTNAME.log	Connect/disconnect events and statement information
Binary log	HOSTNAME-bin.nnnnnn	Binary representation of statements that modify data
Binarylog index	HOSTNAME-bin.index	Contains the List of current binary log files
Relay log	HOSTNAME-relay-bin.nnnnnn	Any Data modifications received by slave server from master
Relay log index	HOSTNAME-relay-bin.index	List of current relay log files
Master info file	master.info	It contains the parameters used for connecting to the master server
Relay info file	relay-log.info	Contains the Status of relay log processing
Slow-query log	HOSTNAME-slow.log	Text of statements that take a long time to process

Table 1. MySQL Status and Log Files

The InnoDB engine has two types of logs namely an undo log and a redo log. The undo log rolls back transactions. It also displays the older versions of the data. The redo log stores the information which is utilized at the time of crash recovery process. It gives permission to the recovery process so that the transactions are re-executed that could or could not have completed before the crash. Once it re-executes those transactions, the database gets into a consistent state.

Log files are most important data facts for investigation, as they contain the text of statements that include sensitive information such as passwords. For example, the following log entry displays the password for the root user; it's certainly not the kind of information you want just anyone to have access to:

```
080412 16:47:24      44 Query      SET PASSWORD FOR
                    'root'@'localhost'=PASSWORD('secret')
```

The server writes log files to the data directory by default. The general query log is one of the most useful logs to monitor the server.

3.4 MySQL Utility Programs For Forensic Analysis

Some MySQL utility programs are studied and described here which can be used at the time of investigations.

3.4.1. mysqldump — A Database Backup Program

The mysqldump client is a backup program. It can be used to dump a database or a collection of databases for backup or transfer to another SQL server. The dump typically contains SQL statements to create the table, populate it, or both. This utility program can also be used to generate files in XML format.

XML output from mysqldump includes the XML namespace, as shown here:

```
shell> mysqldump --xml -u root world City
<?xml version="1.0"?>
<mysqldump xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<database name="world">
<table_structure name="City">
<field Field="ID" Type="int(11)" Null="NO" Key="PRI" Extra="auto_increment" />
<field Field="Name" Type="char(35)" Null="NO" Key="" Default="" Extra="" />
<field Field="CountryCode" Type="char(3)" Null="NO" Key="" Default="" Extra=""
/>
<key Table="City" Non_unique="0" Key_name="PRIMARY" Seq_in_index="1"
Column_name="ID"
Collation="A" Cardinality="4079" Null="" Index_type="BTREE" Comment="" />
<options Name="City" Engine="MyISAM" Version="10" Row_format="Fixed"
Rows="4079"
Avg_row_length="67" Data_length="273293"
Max_data_length="18858823439613951"
Index_length="43008" Data_free="0" Auto_increment="4080"
Create_time="2007-03-31 01:47:01" Update_time="2007-03-31 01:47:02"
Collation="latin1_swedish_ci" Create_options="" Comment="" />
</table_structure>
<table_data name="City">
<row>
<field name="ID">1</field>
<field name="Name">Kabul</field>
<field name="CountryCode">AFG</field>
</row>
...
<row>
</table_data>
</database>
</mysqldump>
```

The example above shows that this utility can retrieve and dump table contents row by row or it can retrieve the entire content from a table in XML format and buffer it in memory before dumping it. This XML can be read using SAX parser to fetch the required information for Forensics like created time, updated time of the table etc.

3.4.2. mysqlaccess — Client for Checking Access Privileges

The mysqlaccess is a diagnostic tool for the MySQL distribution for checking the access privileges. The privileges' are defined for a host name, user name, and also database combinations. It also checks access using only the user, database, and host tables. It does not check table, column, or routine privileges.

3.4.3. myisamlog — Display MyISAM Log File Contents

The contents of a MyISAM log file are processed by the utility program myisamlog. It has the options to perform a recovery operation, specify record position file and record position, perform an update operation, display version information etc.

3.4.4. myisamchk — MyISAM Table-Maintenance Utility

The myisamchk utility gets information about your database tables or checks, repairs, or optimizes them. The myisamchk utility works with MyISAM tables (tables that have .MYD and .MYI files for storing data and indexes).

