



# Global Information Assurance Certification Paper

Copyright SANS Institute  
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

## Interested in learning more?

Check out the list of upcoming events offering  
"Advanced Incident Response, Threat Hunting, and Digital Forensics (Forensics  
at <http://www.giac.org/registration/gcfa>

Table of Contents .....	1
Paul_Wright_GCFA.pdf .....	3
0 Abstract.....	4
1 Contents.....	4
2 Introduction.....	5
3 Summary of supporting Oracle DBA concepts.....	6
4 Oracle Security.....	8
5 Scope LogMiner testing.....	9
6 Tool Description .....	11
7 Test Apparatus.....	15
8 Environmental conditions.....	17
9 Testing overview .....	17
10 Description of the Procedures - preparation.....	18
10.1.1 Logging on and starting Oracle.....	18
10.1.2 Setting up LogMiner.....	18
10.1.3 Setting up SQL* Plus output.....	20
10.1.4 Archiving the redo logs .....	20
10.1.5 Chain of custody.....	20
11 Test 1 - General forensic capability.....	21
11.1 Criteria for approval and tool test execution .....	21
11.1.1 Time line test execution.....	21
11.2 Data and results.....	22
11.3 Analysis.....	24
12 Test 2...TIMESTAMP accuracy of timeline and recovery .....	25
12.1 Timeline TIMESTAMP precision.....	25
12.2 Integrity of TIMESTAMP recovery. ....	26
12.3 Analysis of test 2.....	28
13 Test 3 ... LogMiners error or the logs?.....	30
14 Presentation.....	33
15 Conclusions.....	35
16 Cross Examination .....	37
17 Additional Information.....	38
18 References.....	38
18.1 URLs used during the paper all active 10 th January 2005 23:27 .....	39
19 Appendices.....	41
19.1 LogMiner errors from Metalink.....	41
19.2 SANS Guide for Oracle Security sections relevant to LogMiner.....	41

# Oracle Database Forensics using LogMiner

## Option 3 - Perform Forensic Tool Validation

### GCFA Assignment Version 2.0



© SANS Paul M. Wright - GSEC, GCFW, GCIH

January 10<sup>th</sup> 2005 from London June 2004 Conference

# 0 Abstract

This paper is an evaluation of LogMiners capabilities as a Forensics investigation tool. It starts by assessing its general applicability to 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. Then the paper focuses on LogMiners interpretation of the TIMESTAMP data types. It has been found that fractional seconds data are not present in the reporting the TIMESTAMP column because it is actually a DATE data type. Also it has been found that fractional seconds data values are lost from the original database when recovered by LogMiner. These problems have not been noted in the public domain before so this should be of some contribution to the field of Oracle database forensics.

# 1 Contents

1	Contents .....	2
2	Introduction .....	3
3	Summary of supporting Oracle DBA concepts .....	4
4	Oracle Security .....	6
5	Scope LogMiner testing .....	7
6	Tool Description .....	9
7	Test Apparatus .....	13
8	Environmental conditions .....	15
9	Testing overview .....	15
10	Description of the Procedures - preparation .....	16
10.1.1	Logging on and starting Oracle .....	16
10.1.2	Setting up LogMiner .....	16
10.1.3	Setting up SQL* Plus output .....	18
10.1.4	Archiving the redo logs .....	18
10.1.5	Chain of custody .....	18
11	Test 1 - General forensic capability .....	19
11.1	Criteria for approval and tool test execution .....	19
11.1.1	Time line test execution .....	19
11.1.2	Database recovery execution .....	20
11.2	Data and results .....	20
11.3	Analysis .....	22
12	Test 2-TIMESTAMP accuracy of timeline and recovery .....	23
12.1	Timeline TIMESTAMP precision .....	23
12.2	Integrity of TIMESTAMP recovery .....	24
12.3	Analysis of test 2 .....	26
13	Test 3 – LogMiners error or the logs? .....	28
14	Presentation .....	31
15	Conclusions .....	33
16	Cross Examination .....	35
17	Additional Information .....	36
18	References .....	36
18.1	URLs used during the paper all active 10 <sup>th</sup> January 2005 23:27 .....	37
19	Appendices .....	39
19.1	LogMiner errors from Metalink .....	39
19.2	SANS Guide for Oracle Security sections relevant to LogMiner .....	39
	Figure 1 MD5 Hash Checksums of the files that make up LogMiner on RedHat .....	10
	Figure 2 Version of LogMiner for Solaris 10 .....	11
	Figure 3 Version of LogMiner for RedHat Enterprise Version 3 ES .....	11
	Figure 4 MAC timelines with their corresponding MD5 hash checksums .....	20
	Figure 5 A timeline example created by Oracle reports (Oracle.com) .....	31
	Table 1 A relational database table .....	4
	Table 2 parameter commands for archiving from Oracles documentation .....	12
	Table 3 Hardware and software specifications .....	13

## 2 Introduction

Forensics can be defined as: *“Of or used in connection with a court of law in relation to the detection of a crime... involving the use of forensic science.”* (1999 Oxford English Dictionary)

Forensics is often connected to medicine but now more so to computer science and IT security in particular. Computer forensics being a younger field than its medical relation is still evolving its techniques but shares similar scientific principles to its more elderly relation. Scientific methods rely on objective observations; accurate, precise measurement and experimentation that is thoroughly documented, repeatable and verifiable independently. Therefore scientific results tend to be given more credence if they have been scrutinised openly by the related scientific community.

Database forensics and specifically Oracle forensics is a new field and one which I feel, privileged to be a part of. Oracles unrivalled reputation for database software is richly deserved and I hope that this work will help add to that reputation.

My choice of tool for evaluation is Oracles LogMiner utility. The LogMiner tool 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. The LogMiner tool has been suggested by Oracle security expert, Pete Finnigan for the forensic investigation of database logs at <http://www.petefinnigan.com/orasec.htm>. LogMiner forensics is also part of the new SANS Oracle Security Track at <http://www.sans.org/sans2005/description.php?cid=534>. This is an active and exciting area of investigation. There has not been, to my knowledge, a validation of LogMiners forensic abilities in terms of repeatability, verifiability, integrity, accuracy, precision and general use in order to provide admissible evidence in court. Certainly there is none at NIST and GIAC and there has not been a publicly available confirmation of LogMiners hashes at the NSRL.<sup>1</sup>

During the process of this testing I have been able to validate LogMiner as being very useful for the purpose of forensic investigation. However I have also discovered two anomalies with the way in which LogMiner works that will be very important to bear in mind if this tool is to be used forensically.

The first anomaly is a misleading imprecision in the way that LogMiner reports the time of log entries when being used to monitor previous historical database actions. The second anomaly is that LogMiner completely loses fractional `TIMESTAMP` information when recovering data from the database. Both of these are serious when viewed in the context of using LogMiner as a forensics tool but the most serious problem is that there is no public knowledge, disclosure or discussion of these facts (as of 19.45 08.01.05). This paper is the first discussion of these two problems but I certainly do not wish them to take away from the overall conclusion that LogMiner has forensic value and I hope the content that follows can be of some contribution to the SANS Oracle Security training in the future.

---

National Software Reference Library contains 10 million hashes of often-used files.  
<http://www.nsrl.nist.gov/index/mfg.index.txt>

The core strategy of this testing procedure is to ascertain whether LogMiner can be used as a forensics tool by evaluating its ability to produce repeatable and verifiable results that are precise, accurate and should stand up in a court of law under cross examination. In order to do this I am going to test the ability of LogMiner to create an accurately precise timeline of database activity. “This is the bedrock of an investigation” page 31 Track 8 Course Materials) Then I will test the ability of LogMiner to recover data that has been recorded in the Oracle database.

This paper is different compared to many GCFA papers due to the fact that it concentrates on a database utility and not a mainstream forensics tool like an operating system file analyser/copier. The choice of LogMiner has been checked with GIAC and this represents an exciting new area of forensics investigation. The focus of this paper needs to have the added component of asking “Is this database utility appropriate for forensics at all” as well as asking the normal question of “how good a forensics tool is it”. Also being a database tool there are different concepts involved.

Given that the practical part of the paper will be specialised on database technology, it would be useful to summarise the core database and DBA concepts to aid understanding of the LogMiner analysis. The following two sections are a simplified summary to bring a non Oracle DBA up to speed. If the reader has a reasonable knowledge of Oracle databases and security then the next two sections (3+4) can be bypassed.

### 3 Summary of supporting Oracle DBA concepts

Very simply put, Oracle is a Relational Database Management System (RDMBS), which is software that records data so that it can be queried using the Structured Query Language (SQL). SQL is the original idea of Dr Codd, winner of the Turing award 1981. Thanks to Dr Codd we can use SQL to say things like “find me all the instances of an employee in sales dept who earns less than 30000 pounds” as below.

Select \* from employee where dept='Sales' and salary <30000;

Employee is the table below which is a group of category columns and each employee in the table has their own row.

EmpID	EmpName	EmpDept	EmpSal
1 →	John smith	Sales	20000
2	Fred Jones	Sales	30000
3	Bill Bloggs	Research	40000

**Table 1 A relational database table**

SQL is composed of Data Manipulation Language (DML) and Data Definition Language (DDL). DML includes statements like “select \* from table\_name where table\_id=x” like above. Also insert, delete and generally commands that apply mainly to rows.

DDL statements affect the structure of the database and whole tables which includes statements like create, drop, grant, alter and rename which are used less frequently and usually by a privileged user such as the DBA (database administrator ) rather than a unprivileged user.<sup>2</sup>

We have looked at the logical structure of a database, which will be used by developers and users alike. However the physical structure is the realm of an Oracle

<sup>2</sup> LogMiner since 9i will record both types of statements.

DBA. The database is physically a software program that runs on a computer that allows information to be stored persistently. In other words if the computer is switched off it is not lost as it is still stored on the hard drive of the computer. The physical files on the hard drive can be cached by RAM memory and stored there for faster response times.

The DBA has many duties, which mainly include the following.

1. Installation and building the database
2. Configuring the Oracle network environment
3. Managing the Oracle instance
4. Managing the database storage structures
5. Administering users and security
6. Managing schema objects
7. Performing backup and recovery
8. Monitoring and tuning the database

The database structure has two dimensions. The first is its logical structure and the second its physical structure. The logical structure divides as follows from the larger containing database unit to the smaller contained units of logical structure.

1. Database=the whole collection of tables, relationships and metadata
2. Tablespace=where the information that makes up a table is stored
3. Segment=Contains the actual data, indexing and rollback information
4. Extent=subunit of segment
5. Block=subunit of extent

The smallest unit is the block and all these logical structures are described in the schema for the database. The physical storage structure consists of

1. Data files = database data stored as a group of files accessible through OS
2. Redo log files = record database activity and can be used for recovery  
(Log Miner reads from the redo log files)
3. Control files=contains information about the physical structure of the db

Metadata is data that pertains to data. For example a piece of metadata could be *"there are 5 tables in my database"*. LogMiner is a tool for reading historical metadata. *"There were 5 tables in my database"*. The metadata it reads is from the redo log files, which record all, changes to the database so that those changes can be undone or audited afterwards. The concept of Metadata pertains to lower level hard drive data with the concept of the metadata layer (SANS Track 8 Course materials book one page 80).

Memory used by Oracle server is split into the System Global Area (SGA) and the Program Global Area (PGA). The PGA is separate. The SGA is shared and contains items such as the data dictionary cache that contains data about the database structure like a complete list of all the tables in it for instance. The data dictionary is needed by LogMiner in conjunction with the redo logs to put the historical information together.

There are a number of background processes that will run once an Oracle instance is started. These include the Database Writer process and System Monitor process, but the one that we are particularly interested in is the Log Writer process or LGWR. It is the LGWR process that writes the redo logs that LogMiner takes as input. The redo logs are a binary encoded file that represents a record of all the changes that have been made to the database, whom made the changes and when. They can be used to recover the database to a point in time and to inspect what actions have taken place.

Of utmost importance to this paper is the concept of a database transaction. Each user of the database has a localised view of the central database so that users can make changes without affecting others until those changes are known to be correct. Users insert statements only affect this localised copy until the user decides to "COMMIT" the information. At this stage the changes are made to the actual database itself and other users queries on that database will now reflect this change. However if prior to committing the data the original user had decided that the insert statement was not correct they could have issued a "ROLLBACK" statement, which would erase the potential changes thereby stopping the central database from reflecting the changes that were subject to the "ROLLBACK". So changes on a central database tend to happen in a block when the "COMMIT" is issued. Each "COMMIT" is given a unique System Change Number, sometimes called a System Commit Number. This number is ever increasing and therefore related to time but does not have an exact linear relationship with time as we shall see in the testing phase.

We have briefly summarised the main mechanics that form the workings of the database "machine". The point of this machine is that it allows the users to manipulate data in a flexible way using SQL queries but only according to the privileges assigned by the DBA. Maintaining the hierarchy of privileges between the database users leads the thread of this paper to the general subject of Oracle security. Again if the reader is well read on the subject of Oracle security then this next section can be bypassed.

