Arguments and Methods for Database Data Model Forensics

H. Q. Beyers[1], M.S. Olivier[2] and G.P. Hancke[1]

[1]Department of Electrical, Electronic and Computer Engineering,
University of Pretoria, Pretoria, South Africa
[2]Department of Computer Science, University of Pretoria, Pretoria, South Africa
e-mail: hqbeyers@gmail.com

Abstract

A Database Management System (DBMS) consists of metadata and data. The metadata influences the way the data is presented to the user and this presents various forensic complications. The data model can be viewed as the highest level of metadata which governs the way other metadata and data in the DBMS are presented to the user. The data model can be modified to hide or tamper with forensic evidence. In this study the focus is on the data model of the DBMS and arguments are provided to indicate why the data model is an important consideration when conducting a forensic investigation on a DBMS. Various methods are presented to transform the data model into a desired state for a forensic investigation and these methods are measured against set out criteria. No one method is adequate for every forensic investigation. A forensic investigator should understand the various methods and select the correct data model state and method to convert the data model into that required state.

Keywords

Database Forensics, Data Model, Database Layers

1 Introduction

A Database Management System (DBMS) consists of various layers which can either be a form of metadata or data (Dyche, 2000). The ANSI-SPARC Architecture divided the database into various layers (ANSI/X3/SPARC Study Group, 1975). Some layers in the DBMS may influence the actions and results of other abstract layers and this fact presents a variety of vulnerabilities and forensic possibilities (Olivier, 2009). Based on the ANSI-SPARC Architecture, we divided the metadata and data of the DBMS into four abstract layers and assembled the metadata in order to get different results from the DBMS (Beyers *et al*. 2011). The first abstract layer of the DBMS is the data model which is a type of metadata and can be viewed as the source code of the DBMS. The second layer of the DBMS is the data dictionary which can be viewed as metadata that gets utilised to construct all the databases of the DBMS. The application schema includes user created operations that can manipulate data such as database triggers, procedures, and sequences, as well as the logical grouping of database objects such as views and tables. The fourth abstract layer is the application data which is the rows stored within the tables of a DBMS.

In this study the focus will fall on the data model abstract layer (source code) of the DBMS, because it can be viewed as the highest level of metadata which influences other metadata and data in the DBMS. The data model of a DBMS interprets the metadata and data stored in the DBMS and delivers the results accordingly, but the data model is code which can be altered to deliver almost any result. The data model can be changed to tamper with evidence such as table structures, records in a table etc. (Beyers *et al*. 2011). Similar to any other software program, the source code of the DBMS may present some flaws and vulnerabilities (Litchfield, 2005).

Our premise is that the ideal tool to examine the contents of a database will often be a DBMS: it provides powerful query facilities that enable the investigator to express the wish to easily formulate exactly what data is of interest. Possibilities exist to correlate sets of data, or to find data that is inconsistent with other data. Possibilities exist to formulate simple queries just like a typical user would have done it. Possibilities exist to query the metadata to determine the structure of data. And so on. However, in order to use the DBMS as a tool for such forensic database examination we need to know that we can rely on it to provide forensically useful results. Formulated slightly differently, the challenge is to combine the data to be investigated with a DBMS that is deemed reliable enough. Note that *reliable enough*, in general, means that the tool being used should exhibit sufficiently low error rates. The phrases *sufficiently low* or *reliable enough* are, amongst others, influenced by whether the case at hand is a criminal or civil matter, and whether the outcomes are to be used for evidentiary purposes or some lesser role in an investigation. This paper examines various ways in which a DBMS may be combined with data to be examined. It assumes the data to be examined is located on some computer that potentially contains evidence. For such data to be examined only two alternatives exist: either the data is examined on the computer where it is located, or the data is somehow imaged and copied to a forensic machine. This paper works systematically through the viable options, examines how the option may be effected in an examination and determines the reliability of such an option in qualitative terms.