3.4.5. mysqlbinlog — Utility for Processing Binary Log Files

The server's binary log consists of files containing "events" that describe modifications to database contents. The server writes these files in binary format. To display their contents in text format the mysqlbinlog utility is used. This utility can also be used to display the contents of relay log files written by a slave server in a replication setup because relay logs have the same format as binary logs.

3.4.6. mysqlbinlog Hex Dump Format

The --hexdump option causes mysqlbinlog to produce a hex dump of the binary log contents:

```
shell> mysqlbinlog --hexdump master-bin.000001
```

The hex output consists of comment lines beginning with #, so the output might look like this for the preceding command:

```
# at 4
#051024 17:24:13 server id 1 end_log_pos 98
# Position Timestamp Type Master ID Size Master Pos Flags
# 00000004 9d fc 5c 43 0f 01 00 00 00 5e 00 00 00 62 00 00 00 00 00
# 00000017 04 00 35 2e 30 2e 31 35 2d 64 65 62 75 67 2d 6c |..5.0.15.debug.l|
# 00000027 6f 67 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |og..............|
# 00000037 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
# 00000047 00 00 00 00 9d fc 5c 43 13 38 0d 00 08 00 12 00 |.......C.8......|
# 00000057 04 04 04 04 12 00 00 4b 00 04 1a |.......K...|
# Start: binlog v 4, server v 5.0.15-debug-log created 051024 17:24:13
# at startup
ROLLBACK;
```

The hex output obtained from the binary logs can be used to find similar patterns which can be used to analyze the behaviour of the system.

4. LOGS AND FILE ANALYSIS

Log files are often very large and at times have complex structure. While the process of generating log files is quite simple and straightforward through the utility programs of the database, but its analysis could be a tremendous task that requires enormous computational resources, long time and sophisticated procedures. The log files continuously grow into huge sizes which hold in most cases a bulk of the information with normal operational data which is of less importance in an analysis. Extracting the useful information from logs needed for the target

analysis is one of the challenging and a difficult task [21]. Forensic analysis procedures often demand extracting information from the maximum log files and correlating them to have a broader understanding of the case.

A framework is designed and proposed for this research work. The system architecture for the logs and file analysis is shown in Figure 3 below.

Tasks to be carried out for database forensic analysis would be as:

1. Identify and collect the databases, log files, binary logs and text files (.MYD, .MYI, .FRM etc.) at database server using MySQL Utility programs.
2. The MySQL utility programs are used to dump the database to make a backup copy for analysis. The detailed information like user access, timestamp, date etc. is to be traced. The parsers (Awk/Perl) [19] will be used to read huge and multiple log files and text files. The basic function of awk will be to search files for lines (or other units of text) that contain certain patterns. When a line matches one of the patterns, awk performs specified actions on that line. It keeps processing input lines in this way until it reaches the end of the input files.