## 4 Oracle Security

Oracle security is a vast subject and this paper is only going to give a whirlwind summary in order to bring the uninitiated up to speed so that the LogMiner section will make more sense. A good reference for Oracle security is the Oracle Security Handbook by Theriault and Newman (Theriault 2001). Though four years old now it is still one of the best all round books on the subject. Another important read is the SANS Step-By-Step guide for Oracle Security (Finnigan 2004). The guide is not a textbook of theory but more of a set of practical guidelines for an already knowledgeable Oracle Administrator. As such it is a valuable read. It has steps that should be taken in order to secure a database installation. These steps form an industry accepted standard. The guide does not attempt to explain the concepts behind the steps in depth and so should be implemented by an expert who is also privy to the individual circumstances of the actual implementation.

By looking at the contents of these two books we can see an overview of Oracle security. As usual they start with the actual policy that is being implemented. Who can do what, measuring relative risks, threats and vulnerabilities and identifying the value of assets to be protected. This is standard for the planning stage of most security projects. At the more technical implementation level we can divide the subject area into these main categories.

1. Operating system security
  2. Database security
  3. Network security
  4. Application security
1. Operating system security is mainly concerned with gaining the correct users and permissions on the files that make up the database on the OS itself. Using a defence in depth philosophy we can use the OS security model to act as a barrier

to an attack. This is vital as Oracle authentication is often carried out at the OS layer giving privileged access to the database through their operating system credentials. Since the OS would usually be UNIX, the permission structure may not be fine grained to manage the intricacies of end user management hence the sophisticated system of privilege handled within the database itself.

2. Database privileges are largely about users, passwords and roles. Privileges should be assigned by adding a user to a role so that they can be managed in a group like fashion, similar to NT. Access can then be given either to an individual user or to the role they belong to. Of course when we talk about users this can also refer to an application that runs as a user.
3. Network security encompasses authentication, encryption, network integrity and the ability to secure network-accessed applications. Given that Oracle databases use port redirection to assign the initiating port 1521 to a different port above 1024, managing a firewall through which Oracle must communicate has caused issues generally. These issues have partly been cured by preventing port redirection or limiting access to Oracle through an application proxy.
4. Application security is possibly the most active area of Oracle security research with many bugs being found in the application server by companies such as NGS and Pentest Ltd for instance.

As a forensics tool LogMiner would most commonly be used after an incident has occurred during the second “Identification Phase” of the six-stage Incident Handling process (SANS Track 4 notes 2003). At this point the Handler knows something has happened but needs to identify what the problem is. A false alarm that the system has been hacked should not be raised. Instead a Forensics Analyst can be called in to methodically ascertain all the facts using the tools of the trade. These tools usually include the Coroners Toolkit, Sleuth Kit, Autopsy, netcat, dd, chrootkit and are well known to the other attendees of Track 8. If the Incident was on an Oracle database then LogMiner would be a candidate tool to use in order to trace back the order of events. At this point we will zoom in to the subject of this paper, which is the LogMiner tool and its suitability for the discipline of forensics.

## 5 Scope LogMiner testing

LogMiner is a utility that can be used to analyse the redo log files that are created by an Oracle database. The redo logs contain a record of the changes made to the database in a form that can be used to either recover corrupted data or monitor past actions (paraphrased from Thomas 2002).

Every transaction that occurs on the database will be recorded to the redo logs so that they can be redone if the database (DB) is corrupted, or undone if the DBA wishes the DB to go back to a previous state. LogMiner allows us to read the files that contain this data so that we can see what changes have been made at what time and by whom.

Another method of monitoring the security of a database is the inbuilt auditing functions of the Oracle database. Auditing is well developed and controlled by the aptly named AUDIT command, which can be narrowed down to a particular SQL statement, user, privilege or object. An example that audits all failed logins is below.

```
SQL> AUDIT SESSION WHENEVER NOT SUCCESSFUL
```

But what happens if the auditing information is lost through a database corruption either accidentally or on purpose? What happens if auditing is turned off because it is slowing down the database? This is where the redo logs can be a lifesaver as will be described in the following sections.

As well as Auditing there is another Oracle software product included with the RDBMS that has a role similar to that of LogMiner. This product is called Flashback which was introduced in 9i and improved substantially in 10g. I am not going to cover Flashback in detail in this paper due to the limitations that I have read about at the following URLs.

[http://databasejournal.com/features/oracle/article.php/10893\\_3446681\\_2](http://databasejournal.com/features/oracle/article.php/10893_3446681_2)  
[http://www.oracle.com/technology/deploy/availability/htdocs/Flashback\\_Overview.htm](http://www.oracle.com/technology/deploy/availability/htdocs/Flashback_Overview.htm)  
<http://www.dbazine.com/liu6.shtml>  
[http://www.databasejournal.com/features/oracle/article.php/10893\\_3446681\\_2](http://www.databasejournal.com/features/oracle/article.php/10893_3446681_2)  
<http://www.orafaq.com/articles/archives/000038.htm>

It should be noted that having read all these links I now know that Flashback is designed as an easy, user-friendly way of recovering data in the recent past. One of the features of Flashback is called the “recycle bin” which explains perfectly the reasoning behind its design. Flashback is not designed to go back far in time to high levels of accuracy as shown in this article below. <http://www.oracle-base.com/articles/10g/Flashback10g.php>.

This article specifies a three second error of time whereas the next URL article, directly from Oracle states a possible +/- 5 minute error so this puts it in the less effective forensic tools category and will not feature in our test.

[http://asktom.oracle.com/pls/ask/f?p=4950:8::::F4950\\_P8\\_DISPLAYID:5968150395572](http://asktom.oracle.com/pls/ask/f?p=4950:8::::F4950_P8_DISPLAYID:5968150395572)

LogMiner can take the state of the database back a time span of years in a reasonably quick and easy fashion depending on how well the archived logs are organised of course. Perhaps more interestingly LogMiner can be used to analyse and read the logs from years ago without having to recreate the database. Note the database needs to be in ARCHIVE MODE in order to back up the redo logs so that they can be stored separately from the online redo logs (which are the current ones). See this URL for more information about Archived redo logs.

[http://download-west.oracle.com/docs/cd/B14117\\_01/server.101/b10739/archredo.htm#i1006971](http://download-west.oracle.com/docs/cd/B14117_01/server.101/b10739/archredo.htm#i1006971)

LogMiner allows a timeline of changes to the database to be analysed. The logs can also be used to make a copy of the database. These two concepts are familiar to the forensic analyst. Forming a timeline and making a copy of data are central to forensics theory and practice.

The GCFA practical requires that LogMiner be tested for how verifiable and repeatable its results are. At the centre of this analysis is the requirement to assess how useful LogMiner would be to produce evidence that would be admissible in a court situation which obviously involves verifying the accuracy and precision of its results are “as specified”.

The scope of the evaluation of LogMiner was initially going to be just two fold to evaluate how well LogMiner could make a forensic timeline and recover historical data that may have been deleted. However during the first tests I noticed that LogMiner did not handle TIMESTAMPS as precisely as first appeared. This led me to enter into a second round of testing which also showed that LogMiner did not recover data from the database as accurately as it appeared either. This led me into a third round of testing to identify whether the error lay with the data that was being written to the redo logs or with the LogMiner tool itself.

The flow of this paper should reflect the three stages I went through, which makes it difficult to match exactly to the grading structure outlined in the assignment. I intend to cycle three times through the “Description of Procedures”, Criteria for Approval, Results and Analysis phases, once for each of the three testing phases so the flow makes sense.

These are the three main test phases.

1. General forensic capability:

Can LogMiner create a forensic timeline and reproduce data?

2. Verify Level of Precision:

Investigation of the TIMESTAMP imprecision discovered in test 1.

3. Identify source of the imprecision:

Is the imprecision the fault of LogMiner or the redo logs it reads from?

The overall strategy is underpinned by the need to make all tests reproducible and verifiable with the goal to give an evaluation of LogMiners potential use in a Court of Law.

The test methodology I will use is scientific and largely based on the recommendations of the SANS Track 8 course which introduced to me what is the best methodology I have been able to find for forensic tool testing. <http://www.cfft.nist.gov/Test%20Methodology%207.doc> (NIST 2001). The NIST methodology seems to back up everything I learnt in the SANS forensics training and stresses using repeatable and verifiable testing of accuracy and precision. Precision being the unit of measurement e.g. to the millimetre or the centimetre. Accuracy being the level of error in the measurements made by the tool. For instance the precision of a tool is to the nearest millimetre this is the finest it can measure whereas the level of accuracy may be to the nearest +/- 2 millimetres. The testing of LogMiner is influenced by the way in which forensics tool validations have been carried out by this organisation, for example the test carried on out on GNU's dd utility.

[http://www.cfft.nist.gov/setup\\_for\\_dd\\_tests.pdf](http://www.cfft.nist.gov/setup_for_dd_tests.pdf) (NIST 2002).

More information at the following URL's

<http://www.ojp.usdoj.gov/nij/sciencetech/cfft.htm>

<http://www.cfft.nist.gov/testdocs.html>

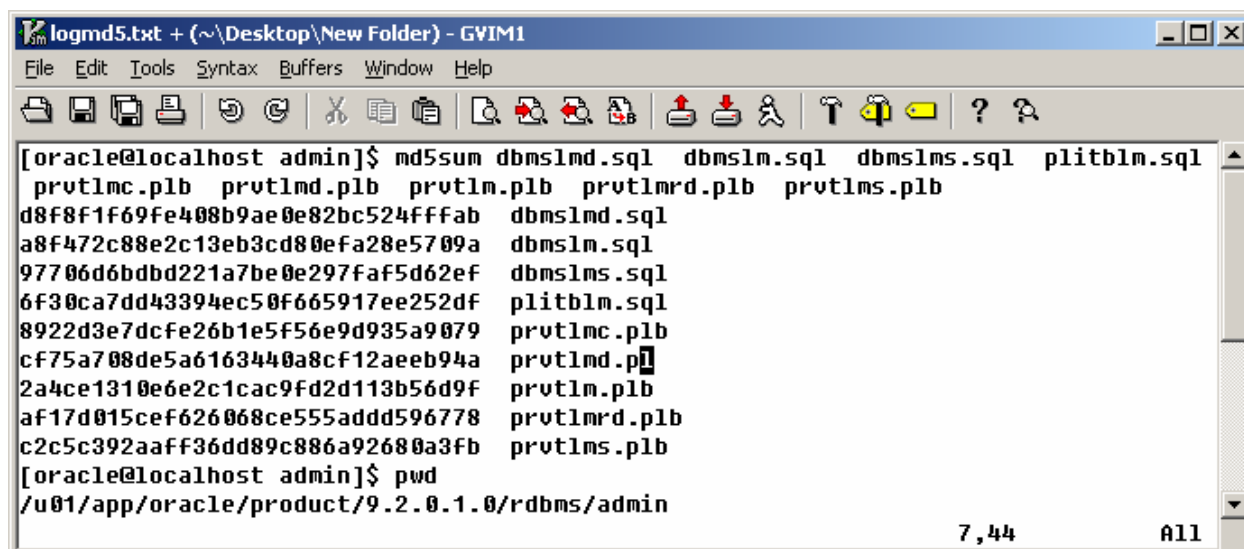
[http://216.239.59.104/search?q=cache:rBTH-x91EB0J:www.csa.syr.edu/James\\_Lyle.pdf+forensic+tool+evaluation&hl=en](http://216.239.59.104/search?q=cache:rBTH-x91EB0J:www.csa.syr.edu/James_Lyle.pdf+forensic+tool+evaluation&hl=en)

<http://www.aic.gov.au/conferences/evaluation/armstrong.html>

## 6 Tool Description

LogMiner is a tool for managing the online redo logs of an Oracle database. The redo logs contain the information necessary to reconstruct changes to the database tables and to analyse those changes in a flexible semi-automated fashion by using SQL statements to query those logs. The “COMMIT” statement that is used to save data in Oracle actually creates the redo logs, which will then allow a subsequent undo of that commit (which helps explain the delay in getting a “COMMIT” to happen).

The LogMiner tool consists of a number of packages, which are listed below with their respective MD5 checksums on Red Hat Linux Enterprise Edition V3 ES, update 3, running Oracle 9i Enterprise version. In order to give a cross platform view I have also tested LogMiner on Solaris 9, which proved to be so reliable that it is now my main machine.