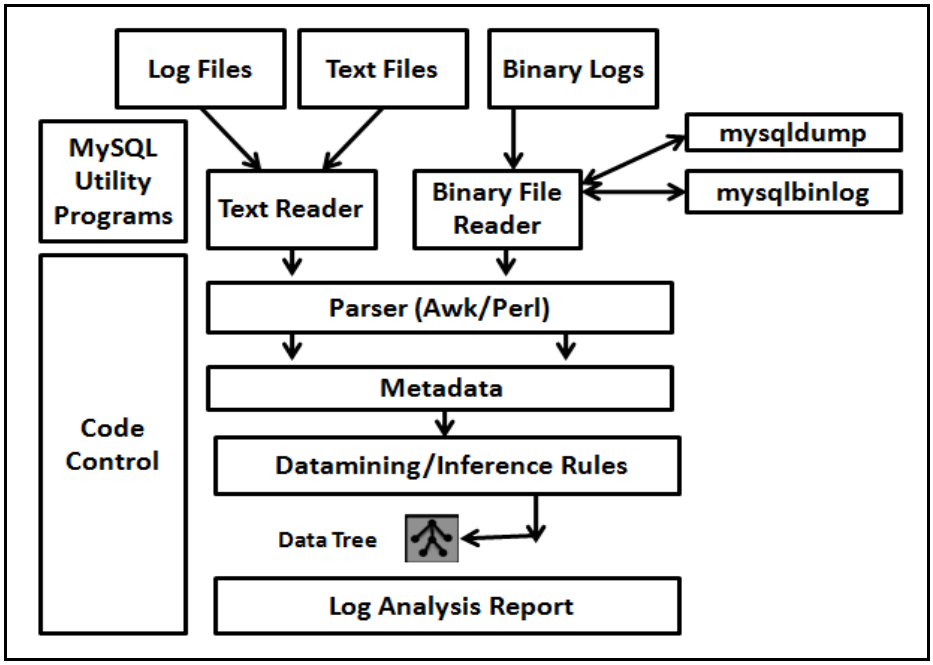

Figure 3. System Architecture for database forensic analysis: Stage 1

3. Binary log information is first retrieved using MySQL utility programs like mysqlbinlog and is then given to the binary reader.
4. The raw data is collected to form Metadata. It contains the collection of facts from the extracted log data/ system data.
5. Data mining techniques is then used to build inference rules based on expert knowledge to make the decisions for extracting most relevant information from the built up Metadata [22]. Failure to make good inferences may delay the analysis in best case and will hamper it from generating any solution to the problem in worst.
6. Finally a detailed log analysis report is generated.

7. The proposed framework would promote reuse from its architecture to facilitate expert knowledge dissemination. It once formed will give efficient knowledge transfer mechanism. This will help in automation.
8. The script (control code) is written for this complete process.

5. ARTIFACT COLLECTION IN MYSQL SERVER

A forensic methodology is a logical and well-thought-out order of operations that is executed during a digital investigation. Forensic methodologies help ensure investigations are documented, repeatable, and executed in a manner that is court friendly, should the collected data need to be submitted as evidence in a court of law [18].

MySQL Server artifacts reside within operating system files and areas of memory that are explicitly reserved for SQL Server use. These data facts can exist within large, core MySQL Server files, such as database data or transaction log files, or within smaller, less visible files. These artifacts form the prime collection of data that can be used for database investigation. There are numerous artifacts, each of which will benefit a MySQL Server investigation in a different way. Some of the data facts identified are described here.

5.1. Query Cache

MySQL has a unique feature for a database that is a query cache. The server is configured to cache the results of every SELECT.

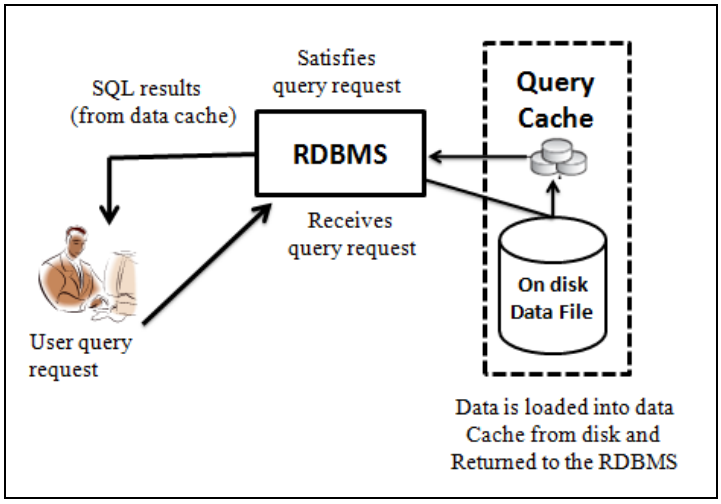

Figure 4. Retrieval from Query Cache

If the tables used in the query are same, then the result cached is returned at once instead of MySQL actually fetching the tables by searching queried records. MySQL has transaction descriptor for query cache which is used to manage logical update logging and keeping track of changed tables. The query cache is able to easily tell if the table has changed or not. Any storage engine, transactional or not, needs to be able to work correctly with the query cache as shown in Figure 4 above.

Usage: Activity Reconstruction

By preserving and examining the query cache, recently accessed data pages can be identified. During investigations involving the potential disclosure of database data, these pages can be used to gain an understanding of the data an attacker may have accessed within the database. The cache can be ideal for qualifying suspected SQL Server misuse, such as unauthorized actions performed by an insider and SQL injection or buffer truncation attacks. Once unauthorized access to a database server is confirmed or suspected, the cache can be analyzed to identify previously executed SQL statements. This information can be used to reconstruct past SQL execution history resulting from ad hoc queries, stored procedures, or function execution.