A screenshot of a terminal window titled "logmd5.txt + (~\Desktop\New Folder) - GVIM1". The window shows the output of the command "md5sum dbmslmd.sql dbmslms.sql dbmslms.sql plitblm.sql prvtlmc.plb prvtlmd.plb prvtlms.plb prvtlms.plb". The output lists the MD5 hashes for each file. The terminal prompt is "[oracle@localhost admin]\$". The bottom right corner of the window shows "7,44" and "All".

```
[oracle@localhost admin]$ md5sum dbmslmd.sql dbmslms.sql dbmslms.sql plitblm.sql
prvtlmc.plb prvtlmd.plb prvtlms.plb prvtlms.plb
d8f8f1f69fe408b9ae0e82bc524ffffab dbmslmd.sql
a8f472c88e2c13eb3cd80efa28e5709a dbmslms.sql
97706d6bdbc221a7be0e297faf5d62ef dbmslms.sql
6f30ca7dd43394ec50f665917ee252df plitblm.sql
8922d3e7dcfe26b1e5f56e9d935a9079 prvtlmc.plb
cf75a708de5a6163440a8cf12aeeb94a prvtlmd.plb
2a4ce1310e6e2c1cac9fd2d113b56d9f prvtlms.plb
af17d015cef626068ce555add596778 prvtlms.plb
c2c5c392aaff36dd89c886a92680a3fb prvtlms.plb
[oracle@localhost admin]$ pwd
/u01/app/oracle/product/9.2.0.1.0/rdbms/admin
```

Figure 1 MD5 Hash Checksums of the files that make up LogMiner on RedHat

The .sql files are text files containing SQL code and the .plb are PL code, which is Oracle's programming language based on ADA. In this case Oracle has chosen to close the source of the PL/SQL code in the .plb files, so that the inner workings cannot be read. This is a pity as there are some changes I wanted to make which I will discuss later regarding the way in which LogMiner handles time. More information on how to protect one's PL source code at this URL <http://www.oraFAQ.com/faqplsql.htm#WRAP> (Naudé 2003).

The fact that LogMiner is made using these files means that a forensic analyst could supply their own version of these files on a CD or simply compare the checksums to verify that they are correct. However the tool cannot be compiled in a completely static way. The .plb files are given already as they are by Oracle. One could bring along a copy that was specific to that OS but one could not statically compile the whole tool separately oneself due to not having the source code. I am in the process of sending in a checksum to NIST so that LogMiner can be included in this list <http://www.nsl.nist.gov/index/mfg.index.txt>. And perhaps an ISO of forensically sound binaries. [ftp://ftp.nist.gov/pub/itl/div897/nsrl/ver\\_2\\_0/nsrl\\_2\\_0.iso](ftp://ftp.nist.gov/pub/itl/div897/nsrl/ver_2_0/nsrl_2_0.iso). It would be possible to add the LogMiner files to an archive of this type specialised for Oracle Forensics but it would still be dependent on the scripting engine and software built into the Oracle database for its running so it is not as independent as a statically linked binary.

Below is a link that summarises the programmatic interface of LogMiner.

[http://www.leidenuniv.nl/awcourse/oracle/appdev.920/a96612/d\\_logmn3.htm#77097](http://www.leidenuniv.nl/awcourse/oracle/appdev.920/a96612/d_logmn3.htm#77097)

Log Miner does not have a version number as such, as the files are included with the installation of Oracle (Enterprise Edition). However the 2 open sourced files of the LogMiner package (.sql) in the screen shot above have dates and versions in them along with authors names which I have provided a screenshot of below. The first screenshot is of the Solaris 9 version of Oracle's LogMiner. The second is of Red Hat Enterprise ES Version 3 update 3 running 9.2.0.4.0 build of Oracle's 9i database. These two OS's are the benchmark OS's that I have used for all testing in this paper. Documentation has been done using Office 2003 on a separate Windows 2003 server and Open Office 1.1.4 on Solaris 9.

```

rem
rem $Header: dbmslms.sql 17-oct-2003.10:53:32 ajadams Exp $
rem
rem dbmslmi.sql
rem
rem Copyright (c) 2000, 2003, Oracle Corporation. All rights reserved.
rem
rem NAME
rem   dbmslmi.sql - logmnr_session package description
rem
rem DESCRIPTION
rem   <short description of component this file declares/defines>
rem
rem NOTES
rem   <other useful comments, qualifications, etc.>
rem
rem MODIFIED   (MM/DD/YY)
rem   ajadams   10/17/03 - signature change for mine functions
rem   ajadams   01/08/03 - clean up purge routines
rem   abrown    11/20/02 - Add session_name to clone_context
rem   ajadams   10/07/02 - remove Adhoc CDC support - separate_undo_redo
rem   jkundu    10/11/02 - purge_session_to_scn param name change
rem   jkundu    08/19/02 - change signature of krpics
rem   mtao      09/30/02 - add purge_session_to_scn
rem   smangala   01/28/02 - add purge_session
rem   bgarin    01/08/02 - change session to private package
rem   abrown    09/17/01 - Add record global name option to krvcxs
rem   ajadams   09/20/01 - new dict option - NO_DDL_TRACKING
rem   smangala   08/30/01 - Add support for clone context.
rem   ajadams   08/21/01 - CREATE_FLAG_TRACK_AUDIT_INFO no longer needed
rem   ikundu    07/27/01 - filter none flag in create_session

```

Figure 2 Version of LogMiner for Solaris 10

```

rem
rem dbmslmi.sql
rem
rem Copyright (c) 2000, 2002, Oracle Corporation. All rights reserved.
rem
rem NAME
rem   dbmslmi.sql - logmnr_session package description
rem
rem DESCRIPTION
rem   <short description of component this file declares/defines>
rem
rem NOTES
rem   <other useful comments, qualifications, etc.>
rem
rem MODIFIED   (MM/DD/YY)
rem   smangala   01/28/02 - add purge_session
rem   bgarin    01/08/02 - change session to private package
rem   abrown    09/17/01 - Add record global name option to krvcxs
rem   ajadams   09/20/01 - new dict option - NO_DDL_TRACKING
rem   smangala   08/30/01 - Add support for clone context.
rem   ajadams   08/21/01 - CREATE_FLAG_TRACK_AUDIT_INFO no longer needed
rem   jkundu    07/27/01 - filter none flag in create_session
rem   qiwang     07/18/01 - add flag for audit info tracking
rem   smangala   03/19/01 - Add ALLOW_MULTIPLE constant
rem   qiwang     03/14/01 - Correct long names in previous check-in.
rem   jkundu    03/12/01 - adding new options to create_session
rem   ajadams   08/24/00 - add mine() functions
rem   ajadams   09/19/00 - add mine functions
rem   ajadams   09/18/00 - add mine functions
rem   ajadams   07/12/00 - move activate_session to debug package
rem   ajadams   07/10/00 - rename cancel_select to stop_session
rem   ajadams   06/13/00 - Creation

```

Figure 3 Version of LogMiner for RedHat Enterprise Version 3 ES

At this point it would be useful to summarise how LogMiner works. The process of log creation is as follows. The LGWR process in Oracle writes to the redo log buffer in the SGA (memory) which are then written to the hard drive as redo logs either when data

is committed or every three seconds. Of interest to a Forensic Analyst is the fact that the LGWR writes to the redo logs sequentially and so the actions that have been committed to a database can be chronologically re-applied in the case of a crash or accidental deletion.

<http://www.ic.leidenuniv.nl/awcourse/oracle/server.920/a96521/onlineredo.htm#3848>

Of course these redo log files can be used even if the database has not suffered an incident and the analyst just wishes to inspect what has occurred on the database previously. Most importantly using the redo logs the analyst can inspect the actions that have been carried out on a database over time in much the same way as a MAC timeline does with OS files (page 118 of the Forensics Day 1 course). MAC stands for Modify, Access and Change times and the supporting theory can be found on the day 2 Forensics Track book at page 167.

There is another process that is important to the creation of the redo logs and that is the Archiver process (ARC). The ARC process allows the archiving of redo log files to another location so that the database can be fully recovered. If the ARC process is not started then the redo log files would rewrite over themselves.

In order to start the ARC process LOG\_ARCHIVE\_START=TRUE must be written as a parameter in the init.ora file. The redo logs go from memory to hard drive as an online redo log file. When in archivelog mode the redo log files are archived optionally to a remote location. If the database in “no archivelog” mode they are overwritten. It is possible to multiplex the creation of online redo logs to different remote locations to provide backup in case of disk failure. The fact that the logs can be duplicated in this way helps to make LogMiners results verifiable. The results are certainly repeatable as one can simply re issue the command and wait to see if the same results occur. Perhaps more useful is the ability to repeat the process with a duplicate copy of the logs. This is both checking for repeatability and verifying the original result.

<http://www.ic.leidenuniv.nl/awcourse/oracle/server.920/a96521/archredo.htm>

When one wishes to specify an archive destinations one can use the following parameters. These are the initialization parameters to select between local and remote copying of the logs.

Method	Initialization Parameter	Host	Example
1	LOG_ARCHIVE_DEST_ <i>n</i>  where:  <i>n</i> is an integer from 1 to 10	Local or remote	LOG_ARCHIVE_DEST_1 = 'LOCATION=/disk1/arc'  LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1'
2	LOG_ARCHIVE_DEST and  LOG_ARCHIVE_DUPLEX_DEST	Local only	LOG_ARCHIVE_DEST = '/disk1/arc'  LOG_ARCHIVE_DUPLEX_DEST = ' /disk2/arc'

**Table 2 parameter commands for archiving from Oracles documentation**

The fact that you can have separate destinations is very good for business continuity as a secure remote store can be used to keep the redo logs in case of a local problem. One can do a manual checksum to see if they are both the same therefore verifying integrity. Or one can configure Oracle to use to verify the redo log files by setting the initialisation parameter LOG\_BLOCK\_CHECKSUM to TRUE. The default is

FALSE. Oracle computes a checksum for each log block written to the current log and uses it to detect corruption in a redo log block. See these URLs for more information

<http://www.cs.rose-hulman.edu/docs/oracle-817/server.817/a76956/online.htm#6163>

Installation notes for LogMiner.

<http://home.clara.net/dwotton/dba/logminer.htm>

The views on the redo log data that LogMiner gives you.

[http://www.adp-gmbh.ch/ora/misc/dynamic\\_performance\\_views.html](http://www.adp-gmbh.ch/ora/misc/dynamic_performance_views.html)

To recap, Oracle will create a redo log of the database which when processed by LogMiner will give the ability to recover the database to its original state and enable the DBA to query the database information as it was in a previous state. The Forensics Analyst will be able to trace back events with usernames and times tied to the actions. The investigator can use SQL to find the records they want in a fast and efficient manner using time or SCN (System Change Number).

The only system file that is accessed is the online data dictionary but this can optionally be exported as a separate file so LogMiner does not have to have any contact with the originating database.

LogMiner runs as a PL/SQL package and so depends on the Oracle RDBMS which accesses the UNIX operating system below it through the Oracle account in which it was installed. The binaries that execute exist as files on the OS but act within the Oracle RDBMS. Since LogMiner can and should be run on a separate server there is no contact with the originating, possibly, offending machine. LogMiner can be run on a completely air gapped machine on the logs of a separated machine as long as the Oracle versions match.

## 7 Test Apparatus

I have used two machines to do all of the testing on. These are the two notebooks I use professionally in my work so I know them to be reliable. The OS installs are completely fresh from binaries that have been cryptographically verified by their checksums.

Specification	Machine 1	Machine 2
Type of machine	Notebook	Notebook
Brand	Toshiba Satellite	Sony Viao
Model	PSM35e-00690-7v	PCG9526
Processor	PIII 1.8	PIII750
Memory	512	512
OS	Solaris 9 x86 -117172-07	RedHat Enterprise 3 v3
Shell	Korn Shell 93	Bash
Patch level	No patches	No patches
Service Packs	None	None
Oracle Installation	10g Enterprise	9i Enterprise
Oracle patches	None	3006854 for Linux
Networking	None	None
Power	AC mains	AC mains

Table 3 Hardware and software specifications

Solaris version identified with

```
# uname -a
```

```
SunOS solaris 5.9 Generic_117172-07 i86pc i386 i86pc
```

LogMiner has been developed and improved for 9i and 10g since its original debut on 8i. I have chosen to focus on its use with 10g as the most recent version running on the most likely OS, which is Solaris 9. I have run the same tests on the Red Hat Linux version of LogMiner to verify my test findings.

The original plan was to do the entire testing using LogMiner viewer, which is a Java based interface integrated with Oracles Enterprise Manager (OEM). LogMiner viewer gave problems on Red Hat 3 enterprise V3 – Machine 2.

Firstly it is more difficult to set up than the simple command line version. OEM appears to require the installation of another database called the repository. The repository is designed to hold data about the actual database. It is possible to save the metadata to the same instance if you only wish to have one db or only have disk space for one db. It is also possible to use LogMiner Viewer in standalone mode but I found it unreliable in its use. The main problems I have found with the LogMiner Viewer are as follows.

1. It locks up regularly. Over the period of four hours it locked up about 5 times.
2. Viewer disappeared. It would just completely disappear from view. At first I thought it was a mistake in the way I was using it but this problem repeated about three times over the four hours.
3. Data not selecting correctly. I tried to do different queries but it permanently locked me into selecting only TIMESTAMP based information.

Rather than reinstall and look for patches for this I decided to go to the command line version. Reasons for this were that my previous experience of Java based GUI's on Linux is that they can quite unreliable even when they come with their own JVM like the Oracle installation disks. Also the command line version is so easy to use and was very reliable. Since it only relied on using the shell and SQL\*Plus there is a lot less to go wrong so I decided to put the LogMiner viewer to one side.

Having said this there are some benefits to getting the GUI version running as the help pages are excellent in helping to understand the tool. The best part of the LogMiner viewer is the ability to convert the GUI commands into SQL commands that could be run at the command line. This mode is similar to Ethereal and NMAP in that it helps the user to understand how the queries are built and enables the user to wean themselves off the GUI and onto the command line. I would at this stage recommend the GUI as a learning tool not as a forensics analysis front end for LogMiner.

There is specific detail on using LogMiner Viewer (on Windows) at this URL.  
<http://www.oracle.com/technology/products/oracle9i/htdocs/9iobe/OBE9i-Public/obe-ha/html/logminer/logminer.htm>

Therefore the interface for LogMiner has been the BASH shell on Linux that is standard with the install /bin/bash. On Solaris it was Korn Shell /usr/bin/ksh again standard with the installation of Solaris 9.

The Oracle client was the SQL\*Plus interface, again standard with the Oracle server installation.

## 8 Environmental conditions

The testing was completed on the machines 1 and 2 in my test lab in an isolated fashion with both the client and the server on the same machine. No one had access to the machines except myself. There was literally no network interaction whatsoever due to the fact that both machines are completely disconnected from any wired or wireless networking. The only transfer of files was done by a USB key between the machines and the USB key had been formatted before use. There are no outside forces that could have enacted on either of these machines as they were locked in the test room during the entire test period. Temperature, humidity, atmospheric pressure are all steady as heating is steady at all hours and there are no sources of heavy magnetism nearby so I am confident that these results cannot be adversely affected by any environmental factors.

## 9 Testing overview

It is worth reiterating that due to my discovery of the TIMESTAMP anomalies there were two further subsequent phases of testing additional to the original which made three in all. These three phases occurred one after another. In order to comply to the GCFA structure and making grading a less ambiguous task I considered grouping the three phases together synchronously as though they happened at the same time for reporting and documentation purposes. However this is not how it happened and feels quite unnatural to report it in this way. Therefore, I will first apologise to the grader and restate that the format this report follows the actual sequence of testing events which is as follows.

### Test 1 – General forensic applicability

- a) Check that LogMiner can create a timeline useful for forensics
- b) Check that LogMiner can recover data in a way useful for forensics

### Test 2 – Accuracy and Precision of the TIMESTAMP

- a) Investigate accuracy/precision of the reporting of TIMESTAMP in LogMiner
- b) Investigate accuracy/precision of the data recovery of TIMESTAMP in LogMiner.

### Test 3 – The source of the imprecision

Identify the source of the inaccuracy/imprecision. Was it the redo logs or was it LogMiner?

Each round of testing has further sub procedures, sub criteria for approval, results and analysis. Therefore I would ask that in grading this paper that the Three tests are considered together. The alternative is either not to report the subsequent tests or report them as though they were done at the same time which does not make as much sense. My overall goal is to evaluate the tool for its forensic ability.

Each of these tests will have to be repeatable and verifiable so that any forensic scientist around the world can duplicate the same results that I have attained.

## 10 Description of the Procedures - preparation

The aim is to produce a time line like a MACTime line that we created using the MACTime Perl script in the SANS forensics course. This will not need the collection of the OS files into a body file. Instead it will require an SQL query that will output to a text file. The end result will hopefully be familiar and possibly able to integrate with a standard .mac timeline file. Then we will attempt to reproduce a table from the redo logs and compare it to the original table.

As part of the results I am going to use an MD5 hash tool called eXpress CheckSum by Irnis Haliullin at <http://www.iris.net/?prod=xcsc&ver=1.0.0>. As part of the preparation I had to test to make sure this tool did correctly calculate MD5 hashes in a repeatable way. I verified hashes created on RH Linux Machine 2 using the inbuilt md5sum command line bash tool which gave consistent hash values. This gave me confidence to use the Windows eXpress tool.

I prepared the documentation processes by checking that the Open Office 1.1.4 would transfer documents to Office 2000 and 2003 as I would be documenting on Solaris and Windows. As long as I saved in Open Office to Word .doc format there were no major compatibility issues.

Testing was done on Solaris Machine 1 and then duplicated on Red hat Machine 2.

There were some stages of preparation needed before doing any testing as described in the next three sections.

### 10.1.1 Logging on and starting Oracle

1. We need to log onto the Solaris machine. Note that if one wants X applications to run with Oracle one should log on as Oracle initially rather than SU to Oracle.

2. Start Korn shell and issue the sqlplus command.

Then enter "sys as sysdba" and password to gain a privileged user on the database.

3. Then issued the "startup" command to start Oracle database.

On Solaris the database is automatically opened.

4. On Linux though one also has to issue the "Alter database open" command

The next procedure is to set up LogMiner and make sure it works before testing proper.

### 10.1.2 Setting up LogMiner

The basic order of events to run LogMiner, in simple terms, are as follows

1 Switch on supplemental logging (optional)

2 Specify the redo log files and the path to them

3 Allocate a Dictionary

4 Start LogMiner

5 Read the data about past state and recover the database

6 Stop LogMiner

1. Supplemental logging should be enabled in order to use LogMiner which can be done with the following command.

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;  
And then check it has worked with the following query.  
SQL> SELECT SUPPLEMENTAL_LOG_DATA_MIN FROM V$DATABASE;  
SUPPLEME  
-----  
YES
```

## 2. Specify the location of the online redo logs.

```
SQL> EXECUTE DBMS_LOGMNR.ADD_LOGFILE(LOGFILENAME =>
'/export/home/u01/app/oracle/oradata/sales/redo01.log', OPTIONS =>
DBMS_LOGMNR.NEW);
```

PL/SQL procedure successfully completed.

```
SQL> EXECUTE DBMS_LOGMNR.ADD_LOGFILE(LOGFILENAME =>
'/export/home/u01/app/oracle/oradata/sales/redo02.log', OPTIONS =>
DBMS_LOGMNR.ADDFILE);
```

PL/SQL procedure successfully completed.

```
SQL> EXECUTE DBMS_LOGMNR.ADD_LOGFILE(LOGFILENAME =>
'/export/home/u01/app/oracle/oradata/sales/redo03.log', OPTIONS =>
DBMS_LOGMNR.ADDFILE);
```

PL/SQL procedure successfully completed.

I issued each of these three commands on a single line as I did not have time to experiment with carriage returns, but the character “–” will allow a new line to extend a command over multiple lines.

Then we need the command to tell it where the dictionary will be taken from the online database directly.

## 3. Start LogMiner with the online data dictionary catalogue.

```
SQL> EXECUTE DBMS_LOGMNR.START_LOGMNR(OPTIONS =>
DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG);
```

PL/SQL procedure successfully completed

This means that in this case LogMiner will only work correctly when the database is started and open as we are using the source DB's online dictionary. The problem with using the online catalogue is that only the current version of the db can be queried as the old schemas are lost. Therefore it is advisable if using LogMiner in production circumstances to back up the versions of the schema either in an accompanying flat text file or in the redo logs themselves. LogMiner is now started and ready to query.

## 4. Example query run upon the LogMiner view - v\$logmnr\_contents

```
SQL> select scn,timestamp,username,table_name,operation from v$logmnr_contents;
509304 04-JAN-2005 14:00:57 WRH$_SQLBIND INSERT
509304 04-JAN-2005 14:00:57 WRH$_SQLBIND UPDATE
509304 04-JAN-2005 14:00:57 INTERNAL
509304 04-JAN-2005 14:00:57 WRH$_SQLBIND INSERT
509304 04-JAN-2005 14:00:57 WRH$_SQLBIND UPDATE
509304 04-JAN-2005 14:00:57 INTERNAL
509304 04-JAN-2005 14:00:57 WRH$_SQLBIND INSERT
509304 04-JAN-2005 14:00:57 WRH$_SQLBIND UPDATE
```

This is an example query on the v\$logmnr\_contents view which represents all the data LogMiner is able to extract from the redo logs.

## 5. End the LogMiner session

```
SQL> EXECUTE DBMS_LOGMNR.END_LOGMNR;
```

One can now quit the shell. This has gone through a LogMiner testing session to make sure it is working well before the actual test. It did this first time no problems.

This is the end of the set-up phase for LogMiner and we are now ready to test how well it performs the duties of a forensics analysis tool. Please note that I experimented with scripting the LogMiner set up procedure but when done as a single script it crashed LogMiner. Therefore I would issue these commands manually.

The documentation has been done with a combination of Solaris Open Office 1.4, Microsoft Office 2003, 2000, Vi, Gvim and screen shots.

Oracle has automatic checksum procedures built into the software. As long as skip corruption is not selected then a corruption will interrupt the process automatically.

### 10.1.3 Setting up SQL\* Plus output

Oracles guidelines for setting up LogMiner recommend the following setting of the session for the DATE type.

```
SOL> ALTER SESSION SET NLS DATE FORMAT='DD-MON-YYYY HH24:MI:SS';
```

This will give us as presentation of the date with hours, minutes and seconds.

### 10.1.4 Archiving the redo logs

An important point when using LogMiner is that the online redo logs are constantly being written to and so changing. There are three online redo logs and my experience using them they completely overwrite in a period of 24 hours. They can be very big even with little data being changed due to the system information that is recorded by default. Eriks post at ASKTOM points this out

[http://asktom.oracle.com/pls/ask/f?p=4950:8:1320218815388063256::NO::F4950\\_P8\\_DISPLAYID,F4950\\_P8\\_CRITERIA:4635285328580](http://asktom.oracle.com/pls/ask/f?p=4950:8:1320218815388063256::NO::F4950_P8_DISPLAYID,F4950_P8_CRITERIA:4635285328580)

However we are not all running a SAN yet and as long as we know which redo log to go to then I have not experience the delay that Erik mentions.

What I have experience is losing data because it gets overwritten therefore Archive logging is preferable. If archive logging is not set then make a copy of the redo log. This can then be hashed and passed into a chain of custody.

### 10.1.5 Chain of custody

Chain of custody is verifiable line of responsibility for a piece of evidence. The redo log that has been backed up should be hashed to and the hash compared to the original. Then it should be taken off the machine by neutral media preferable read only like a CDROM that has been finalised. Then documented. This will include signing for and dating the evidence as intact and in the possession of Mr or Mrs X. There are forms and FAQs on how to interface with Law enforcement and all evidence has to treated as though it may end up in court.

[http://www.sans.org/score/faq/law\\_enf\\_faq/](http://www.sans.org/score/faq/law_enf_faq/). The CDROM should be in an airtight bag and kept in an environmentally neutral state i.e. away from sunlight in the case of a CDROM. A photograph can be taken with a digital camera of the signed bag for future reference. This will be crucial when it comes to section 14 court presentation and 16 the cross examination.

# 11 Test 1 - General forensic capability

This is the basic level of testing to make sure that LogMiner can create a forensic timeline and can recover data in a repeatably reliable manner.

## 11.1 Criteria for approval and tool test execution

If LogMiner is to be considered a forensically sound tool to use it should be able to report events in the order they occurred and relate these to standard time. LogMiner should be able to do this repeatedly and also should have the ability to verify the timeline. For the purposes of testing I have defined “*repeatedly*” as 20 iterations. The reason for this is that if all 20 iterations are correct then this gives a less than 5% uncertainty value. Greater than 95% certainty is a statistical guideline often used to act as line dividing what we can be reasonably sure of. LogMiner should be usable and not too slow in producing its results. LogMiner should be able to preserve the precision of data passed to it so that it can recover the database as it was.

The timeline is in chronological order and produced the same results 20 times as verified by a checksum of the spooled off file. The file is too long to include here but I have included the timeline in the Appendix and the next section shows a section of it.

A good question to think about whilst doing this is “Do the redo log files change when LogMiner reads them?” The answer to this is “NO” as Oracle is automatically checking the checksums of the logs as described previously.

For recovery, LogMiner should be able to recover the database table as it was originally repeatedly. This recovery should be verifiably accurate. Then I will delete and recover the table repeatedly to see if it can reliably recover the table. It is worth making the observation that I thoroughly expect the database to be able to carry out both of these tests reliably as databases are very good at repeating operations in the same way. The question that this paper moves onto later about levels of precision and suitability for Forensics maybe more pertinent. The tool is executed by querying the database view it produces using SQL. A view is a like a normal table except that it is virtually created from something else behind it. The source could be another table but in this case it is the redo log file. This is great as we have a normal Oracle db interface to flat file logs and can use all the benefits of SQL on those logs to query any of the information with infinite permutations.

### 11.1.1 Time line test execution

The timeline procedure starts with spool command to output to a text file, then has the SQL command that will create the timeline from the LogMiner table view called v\$logmnr\_contents It finally ends the creation of the text file with spool off.