The other caches for MySQL are Key cache, record cache, Table cache, Hostname cache, privilege cache, Heap table and Join buffer cache. The information of various table keys, the records in the table, most recently used table information, Hostname login, the information of last used privileges are cached for each user/database combination.

Usage: Activity Reconstruction, Authentication and Authorization

Analyzing authentication and authorization data will allow you to identify MySQL Server permission assignments throughout the permission hierarchy. It will also allow you to determine effective permission, which is the level of access a user had within the database server.

5.2. Other MySQL cache

5.2.1. Key Cache

It is a shared cache for all B-tree index blocks found in different ISAM files. It supports quick caching of the most recently used blocks. It also supports quick flushing of changed entries for a specified table.

5.2.2. Record Cache

All the records present in a table can be quick scanned by the record cache.

5.2.3. Table Cache

This holds the most recently used tables.

5.2.4. Hostname Cache

It is used for quick lookup (with reverse name resolving). This is used with slow DNS.

5.2.5. Heap Table Cache

Many uses of GROUP BY or DISTINCT caches all found rows in a HEAP table with hash indexing.

Usage: Activity Reconstruction

By preserving and examining the respective above cache, recently accessed user data, rows and tables can be identified and can be used in reconstructing the activity.

5.2.6. Privilege Cache

The recent used privileges are cached for each user or database combination which allows immediate change between databases.

Usage: Activity Reconstruction, Authentication and Authorization

Analyzing authentication and authorization data will allow you to identify last used privileges. It will also allow you to determine effective permission, which is the level of access a user had within the database server.

5.3. Triggers

Triggers consist of pre developed SQL syntax that is automatically executed in response to DDL operations such as the creation of table or DML statements such as inserting, updating, or deleting table data.

Usage: Activity Reconstruction

An attacker can use triggers to record or even alter table operations. For example, an attacker may place a trigger on a payment refund table such that each time a payment refund is written to the table, the trigger intercepts the write operation, changes the account number to be refunded to that of the attacker's choosing, and then writes the data to the table. Analyzing trigger data can identify triggers created or updated during the timeline of an attack that warrant further investigation.

5.4. Data Files

MySQL creates files for the created Table as Table1 .MYD ("MySQL Data"), Table1.MYI ("MySQL Index"), and Table1.frm ("Format"). These files will be in the directory.

Usage: Data Recovery

When table data is deleted, the data is hidden rather than actually being purged from the system. By analyzing MySQL Server data files, you can often recover previously deleted table data. Data files can also be attached to a trusted forensic machine and used to support activity reconstruction artifact analysis.

5.5. InnoDB tablespace

The InnoDB workspace has tablespace and log files. A tablespace contains a segment which is generally a file or it can be a raw disk partition. A segment has extents of 64 pages. A page 16KB in length is made up of a page header with some rows.

InnoDB has two logs namely the redo log and the undo log. The redo log is for modifying data before written to disk at the time of crash. There is one redo log which is defined for the entire workspace, it contains multiple files and works in circular. The file header highlights the last successful checkpoint.

Usage: Activity Reconstruction

Often during a database intrusion, traces of an attacker's actions will be found within the tablespace artifact. Whether an attacker creates a table to store stolen data or a procedure or function to execute malicious code, these actions can be quickly pinpointed by reviewing the tablespace and its log fies.

6. ARTIFACT ANALYSIS

During artifact analysis, all data acquired through the incident verification and collection phases are consolidated and analyzed. Notable events such as failed database login attempts, successful user logins, and anomalous database activity can be identified and added to an investigation timeline. This timeline will aid an investigator in identifying activity patterns and related database activity, which may not be sequentially logged within collected log files. These artifacts are collections of related Server data. Many operations leave a footprint within facts, which make these facts an invaluable resource during an investigation. A good understanding of MySQL Server artifacts is required to satisfy the objective of an investigation.

The analysis starts with a set of log analysis report as shown in system architecture in Figure 3.