```
SQL>spool mactimeline.mac
SQL> select scn, timestamp, username, table_name, operation from v$logmnr_contents;
SQL>spool off
```

To test for repeatability and to verify the result I will repeat this procedure 20 times and take an MD5 has of each timeline so that they can be compared to make sure they are the same. No system files are used during this test.

### 11.1.2 Database recovery execution

The database recovery procedure is very simple as one simply selects the sql\_redo from the v\$logmnr\_contents for the transaction that one wishes to recover. For example below.

```
SQL> select sql_redo from v$logmnr_contents where scn=689997;
```

It is most accurate and recommended to use the SCN number to reference each transaction. Then one will need to compare the original row with the newly created one using the LogMiner redo\_sql. (No system files are used during this testing).

## 11.2 Data and results

The full time line that was created is in the Appendix and does not count toward my page count. For GCFA it can be handed in as a separate document. This is a representative segment below using the SQL query given in the procedures section

541408	04-JAN-2005	20:02:38	SYS		COMMIT
541424	04-JAN-2005	20:03:29	SYS		START
541424	04-JAN-2005	20:03:29	SYS	JOB\$	UPDATE
541425	04-JAN-2005	20:03:29	SYS		COMMIT
541425	04-JAN-2005	20:03:29	SYSMAN		START
541425	04-JAN-2005	20:03:29	SYSMAN	MGMT_SYSTEM_PERFORMANCE_LOG	INSERT
541425	04-JAN-2005	20:03:29	SYSMAN		INTERNAL
541426	04-JAN-2005	20:03:29	SYSMAN		COMMIT
541426	04-JAN-2005	20:03:29	SYSMAN		START
541426	04-JAN-2005	20:03:29	SYSMAN	JOB\$	UPDATE
541426	04-JAN-2005	20:03:29	SYSMAN		INTERNAL

The timeline was created into a text file in much the same way as the MACtime Perl script from the SANS forensics course. The SCNs and times all incremented in order. The 20 repeats of the query all gave the same MACtime line as verified by the same md5 hash.

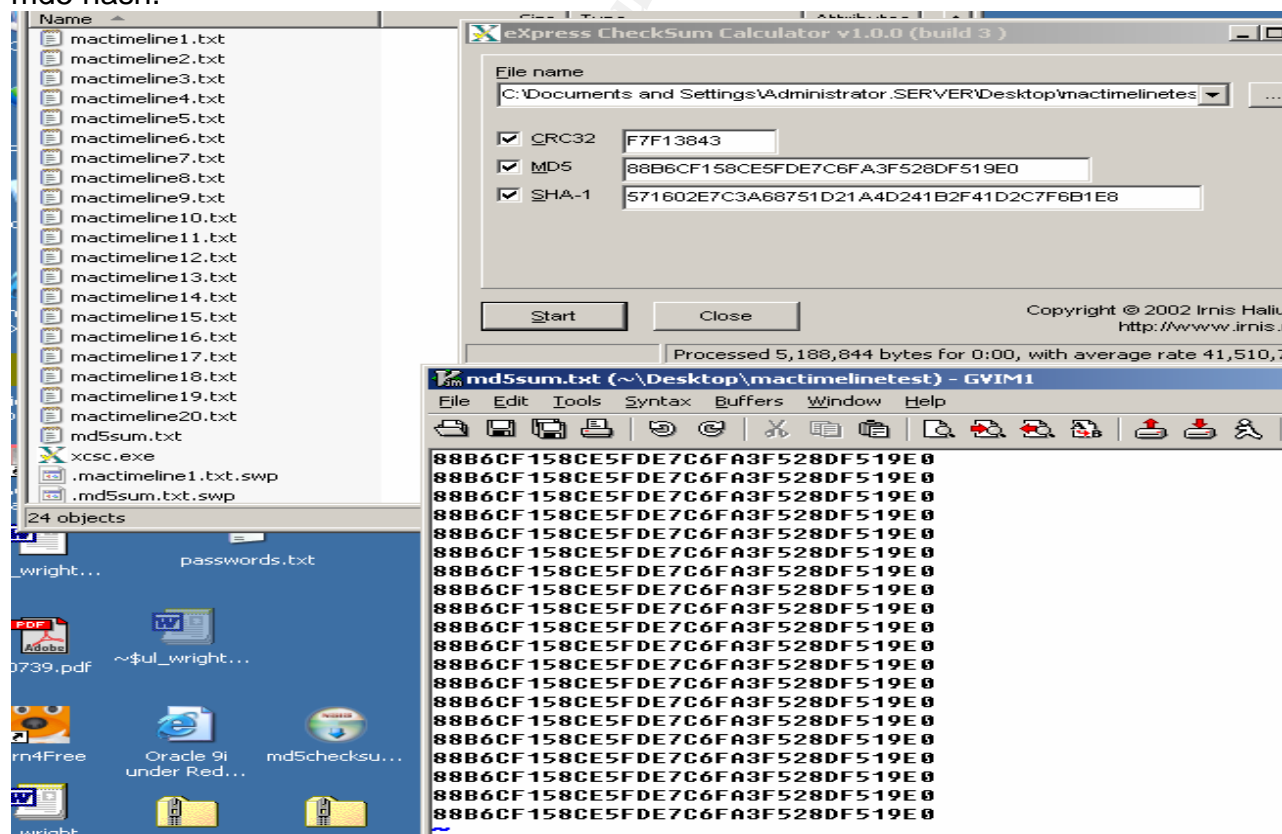


Figure 4 MAC timelines with their corresponding MD5 hash checksums

MD5Checksum comparison done on Windows 2003 Enterprise Server using CheckSum Calculator 1.0.0. <http://www.softpedia.com/progDownload/CheckSum-Calculator-Download-17463.html>  
 It was noticed that there were a lot of both repeated SCN number entries and entries with the same time. Also the timestamp column was only reporting to an accuracy of one second when a timestamp by default gives 6 decimal places of one second. This will be looked at in the analysis phase.

The data recovery part of the test went well too as can be seen by the code out put results below. I was able to insert a row, find the row SCN in the redo logs using the table name, then query the sql\_redo column with that SCN to give me the SQL necessary to redo that row which it did without fault.

```
SQL> insert into emp values (5432,'Stephen','admin',5555,899,40)
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	SAL	DEPTNO
7369	smith	clerk	7902	800	20
1234	john	clerk	8989	888	30
7369	smith	clerk	7902	800	20
7369	smith	clerk	7902	800	20
7369	smith	clerk	7902	800	20
7369	smith	clerk	7902	800	20
5432	Stephen	admin	5555	899	40
1111	paul	clerk	7902	800	20

8 rows selected.

```
SQL> select scn, operation from v$logmnr_contents where table_name='EMP'
```

SCN	OPERATION
689997	INSERT

```
SQL> select scn, timestamp, operation from v$logmnr_contents where table_name='EMP';
```

SCN	TIMESTAMP	OPERATION
689997	09-JAN-2005 00:00:19	INSERT

```
SQL> select sql_redo from v$logmnr_contents where scn=689997;
SQL_REDO
```

```
set transaction read write;
insert into "SYS"."EMP"("EMPNO","ENAME","JOB","MGR","SAL","DEPTNO") values
('5432','Stephen','admin','5555','899','40');
```

```
SQL> insert into "SYS"."EMP"("EMPNO","ENAME","JOB","MGR","SAL","DEPTNO") values
('5432','Stephen','admin','5555','899','40');
```

1 row created.

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	SAL	DEPTNO
7369	smith	clerk	7902	800	20
1234	john	clerk	8989	888	30
7369	smith	clerk	7902	800	20
7369	smith	clerk	7902	800	20
7369	smith	clerk	7902	800	20
7369	smith	clerk	7902	800	20
5432	Stephen	admin	5555	899	40
1111	paul	clerk	7902	800	20
5432	Stephen	admin	5555	899	40

9 rows selected.

I repeated this simple test 20 times and it worked perfectly each time. This was as I expected as this is exactly what databases are good at; persisting data in the same form. What may be more interesting is how accurately it does this with different datatypes as we shall see.

## 11.3 Analysis

The basic forensics usefulness has been shown in this first set of tests as LogMiner can produce a useful timeline and can also recover a row in a table from the redo logs. However there are some questions thrown up by the test.

1. Why do some of the time line entries have the same times and SCNs.
2. Why is time only shown precise to a single second when the **TIMESTAMP** datatype is accurate to 9 decimal places maximum and 6 by default.

The first question is easy to explain as the SCN and time entries are created at the point that the database entries are committed in the transaction. There may be many actions that are committed at the same time which will be indistinguishable from each other in terms of SCN number and time.

However the second question about the **TIMESTAMP** field is confusing as I know the **TIMESTAMP** is more accurate than a second which can be shown by this query. How a **TIMESTAMP** should look.

```
SQL> select systimestamp from dual;
04-JAN-2005 23:06:12.165048 +00:00
```

Oracle versions prior to 9i were only able to record time to the nearest second by using the **DATE** datatype. 9i and after can record times to the nearest nanosecond or 9 decimal places using the data type **TIMESTAMP**. The column in LogMiners view is called **TIMESTAMP** and so should show like the above **TIMESTAMP**.

How the **TIMESTAMP** looks in LogMiners reporting table.

```
SQL> Select TIMESTAMP from v$LOGMNR_CONTENTS where scn=598212;
TIMESTAMP
-----
05-JAN-2005 12:35:34
05-JAN-2005 12:35:34
05-JAN-2005 12:35:34
05-JAN-2005 12:35:34
05-JAN-2005 12:35:34
```

The lack of precision on the timestamp column brought up two questions. Firstly does LogMiner correctly report the time at which a record was made in the redo logs? This is a timeline question.

Secondly can LogMiner correctly recover **TIMESTAMPS** that were recorded in the original database so that the recovered database still has the same **TIMESTAMP** data in it. This is a data recovery question.

Both these questions necessitated a second round of testing to find out the answer to what was becoming an interesting investigation.

## 12 Test 2–TIMESTAMP accuracy of timeline and recovery

### 12.1 Timeline TIMESTAMP precision

The TIMESTAMP not showing decimal places for seconds could be because:

1. The TIMESTAMP format mask is only set to report seconds.
2. The TIMESTAMPS were all created exactly on the second for some reason.
3. They are not really TIMESTAMPS at all.

At this point it would be useful to do a “describe” of the v\$LOGMNR\_CONTENTS view which will tell us all the data that LogMiner records and the datatypes it uses to record this data. The query that follows gives the entire contents of the LogMiner view.

```
SQL> desc v$logmnr_contents;
Name                                         Null?    Type
-----
SCN                                           NUMBER
CSCN                                           NUMBER
TIMESTAMP                                    DATE
COMMIT_TIMESTAMP                           DATE
THREAD#                                       NUMBER
LOG_ID                                        NUMBER
XIDUSN                                         NUMBER
XIDSLT                                         NUMBER
XIDSQN                                         NUMBER
PXIDUSN                                        NUMBER
PXIDSLT                                        NUMBER
PXIDSQN                                        NUMBER
RBASQN                                         NUMBER
RBABLK                                         NUMBER
RBABYTE                                        NUMBER
UBAFIL                                         NUMBER
UBABLK                                         NUMBER
UBAREC                                         NUMBER
UBASQN                                         NUMBER
ABS_FILE#                                    NUMBER
REL_FILE#                                    NUMBER
DATA_BLK#                                    NUMBER
DATA_OBJ#                                    NUMBER
DATA_OBJD#                                   NUMBER
SEG_OWNER                                    VARCHAR2(32)
SEG_NAME                                     VARCHAR2(256)
TABLE_NAME                                  VARCHAR2(32)
SEG_TYPE                                     NUMBER
SEG_TYPE_NAME                              VARCHAR2(32)
TABLE_SPACE                                 VARCHAR2(32)
ROW_ID                                       VARCHAR2(18)
SESSION#                                    NUMBER
SERIAL#                                     NUMBER
USERNAME                                    VARCHAR2(30)
SESSION_INFO                                VARCHAR2(4000)
TX_NAME                                     VARCHAR2(256)
ROLLBACK                                    NUMBER
OPERATION                                   VARCHAR2(32)
OPERATION_CODE                             NUMBER
SQL_REDO                                    VARCHAR2(4000)
SQL_UNDO                                    VARCHAR2(4000)
RS_ID                                       VARCHAR2(32)
SEQUENCE#                                   NUMBER
SSN                                         NUMBER
CSF                                         NUMBER
INFO                                        VARCHAR2(32)
STATUS                                     NUMBER
REDO_VALUE                                 NUMBER
UNDO_VALUE                                 NUMBER
SQL_COLUMN_TYPE                            VARCHAR2(30)
SQL_COLUMN_NAME                            VARCHAR2(30)
REDO_LENGTH                                NUMBER
REDO_OFFSET                                NUMBER
UNDO_LENGTH                                NUMBER
UNDO_OFFSET                                NUMBER
DATA_OBJV#                                 NUMBER
SAFE_RESUME_SCN                             NUMBER
XID                                          RAW(8)
PXID                                         RAW(8)
AUDIT_SESSIONID                             NUMBER
```

This query is very informative as it tells us that the `TIMESTAMP` column is not a `TIMESTAMP`, it is in fact a `DATE`. A date can only be accurate to a single second so could never reproduce full `TIMESTAMP` information. The “`TIMESTAMP`” name of this column is certainly misleading and in a forensics context could cause misunderstanding and so I think this needs to be brought to the general attention of the Oracle forensics community.

If many transactions are committed in the same second then how can they be separated from one another? On a large, busy, mission critical enterprise database the time that events occur will be crucial. When analysing time forensically we have to think about how precise the time is and how accurate it is. Precision of course gives us a way to state how fine our measuring units are i.e. to the second or millisecond. Accuracy refers to how much error is experienced in a tools use. Accuracy can be difficult to measure but precision is part of the system design and needs to be stated correctly. The precision of LogMiners time scale is ambiguous due to the misnaming of the `TIMESTAMP` column.

From the tuition I received during the SANS Track 8 course I know that a precise timeline can be very important in Forensics. What is much more important than the level of precision is “*knowing*” correctly your level of precision. The level of precision that a tool advertises can give a false impression of the accuracy that can be achieved by using the tool. For instance if we gave a micrometer to some one to measure from here to the edge of the Universe this would be misleadingly precise as we cannot measure this distance that accurately. If one falsely believes ones level of accuracy to be more than it actually is then there can be capacity for mistakes. On this point I think we have hit a small problem with LogMiners forensic design. However an experienced DBA should notice the difference, as I did, but I think it unwise to assume that a forensics investigator will also have good DBA skills. The anomaly is not a long-term problem as long as we now know that the precision of the LogMiner “Timestamp” is one second we will not make any false assumptions. It would be helpful for Oracle to either change the name of the column to `DATE` or to make the datatype `TIMESTAMP` as it says in the heading for the column. If this cannot be done then the discrepancy should be documented so that users do not make an easy mistake. I would have liked to edit the .plb files myself to change the `DATATYPE` manually in the PL code but it is closed source.

The reason for the discrepancy, I believe, is that LogMiner was first developed on Oracles 8i database, which was before the `TIMESTAMP` datatype was introduced in 9i. It appears that this discrepancy was not noticed when it has been upgraded over the years after the introduction of the `TIMESTAMP` datatype.

## 12.2 Integrity of `TIMESTAMP` recovery.

The other point regarding time that is again crucial in this evaluation of LogMiner is whether or not LogMiner can preserve data that is already in “real” Timestamp format in the original database (i.e. with ability to have decimal places). If LogMiner cannot preserve `TIMESTAMP` data that has decimal places during recovery this might be a serious problem. Data that is recorded to a decimal place would usually be done so for a reason. Subsequent rounding off could be disastrous. The data would be lost but also the Checksums of the data files would be different thus implying corruption/tampering when this was not the case. I devised a simple test to see if

LogMiner can preserve TIMESTAMP data. I will create a table with a TIMESTAMP then insert a TIMESTAMP into it and then redo the TIMESTAMP from the entry in the redo logs that will be gained via LogMiner to see if LogMiner can preserve the decimal places or not as the case may be. The test is shown on the next page using Korn shell on Solaris Machine 1. I repeated the test three times to give a scientific sample and it was the same in all three cases. I also rebooted the machine before doing the test to make sure that the OS was running smoothly.

```

Terminal
Window Edit Options Help

SQL> create table timestamptest(timestamp TIMESTAMP);
Table created.

SQL> alter session set NLS_TIMESTAMP_FORMAT='yyyy-mm-dd hh:mi:ssxff';
Session altered.

SQL> insert into timestamptest values (to_timestamp('2005-01-04 10:10:37.474839'));
1 row created.

SQL> select * from timestamptest;

TIMESTAMP
-----
2005-01-04 10:10:37.474839

SQL> commit
2 ;
Commit complete.

SQL> select scn, operation, timestamp, username from v$logmnr_contents where table_name='TIMESTAMPTEST';

      SCN OPERATION                TIMESTAMP                USERNAME
-----
  622059 DDL                      05-JAN-2005 14:35:39 SYS
  622104 DDL                      05-JAN-2005 14:37:26 SYS
  622121 DDL                      05-JAN-2005 14:37:34 SYS
  622175 DDL                      05-JAN-2005 14:39:02 SYS
  622193 DDL                      05-JAN-2005 14:39:15 SYS
  622199 INSERT                   05-JAN-2005 14:39:30 SYS

6 rows selected.

SQL> select sql_redo from v$logmnr_contents where scn=622199;

SQL_REDO
-----
set transaction read write;
insert into "SYS"."TIMESTAMPTEST"("TIMESTAMP") values (TO_TIMESTAMP('2005-01-04 10:10:37'));

SQL> insert into "SYS"."TIMESTAMPTEST"("TIMESTAMP") values (TO_TIMESTAMP('2005-01-04 10:10:37'));
1 row created.

SQL> select * from timestamptest;

TIMESTAMP
-----
2005-01-04 10:10:37.474839
2005-01-04 10:10:37.000000

```

As can be clearly seen above the fractional TIMESTAMP data from the original has been lost in the LogMiner recovered row. I repeated this test three times on Solaris Machine 1 and Red Hat Machine 2 and the results were consistent as expected.

## 12.3 Analysis of test 2

Now to talk you through the test in the Solaris screenshot previously to show what is happening in the test.

Below creates a table with a column that can record time values in the TIMESTAMP datatype to a maximum of 9 decimal places (a nanosecond) but by default to 6 decimal places.

```
SQL>create table timestamptest(timestamp TIMESTAMP);
```

Then we format the screen output to display this level of precision.

```
SQL >alter session set NLS_TIMESTAMP_FORMAT='yyyy-mm-dd hh:mi:ssxff';
```

Then we insert a record into the new table with a TIMESTAMP value.

```
SQL >insert into TIMESTAMPTEST values (to_timestamp('2005-01-04 10:10:37.474839'));
```

Then select from it to show that the insert has succeeded.

```
SQL >select * from timestamptest;
```

Then we commit to make the changes actually effective to the database itself.

```
SQL >commit
```

Then we need to get the SCN values which is Oracles chronological numbering system which is independent of UTC/GMT time.

```
SQL >select scn,operation,timestamp,username from v$logmnr_contents where table_name='TIMESTAMPTEST';
```

The query returns

```
SQL >622199 INSERT          05-JAN-2005 14:39:30 SYS
```

Use this SCN to get the sql\_redo statement from the redo logs via LogMiner view. This redo statement is the main point of LogMiner as it is saying that this is the command that is needed to bring the record back in case the record was deleted. We have not actually deleted the record as it can still be seen but we can still test the LogMiners statement to see if it is true to the original. This query gets us LogMiners redo statement to recreate the row we have inserted.

```
SQL >select sql_redo from v$logmnr_contents where scn=622199;
```

This is the result of the above select statement giving us LogMiners version of how to recreate the row.

```
SQL >insert into "SYS"."TIMESTAMPTEST"("TIMESTAMP") values (TO_TIMESTAMP('2005-01-04 10:10:37'));
```

```
SQL> insert into "SYS"."TIMESTAMPTEST"("TIMESTAMP") values (TO_TIMESTAMP('2005-01-04 10:10:37'));
1 row created.
```

The insert has completed and a select will show the result.

```
SQL> select * from timestamptest;
```

```
TIMESTAMP
```

```
-----
2005-01-04 10:10:37.474839
2005-01-04 10:10:37.000000 --fractional second data lost!
```

The top row is the original and the second row is the redo log from LogMiner. As can be seen LogMiner has failed to preserve the original decimal seconds and has rounded them to 0.

It is reasonable to assume that fractional second data recorded in a database is there for a reason otherwise it would not be recorded. Losing that data during a recovery without any warning to the user could be disastrous. LogMiner gives the impression that it is precise to the fraction of a second by the use of a "TIMESTAMP" column when in fact it is not. This fact should in my opinion be built into any proposed use of LogMiner as a Forensics investigation tool. If an MD5 hash was taken of the original data files it would be different from the copy as the data is different. The difference could be mistakenly interpreted as a corruption or deliberate attempt to tamper with evidence. The datafiles are the actual data that Oracle records in a file viewable through the OS. To an investigator this is quite an important issue and might cause a lack of confidence with using LogMiner for forensic activity.

The problem has arisen I think because the system has not been updated correctly since 8i as the column has the same name TIMESTAMP when the datatype is in fact a DATE. 8i did not have TIMESTAMP datatypes only DATES and so the name of the column was not misleading before 9i brought us fractional second data with 9i. This describes the original tool, which was developed for 8i.

<http://metalink.oracle.com/metalink/plsql/showdoc?db=not&id=114482.1>

Since the TIMESTAMPS in LogMiner are actually DATES and only precise to one second this does not explain the grouping of when one checks the SCN numbers one can see that they are the same also. This is because multiple items of change occur at the same time in a transaction. The redo logs record the changes as they occur to the database in a transaction. The logs do not record each individual change exactly at the time they are issued by the user. This is quite an important concept for an analyst to grasp but as long as it is grasped their use of the tool will be based upon this knowledge and no misrepresentation should occur.

There is a possibility that LogMiner is correctly reporting what it sees in the redo logs and the logs are only correct to a single second. I expect that the redo log does record TIMESTAMP values.

In order to accurately evaluate whether or not LogMiner is the source of the imprecision we are going to have to manually read the redo logs. This has some difficulty because they are in binary format. But there is a way to read them.

## 13 Test 3 – LogMiners error or the logs?

The point of this test is to try to ascertain whether it is LogMiner that is the source of the imprecision in reporting TIMESTAMPS from the redo logs or perhaps it is that the redo logs themselves that lose the fractional decimal places of a second.

I have been able to dismiss the redo logs as the source of the error by translating the hexadecimal code for TIMESTAMP and DATE datatypes, which are encoded in the redo logs. I will describe the process by which I did this as I believe it is of interest and relevance to the paper as this is a way of directly verifying the data that LogMiner is reporting.

We are going to have to open up the redo logs and try to read the TIMESTAMP values in there and see if they are recorded with the full decimal places or not.

A paper by Graham Thornton explains how to convert the redo logs into ASCII.

Dissassembling the Oracle Redolog written by Graham Thornton

<http://www.oraFAQ.com/papers/redolog.pdf> (Thornton 2000). This is a good paper, though it is not too difficult to convert the redo logs to ASCII as one just uses the “alter system dump logfile” command. The real problem is then trying to read the ASCII that this made, especially when it is in HEX and the HEX refers to an Oracle specific numbering system representing time as I will show. I must also give thanks to John Netherwood at this point for his expert guidance regarding the Oracle logging system.

The method used is to put a timestamp into the db one with decimal places and another with out and then read the difference in the converted log.

Here are two TIMESTAMPS put into a table one with decimal places one and without.

```
SQL> insert into TIMESTAMPTEST values ( TO_TIMESTAMP('07-JAN-05 03.19.36.654321'), 'xxx');
SQL> insert into TIMESTAMPTEST values ( TO_TIMESTAMP('07-JAN-05 03.19.36'), 'xxx');
```

Now check that the data has been recorded in the database.

```
SQL> select * from timestamptest;
```

TIMESTAMP	MARKER
04-JAN-05 10.10.37.474839 AM	
04-JAN-05 10.10.37.000000 AM	
07-JAN-05 03.19.36.000000 AM	xxx
07-JAN-05 03.19.36.654321 AM	xxx

Note that because the datatype is TIMESTAMP it reports times without decimal information as .000000

Now we wish to convert the timestamp into ASCII from its normal binary format. Problem is that we do not know which is the log that is currently being written to with our inserts in. Therefore we need to know which redo log file is active .

```
SQL> select * from v$log;
```

GROUP#	THREAD#	SEQUENCE#	BYTES	MEMBERS	ARC	STATUS
FIRST_CHANGE#	FIRST_TIM					
1	1	44	10485760	1	YES	INACTIVE
2	1	45	10485760	1	YES	INACTIVE
3	1	46	10485760	1	NO	CURRENT

Group 3 is current and has not been archived, as can be seen, so this is the one we will convert to ASCII. This gives us a number to search the next column in order to find the redo log number.

```
SQL> select * from v$logfile;
GROUP#      STATUS      TYPE MEMBER                                IS_
-----
3          ONLINE /export/home/u01/app/oracle/oradata/sales/redo03.log NO
2          ONLINE /export/home/u01/app/oracle/oradata/sales/redo02.log NO
1          ONLINE /export/home/u01/app/oracle/oradata/sales/redo01.log NO
```

So we can see that the **redo03.log** is the current log with the group number 3. Then we must convert redo log 3 into ASCII.

```
SQL> alter system dump logfile '/export/home/u01/app/oracle/oradata/sales/redo03.log'
```

But where has it saved the ASCII dump that it converted from redo03.log? We can query the database to see where it is putting the dump.

```
SQL> select value from v$parameter where name='user_dump_dest'
VALUE
-----
/export/home/u01/app/oracle/admin/sales/udump
```

Then from Korn shell

```
cd /export/home/u01/app/oracle/admin/sales/udump
$ ls -ltr to find the last file date and reverse order
```

Now we can open the redo log dump file. It is easy to identify as it is the one that is getting bigger and grows quite quickly to over 200 megabytes on Machines 1 and 2. In fact too big to use Vi and have to use More instead to view the file.