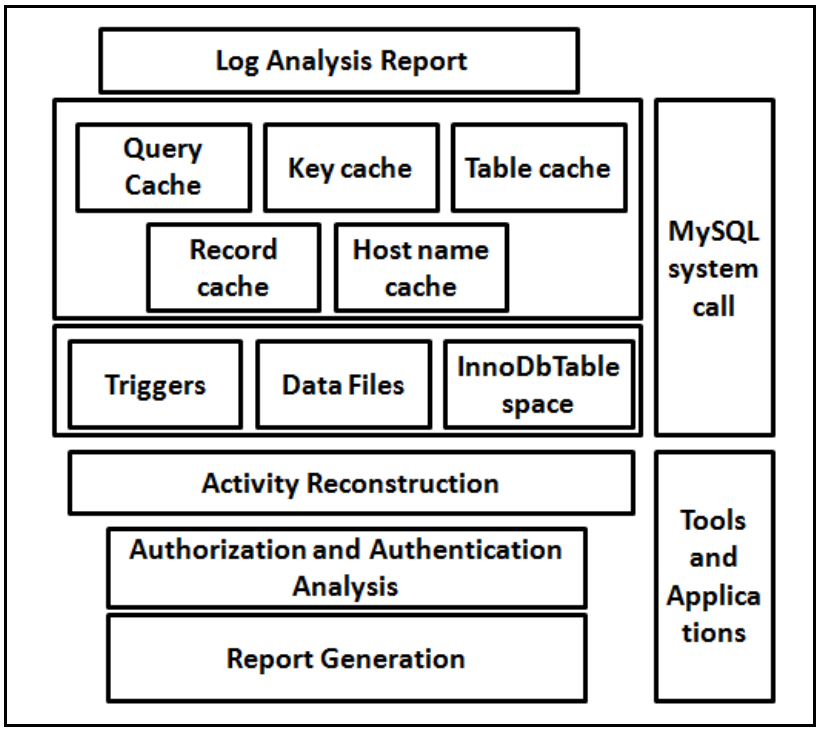

Figure 5. System Architecture for database forensic analysis: Stage 2

A script would be written to read and extract data from each log files. The output yields a tree containing the data extracted from the corresponding log file based on the inference rules which can be subsequently applied to get a resulting tree which holds the facts that can be used in further analysis or making decisions. The decisions are then processed with the various artifacts identified and collected as shown in Figure 5 Stage 2.

The artifacts (multiple cache, triggers, data files and InnoDb Tablespace etc.) are identified, collected and validated against the Log analysis report. A forensic methodology is carried out for further investigations for Activity Reconstruction and analyzing the various authenticated and authorized users. Finally a Forensic report is generated with the tool and application developed.

7. CONCLUSION AND FUTURE WORK

Most organizations would not have a separate policy for forensics, either due to lack of awareness about importance of database forensics or due to budgetary issues. Thus this paper makes familiar with the concept of database forensics and proposed a framework which builds the expert system for database analysis in two stages. To prove in the concept MySQL database 5.5 is used here. To interpret the data one has to know a lot about the MySQL Internals. Thus we highlighted some components of MySQL from investigation point of view. The problem which persists in auditing system where there is no intelligence built into it can overcome with our proposed framework. It will give add on features to auditing system to built and retrieve meaningful results in quality time. Thus we contend that determining the identity of the user can be revealed through Database Forensics. In this paper, the framework is proposed for MySQL which would be implemented to generate the Forensic reports. Similarly the framework can be modified and reused for the other DBMS with its own identified artifacts.