We now need to know what to look for in the ASCII version of the redo log. Most of the values are in HEX as the dictionary information needed to make it human readable is not in the log, therefore all the column references are in machine code and the values are in HEX. HEX conversion is not a problem, in fact as a test we converted the systimestamp to HEX using the dump command. This shows timestamp to HEX notation which as I say does not represent human readable times as you can see.

```
SQL> select dump(SYSTIMESTAMP) FROM dual;
DUMP(SYSTIMESTAMP)
-----
Typ=188 Len=20: 213,7,1,7,16,41,4,109,184,176,94,11,0,0,5,0,0,0,255,127
```

We do not actually have to decode the Oracle time numbering system as all we need to know is whether the redo log records TIMESTAMPS with decimal place data or not.

We now need to map the SQL insert statements to the entries in the ASCII redo logs to see if the redo logs record fractional seconds data or not. So going back to the inserts that we made at the beginning of this test we can see that the TIMESTAMP without the decimal places data is the same HEX number that is present in the redo logs. The 78 78 78 “xxx” marker is the location marker in the big file.

```
SQL> insert into TIMESTAMPTEST values ( TO_TIMESTAMP('07-JAN-05 03.19.36.'), 'xxx');
```

The letters are recorded in their usual HEX format in the redo logs unlike TIME.

```

null: --
col 0: [ 7] 78 69 01 07 04 14 25
col 1: [ 3] 78 78 78
CHANGE #4 MEDIA RECOVERY MARKER SCN:0x0000.00000000 SEQ: 0 OP:5.20
session number = 162
serial number = 3
transaction name =

```

(note this is from the ASCII redo logs)

This tells us that the number above 78 78 78 is the TIMESTAMP field in the table we have created. That is 78 69 01 07 04 14 25 represents the TIMESTAMP without the extra decimal places.

78 69 01 07 04 14 25 is the number that it generated for the “DATE like” TIMESTAMP. What does the number mean? There are 7 fields of double HEX. <http://centricle.com/tools/ascii-hex/> tells us that 07 is the ‘bel’ signal which will not be use much use for a TIMESTAMP unless we are making an alarm clock. However we do not need to decode it, we can just compare the same field when we look at the ASCII corresponding to the full TIMESTAMP example below.

```
SQL> insert into TIMESTAMPTEST values ( TO_TIMESTAMP('07-JAN-05 03.19.36.654321'), 'xxx' )
```

```

null: --
col 0: [11] 78 69 01 07 04 14 25 27 00 25 68
col 1: [ 3] 78 78 78
CHANGE #4 MEDIA RECOVERY MARKER SCN:0x0000.00000000 SEQ: 0 OP:5.20
session number = 162
serial number = 3
transaction name =

```

We can see the added decimal places in the insert and in the redo log compared to the previous example, which did not have extra decimal place information in either. The GREEN shows us the extra data that is recorded in the redo logs. Here we have proof that the redo logs do record the fractional decimal places of a second as a true TIMESTAMP, so it must be LogMiner that is the source of the imprecision.

LogMiner is converting all the TIMESTAMPS to DATES and losing 9 potential decimal places of precision in the timelines it is used to create.

Since this conclusion has weighty ramifications it is certainly worth running further tests to verify this and show repeatability. I repeated this test three times and the same results occurred. Here is output from one of these verification tests below with a different marker.

```
insert into TIMESTAMPTEST values ( TO_TIMESTAMP('22-JUN-04 11.11.11'), 'yyy' );
```

```

null: --
col 0: [ 7] 78 68 06 16 0c 0c 0c
col 1: [ 3] 79 79 79
CHANGE #4 MEDIA RECOVERY MARKER SCN:0x0000.00000000 SEQ: 0 OP:5.20
session number = 162
serial number = 3
transaction name =

```

## 14 Presentation

The subject of test result presentation for LogMiner is an interesting one. It would be tempting to use the LogMiner viewer for presentation purposes but I did find it unreliable on Linux. I guess it would not give a good impression of the tool if the viewer did not work reliably. Since the viewer and the command line tool are separate and the command line tool is very reliable in its working I would still not use the viewer. I would like to make further tests on the viewer on Solaris Sparc with different Java Virtual Machines as I suspect that there will be a combination that is reliable.

The beauty of the v\$logmnr\_contents table view is that one can use SQL to tailor the report in a very flexible way. It can be spooled off to a text file or printer very easily using the "SPOOL" command as I have shown in the previous tests. Further to that Oracle has really excellent reporting tools in the form of Forms and Reports software, which one can download from Oracle.com for free trial. The data can be presented either through a html interface over http using recent versions of Forms software or using Forms 6 can be presented through a native client interface. Either way, since the data is accessible through SQL, presentation options are very flexible.

<http://www.oracle.com/technology/products/forms/index.html> Pictorial and graphical representation of statistics would help increase the understanding of people who needed to interface with the results and so the added functionality of Oracle reports would probably be better for end users who did not need to interact with the data.

<http://www.oracle.com/technology/products/reports/index.html> The timeline especially would be well represented by an X/Y axis graph like the following screenshot that is taken from Oracle reports 9i.

<http://www.oracle.com/technology/products/reports/showcase/demo2b.jpg> Though this is a financial based report the same visual techniques can be used to report the forensic data from the v\$logmnr\_contents table view.

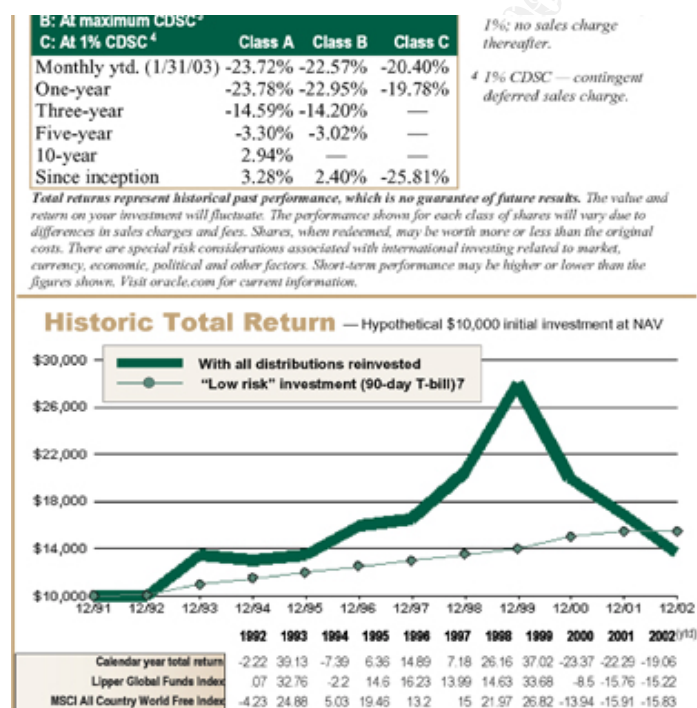


Figure 5 A timeline example created by Oracle reports (Oracle.com)

Perhaps the simplest and best method of presentation is the method I have used in the tests in this paper with the timeline, simply spooling the result of the SQL query to a text file. This has a number of benefits.

1. Can do an MD5 checksum of the report very easily.
2. Less things to go wrong so probably will not give inaccurate result.
3. Visually verifiable by a witness who is watching the procedure as it is a simple procedure.
4. Easily documented as the report generation is done on command line and can be repeated easily if needed.
5. The whole database can be exported to text format and backed up easily making verification of the data at a later stage easier.
6. Can potentially integrate the text output with .mac timeline information from the OS or IDS. Very powerful.

In the courtroom I think it would be best to have both command line SQL query capability as well as Oracle Reports in a semi processed format that is easier to represent the information. This way we can have the advantages of both. It would be beneficial and not too difficult to be able to query the information as required live in the court room to give confidence to the staff there that the technology is reliable. A live demonstration shows that the analyst is confident in their tools. This is a point that is open to discussion and feedback with the court officials.

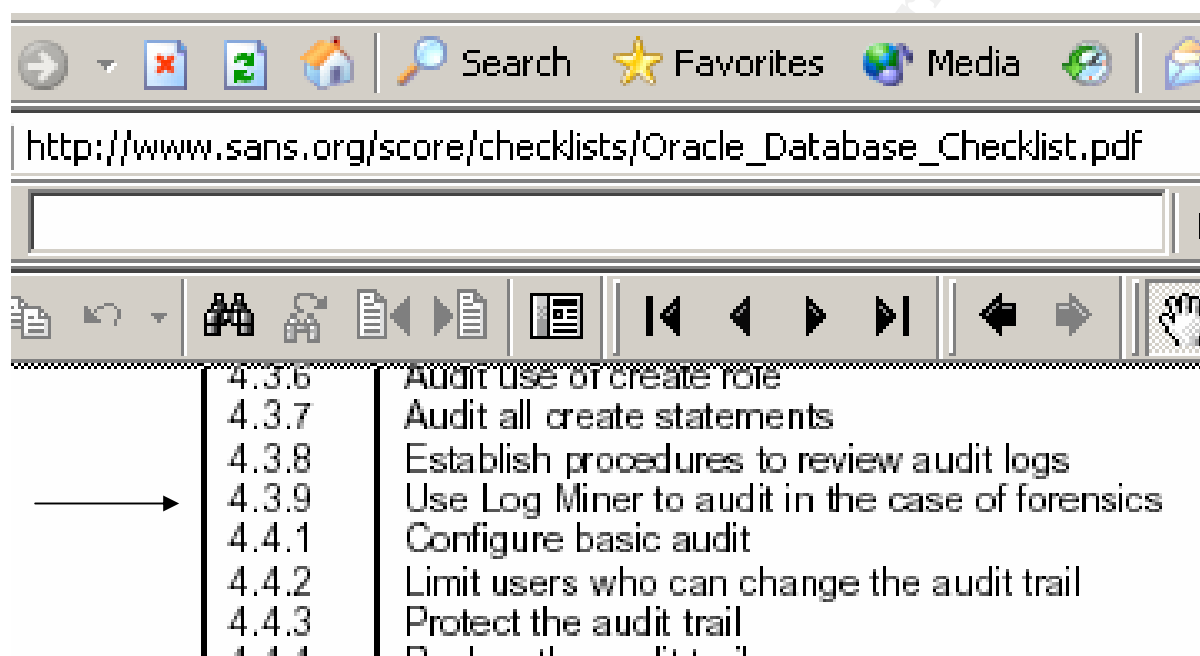
An important point regarding presentation in the courtroom is how the physical evidence is presented. We have talked about statistical evidence but this may seem easily manipulated or intangible to a court. The actual physical evidence in the case of LogMiner is the source redo logs. In section 10.1.5 we have backed up the redo logs that are being used and put them into a chain of custody with all the measures that need to be taken to guarantee integrity of evidence. This includes signing the bag and accompanying documentation with clear details of names and roles of each person handling the bag of which there should not be many. The airtight evidence bag label should be impossible to get off the bag without breaking the seal which is unique to that bag. This is backed up with photographic evidence of the bag and its original state to compare with the state with which it is presented in court. Of course MD5 hash sums of the original must be recorded in the bag and if necessary these can be confirmed in the court room. This is part of the presentation as the way in which evidence has been handled shows the integrity of the whole case.

## 15 Conclusions

This testing attempted to evaluate the usefulness of LogMiner to a forensics investigator. This was done successfully as expected. 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.

There are caveats to LogMiners use for forensics which are already known especially in action 4.3.9 of the Pete Finnegan's checklist for Oracle database security which states at this URL

[http://www.sans.org/score/checklists/Oracle\\_Database\\_Checklist.pdf](http://www.sans.org/score/checklists/Oracle_Database_Checklist.pdf)



**Figure 6 SANS Score checklist**

The caveats Pete already mentions include the fact that the actual SQL used to make db changes is not recorded, only equivalent SQL, which may or may not be the same. Plus LogMiner does not support the following.

1. MTS environments.
2. Objects.
3. Chained and migrated rows (some versions).
4. IOT, clustered tables or indexes.

I have been able to add some more caveats to LogMiners use as the follow up tests identified an imprecision in the way that LogMiner deals with TIMESTAMP data both in the way it reports TIMESTAMPS and recovers them as data from the redo logs. More specifically the TIMESTAMP column in the LogMiners reporting table is in fact a date, which means that all fractional second data is lost. The name of the column gives a misleading impression of the precision of the tool which is not designed to handle fractions of a second. Also when recovering TIMESTAMPS LogMiner losses all

fractional seconds data. This is not documented and has not been discussed in the public domain until now. The fault appears to be due to the upgrade of LogMiner from 9i not taking on board the creation of the new TIMESTAMP data type in 9i. These faults are more important to the Forensic Analyst than a normal user of the database, therefore forensic use of LogMiner should be tempered by these facts.