REFERENCES

[1] Sohail Imran, Dr. Irfan Hyder. (2009) “Security Issues in Databases”, Second International Conference on Future Information Technology and Management Engineering, IEEE.
[2] John oltsik. (2009), “Database security and Compliance Risk”, ESG Market research study, Application Security, Inc.
[3] U.S. health insurance portability and accountability act (HIPAA). Available at www.hhs.gov/ocr/hipaa
[4] By Aaron C. Newman, CTO & Founder (2005), “Security Auditing In Microsoft SQL Server”, Application Security, Inc.
[5] “MySQL 5.5 Reference Manual”, www.dev.mysql.com/doc/refman/5.5/
[6] Nina Godbole and Sunit Belapure. (2011) “Cyber Security, Understanding Computer Forensics and Legal Perspectives”, Wiley-India. ISBN: 978-81-265-2179-1.
[7] Jasmin Azemovic and Denis Music.(2010). “Methods for Efficient Digital Evidences Collecting of Business Processes and Users Activity in eLearning Environments”, IEEE, 2010 International Conference on e-Education, e-Business, e-Management and e-Learning.
[8] Article by David Litchfield (2011, August), www.darkreading.com/database-security/167901020/security/attacks-breaches/231300307/database-forensics-still-in-dark-ages.html.
[9] Article by David Litchfield (2011, August), http://www.computerweekly.com/Articles/2007/08/03/225987/New-database-forensics-tool-could-aid-data-breach-cases.htm
[10] Martin S. Olivier. (2009, March), “On metadata context in Database Forensics, Digital Investigation”, Elsevier, www.sciencedirect.com, Volume 5, Issues 3-4, Pages 115-123.
[11] Peter Frühwirt, Markus Huber and Martin Mulazzani, Edgar R. Weippl (2010), “InnoDB Database Forensics”, 24th IEEE International Conference on Advanced Information Networking and Applications.
[12] Paul M. Wright, (2005) “Oracle Database Forensics using LogMiner”, June 2004 Conference, SANS Institute 2005
[13] K. E. Pavlou and R. T. Snodgrass. (2010, April), “The Tiled Bitmap Forensic Analysis Algorithm” IEEE Transactions on Knowledge and Data Engineering, 22(4):590-601.
[14] Kyriacos Pavlou & Richard T. Snodgrass. (2006) “Forensic Analysis of Database Tampering”, International Conference on Management of Data, Proceedings of the ACM SIGMOD International Conference on Management of data, SESSION: Authentication, Pages: 109 – 120.

[15] Kyriacos Pavlou, (2011) Database Forensics in the Service of Information Accountability, http://www.cs.arizona.edu/projects/tau/dragoon/
[16] Harmeet Kaur Khanuja and Dr. D. S. Adane (2011), "Database Security Threats and challenges in Database Forensic: A survey", Proceedings of 2011 International Conference on Advancements in Information Technology (AIT 2011), available at http://www.ipcsit.com/vol20/33-ICAIT2011-A4072.pdf
[17] Dongchan Lee, Jaemin Choi, Sangjin Lee (2009), "Database Forensic Investigation based on Table Relationship Analysis Techniques", Centre for Information Security and Technologies, Digital Forensic Research Center, Korea University, Seoul, Republic of Korea.
[18] Kevvie Fowler, "SQL Server Forensic Analysis",ISBN:9780321533203, Addison–Wesley-2009
[19] Arnold D. Robbins,(Edition 4 June, 2011) "GAWK: Effective AWK Programming", A User's Guide for GNU Awk.
[20] Sasha Pachev, "Understanding MySQL Internals", Copyright © 2007 O'Reilly Media, Inc
[21] J. Valdman. "Log file analysis", (2001) Technical Report DCSE/TR-2001-04, Department of Computer Science and Engineering (FAV UWB).
[22] H. Saneifar, S. Bonniol, A. Laurent, P. Poncelet. (2009), "Terminology extraction from log files", In: KDIR'09. Proc. Of 20th International Conference on Database and Expert Systems Applications. pp. 769-776. Lecture Notes in Computer Science, Springer 2009.

Authors Profile

Harmeet Kaur Khanuja, currently a Researcher in the area of Database Forensics. She is working as Assistant Professor in the Department of Computer Engineering at Marathwada Mitra Mandal's College of Engineering. Pune, India. She is a life member of ISTE. She has presented and published papers in several International Conferences and Journals. Her areas of interest are Information security Applications, Digital Forensics and Mobile Computing.

Dr. D. S. Adane, currently is a Professor and Head of Information Technology Department at Ramdeobaba College of Engineering and Management, Nagpur, India. He received Ph.D. in Computer Science and Engineering from VNIT Nagpur, India. He has over 15 research papers to his credit in reputed International Journals / Conferences and also reviewed the papers for many. His research interests include Distributed and Mobile Computing, Mobile Agents and Network Security.