#### Recommendations:

1. Use the command line version not the GUI LogMiner viewer.
2. Realise that time and SCN relationship can vary and that this is not accurate to the fraction of a second.
3. Do not use LogMiner for recover of data that has fractional parts of second.
4. Try to synchronise time between all systems so that a single time line can be referred to by all machines. Page 9 of the SANS Track 8 notes recommends using GMT (Greenwich Mean Time) for this purpose.
5. If called in to do a forensic investigation of an Oracle database bear in mind that the online redo logs will overwrite themselves quite quickly (days) so if archiving is not enabled this evidence will be lost. Therefore backing up the redo logs would be an early action to take.

These tests were very successful in my opinion, as I have found out that the command line version of LogMiner can be used forensically with the caveats that I have identified. The tool can be used for incident response and will not change the file when reading from it.

#### Further work:

- 1) Write Perl script to integrate SQL generated Time line with OS level MACTimeline as per page 184 of Track 8 course materials.
- 2) Test LogMiner viewer with different VMs
- 3) Do time measurements of how accurate Oracle keeps time
- 4) Decode the Oracle specific time numbering system by working through more HEX examples.
- 5) How does the LogMiner tool know which order to put the redo log entries in within their SCN numbers. Also when a recovery is done from the RDBMS not through LogMiner how does it know which redo log entries to apply first within a group of entries with the same SCN or "TIMESTAMP"(DATE). I think it must use fractional seconds data but would like to spend more time and pages to find out but we are at our limit now.

The bigger picture behind the issues with forensic analysis is the relationship between a machine number like the SCN and time, either GMT or UTC. Tracing an event by SCN may be more accurate but humans do not work by SCN. Therefore a central point of this paper is to recommend getting to know the time limitations of the logging system that is running. There will probably be times when an event has to be accessed by time value and not by SCN (as it is not known) and knowing in what way the system is usually inaccurate could be a real lifesaver.

## 16 Cross Examination

Four arguments from the prosecution about the validity of the test results gained from LogMiner and my counter arguments. For the purposes of this exercise I will assume that a users actions have been recorded in the redo logs despite the user deleting the evidence as they thought. LogMiner was used to go back over the results and showed that the users actions caused the unauthorised deletion that was used to cover their tracks.

### 1. LogMiner is not a reliable time keeper

I would state that time itself is an abstract measuring system that can change depending on the speed you are travelling as Einstein discovered. Also GMT is 22 seconds adrift from UTC time. Plus we have the leap second <http://tycho.usno.navy.mil/leapsec.html>. Therefore Oracles decision not to base its continuum on time and instead use the SCN number is a wise one. The SCN number is completely reliable and shows the course of events in the order they occurred. These events can be cross referenced with time through LogMiner and through the auditing functionality of Oracle as a verification. All of which prove beyond reasonable doubt that the defendant deleted the records of which they are accused.

### 2. LogMiner is not able to give generally accurate results as can be seen by the fact that computers always go wrong.

I would state that the defendants counsel may have experience of computers going wrong because Windows is prevalent in the market place and is prone to unreliability. Mission critical systems use UNIX which are very reliable. LogMiner runs on UNIX and has been shown by my tests to work perfectly.

### 3. LogMiners results could have been tampered with

I would explain the concept of an MD5 hash and show that the hashes of the redo log files had not changed before and after and that a separate copy of the redo log had been archived with the same hash. I would invite the counsel to use the back up copy if they wished as it would show exactly the same results.

### 3. LogMiner could have been used incorrectly

I would explain that it is actually a very easy tool to use and demonstrate its use live in the courtroom with a projector so that the defending counsel and the court could witness its use for themselves.

## 17 Additional Information

More links relevant to this subject.

<http://www.samoratech.com/TopicOfInterest/swLogmnr.htm>

Application forensics. Forensically analysing the actions of an application. <http://www.securityfocus.com/infocus/1808>

Hackers becoming aware of forensics techniques and using anti forensics methods. <http://infosecuritymag.techtarget.com/2003/may/antiforensics.shtml>

The subject of database forensics has been touched upon before and suggestions to link database snapshots with IDS  
<http://www.cs.fsu.edu/~yasinsac/group/slides/suen2.pdf>

LogMiner has some shortcomings.

<http://www.securityfocus.net/infocus/1646> as Pete said here.

Lots of forensics papers that are quite interesting can be found at <http://www.e-evidence.info/s.html>.

## 18 References

(Finnigan 2004) - Oracle Security Step-By-Step, A survival guide for Oracle Security Version 2.0 by Pete Finnigan et al, published by SANS Press.

(Loney 2002)

Oracle9i: The Complete Reference

by Kevin Loney, George Koch McGraw-Hill 2002

(Loney 2000)

Oracle8i: The Complete Reference (Book/CD-ROM Package)

by Kevin Loney, George Koch Osborne/McGraw-Hill 2000

(NIST 2001)

General Test Methodology for Computer Forensic Tools

Version 1.9 -- November 7, 2001

National Institute of Standards and Technology

U.S. Department of Commerce

(NIST 2002)

Setup and Test Procedures

dd (GNU fileutils) 4.0.36 Forensic Tests

Version 1.0

August 1, 2002

National Institute of Standards and Technology

U.S. Department of Commerce

(Naudé 2003)

Oracle PL/SQL FAQ

\$Date: 01-Aug-2003 \$

\$Revision: 2.06 \$

\$Author: Frank Naudé \$

<http://www.orafaq.com/WRAP>

(1999 Oxford English Dictionary) Judy Pearsall and Bill Trumble, The Oxford English reference Dictionary Second Edition, Oxford University Press- Oxford and New York.

SANS Track 8 course materials London 2004.

SANS Track 4 course materials London 2003.

(Theralt 2001) – Oracle Security Handbook by Marlene Theralt and Aaron Newman, published by Oracle Press.

(Thomas 2002) – Oracle 9i DBA Fundamentals 1 study guide by Biju Thomas and Bob Bryla, published by SYBEX.

(Thornton 2000).

Dissassembling the Oracle Redolog written by Graham Thornton

Snr. Database Architect / Oracle DBA

Document Revision Date: 9/26/00

<http://www.orafaq.com/papers/redolog.pdf>

## 18.1 URLs used during the paper all active 10<sup>th</sup> January 2005 23:27

<http://www.petefinnigan.com/orasec.htm>

<http://www.sans.org/sans2005/description.php?cid=534>

<http://www.nsr.nist.gov/index/mfg.index.txt>

[http://databasejournal.com/features/oracle/article.php/10893\\_3446681\\_2](http://databasejournal.com/features/oracle/article.php/10893_3446681_2)

[http://www.oracle.com/technology/deploy/availability/htdocs/Flashback\\_Overview.htm](http://www.oracle.com/technology/deploy/availability/htdocs/Flashback_Overview.htm)

<http://www.dbazine.com/liu6.shtml>

[http://www.databasejournal.com/features/oracle/article.php/10893\\_3446681\\_2](http://www.databasejournal.com/features/oracle/article.php/10893_3446681_2)

<http://www.orafaq.com/articles/archives/000038.htm>

<http://www.oracle-base.com/articles/10g/Flashback10g.php>

[http://asktom.oracle.com/pls/ask/f?p=4950:8:::F4950\\_P8\\_DISPLAYID:5968150395572](http://asktom.oracle.com/pls/ask/f?p=4950:8:::F4950_P8_DISPLAYID:5968150395572)

[\[west.oracle.com/docs/cd/B14117\\\_01/server.101/b10739/archredo.htm#i1006971\]\(http://west.oracle.com/docs/cd/B14117\_01/server.101/b10739/archredo.htm#i1006971\)](http://download-</a></p></div><div data-bbox=)

<http://www.cftt.nist.gov/Toc532194078>

[http://www.cftt.nist.gov/setup\\_for\\_dd\\_tests.pdf](http://www.cftt.nist.gov/setup_for_dd_tests.pdf)

<http://www.ojp.usdoj.gov/nij/sciencetech/cftt.htm>

<http://www.cftt.nist.gov/testdocs.html>  
[http://216.239.59.104/search?q=cache:rBTH-x91EB0J:www.csa.syr.edu/James\\_Lyle.pdf+forensic+tool+evaluation&hl=en](http://216.239.59.104/search?q=cache:rBTH-x91EB0J:www.csa.syr.edu/James_Lyle.pdf+forensic+tool+evaluation&hl=en)  
<http://www.aic.gov.au/conferences/evaluation/armstrong.html>  
<http://www.orafaq.com/faqplsql.htm#WRAP>  
<http://www.nsr1.nist.gov/index/mfg.index.txt>  
[ftp://ftp.nist.gov/pub/itl/div897/nsr1/ver\\_2\\_0/nsr1\\_2\\_0.iso](ftp://ftp.nist.gov/pub/itl/div897/nsr1/ver_2_0/nsr1_2_0.iso)  
[http://www.lc.leidenuniv.nl/awcourse/oracle/appdev.920/a96612/d\\_logmn3.htm#77097](http://www.lc.leidenuniv.nl/awcourse/oracle/appdev.920/a96612/d_logmn3.htm#77097)  
<http://www.lc.leidenuniv.nl/awcourse/oracle/server.920/a96521/onlineredo.htm#3848>  
<http://www.lc.leidenuniv.nl/awcourse/oracle/server.920/a96521/archredo.htm>  
<http://www.cs.rose-hulman.edu/docs/oracle-817/server.817/a76956/onlinere.htm#6163>  
<http://home.clara.net/dwotton/dba/logminer.htm>  
[http://www.adp-gmbh.ch/ora/misc/dynamic\\_performance\\_views.html](http://www.adp-gmbh.ch/ora/misc/dynamic_performance_views.html)  
<http://www.oracle.com/technology/products/oracle9i/htdocs/9iobe/OBE9i-Public/obe-ha/html/logminer/logminer.htm>  
<http://www.iris.net/?prod=xcsc&ver=1.0.0>  
<http://www.softpedia.com/progDownload/Checksum-Calculator-Download-17463.html>  
<http://metalink.oracle.com/metalink/plsql/showdoc?db=not&id=114482.1>  
<http://www.orafaq.com/papers/redolog.pdf>  
<http://centricle.com/tools/ascii-hex/>  
<http://www.oracle.com/technology/products/forms/index.html>  
<http://www.oracle.com/technology/products/reports/index.html>  
<http://www.oracle.com/technology/products/reports/showcase/demo2b.jpg>  
[http://www.sans.org/score/checklists/Oracle\\_Database\\_Checklist.pdf](http://www.sans.org/score/checklists/Oracle_Database_Checklist.pdf)  
<http://tycho.usno.navy.mil/leapsec.html>  
<http://www.samoratech.com/TopicOfInterest/swLogmnr.htm>  
<http://www.securityfocus.com/infocus/1808>  
<http://infosecuritymag.techtarget.com/2003/may/antiforensics.shtml>  
<http://www.cs.fsu.edu/~yasinsac/group/slides/suen2.pdf>  
<http://www.securityfocus.net/infocus/1646>  
<http://www.e-evidence.info/s.html>

© SANS Institute

# 19 Appendices

MAC like time line. Separate file too big.

## 19.1 LogMiner errors from Metalink

2205649 ORA-29531 from DBMS\_LOGMNR\_CDC\_PUBLISH.DROP\_SUBSCRIBER\_VIEW

2259246 ORA-1280 if both DBMS\_LOGMNR.CONTINUOUS\_MINE and SCN range set

2336819 2nd call to START\_LOGMNR with DDL TRACKING fails

2338739 DBMS\_LOGMNR\_CDC\_PUBLISH / DBMS\_CDC\_PUBLISH may error with ORA-1426

2479415 Dump / OERI with >1 cursor selecting from V\$LOGMNR\_CONTENTS

2759170 Simultaneous DBMS\_LOGMNR\_D.BUILD may hang

2879367 Logminer needs logs from all threads in RAC or no rows are returned

2881035 OERI:17158 possible selecting from V\$LOGMNR\_CONTENTS

2901586 LOGMINER / Streams apply of CTAS with > 122 columns dumps

2942371 Redo for tables with row dependencies cannot be mined

3456259 V\$LOGMNR\_CONTENTS can get duplicate transaction start records for the same id

3492040 OERI selecting from V\$LOGMNR\_CONTENTS for long DDL

3528916 Logminer may dump with ROWNUM predicate and ASYNC IO used

3602321 Streams/logminer cannot handle partial rollback of a failed BULK BIND operation

3733268 Streams / Logminer has problems resolving redo for rolled back incomplete row changes

## 19.2 SANS Guide for Oracle Security sections relevant to LogMiner

1.4.5 Monitor Oracle log files (this implies a process over time so not within tool)

1.9.2 Save log files to a separate server using Syslog or Windows event viewer

1.13.1 Locate archive log files and check no user except software owner can read them

1.13.2 Save archivelog files to disk and purge

4.3.9 Use Log Miner to audit in the case of forensics

4.10.1 Audit and review the Oracle generated log files