These viable options will transform the data model into a required state. The state of the data model will have to fulfil one or more of the following criteria.

- Provide an environment where the evidence output from the DBMS is trustworthy and have not been altered by the data model.
- Provide an environment where the processes used to acquire consistent evidence from the DBMS have not altered the evidence.

In order to discuss the various methods and whether their outcomes adhere to the criteria mentioned above, this study is structured into the following four key areas. The various reasons why either a found data model or clean data model is important are discussed. In the next section a range of methods are presented to achieve a clean or found data model environment. The following section explains how this study fits into the bigger picture of database management system forensics and the remaining three abstract layers. Finally, a conclusion is presented.

2 Forensic Environment with a Clean or Found Data Model

The previous section briefly discussed how the data model may influence the output of the DBMS. Two types of data model forensic environments will now be explored; the clean and found environment. The difference between these two environments is that the found environment contains the original data model that was present when the incident of forensic interest occurred, while the clean environment represents an environment where the data model has been cleaned up in such a way that we can trust the data model. The found environment can further be divided into a found environment that either resides on the live machine or a found environment that is copied onto a completely different machine. In this section the various reasons why the found or the clean environments are useful will be discussed. Several scenarios are presented where the found or clean environments are advantageous.

2.1 Arguments for a Clean Data Model Environment

The clean environment is a setting where we have ensured that the data model will not alter the output of the DBMS. It is important to understand that a clean state differs from a post-mortem state characteristic of traditional digital forensics. A clean state is not merely a copy of the evidence that needs to be analysed, but rather a copy of the evidence program that runs like the original copy and from which we can query output. This means that the clean environment is set up to run like the original DBMS, but we are sure that the data model is not corrupting the output that we receive from the DBMS. The various reasons why it can be important consist of the following: (1) the data model as code, (2) the data model as a rootkit, and (3) the data model as a toolkit.

1. The Data Model as Code
The data model is the underlying software that controls the way the metadata and data of the DBMS are utilised. Just like any other code, the data model might have the ability and authority to control the way the metadata and data of the DBMS is stored, interpreted and processed. We can imagine that the code can be changed in endless different ways to achieve almost any outcome. Depending on the DBMS, the source code may be freely available on the internet. An attacker can edit the code to execute the attacker`s intended commands before installing or updating the compromised code on a host or server. Due to the fact that the data model can be rewritten to execute malicious tasks, hide data etc. it is important to ensure that the data model is clean when doing a forensic analysis.

2. The Data Model Rootkit
In this section we discuss the second reason why a clean data model state could be required. Rootkits may replace programs or executable files on a host to execute a different task than the executable file was originally intended to execute (Jakobsson and Ramzan, 2008). Rootkits are well known to attack operating systems, but it is possible that a rootkit may attack a DBMS. The database is similar to an operating system because both have users, processes and executables. Red Database Security (Kornbrust, 2005) illustrated how the Oracle database can be attacked by a rootkit.

Users were hidden in the database by inserting a username in the users table and then rewriting rules of a view to not display the inserted username. The view name in Oracle was sys.all_users and was changed to system.all_users and was set up to refer to the original view, but concealing some users. In a similar fashion the database processes were hidden by changing the view name of where the processes are stored from vsession to v_session.

An intruder can install a rootkit on a DBMS by acquiring access to the database through default password guessing, TNS Listener Exploits, Operating System Exploits etc. The example of installing a rootkit in an Oracle database is illustrated by Kornbrust (2005). (1) Create a file rootkit.sql on your own webserver. (2) Insert a HTTP-call into the glogin.sql file on the attacked DBMS client. (3) The next time the database administrator logs into the attacked DBMS the rootkit.sql is downloaded from the webserver. Then the rootkit.sql file is executed to disable terminal output, create a user, modify data dictionary objects, download and execute a keylogger (keylogger.exe), or do any other malicious task. A clean installation of the data model is required to ensure that the forensic evidence collection process will not be influenced by a rootkit.

3. Data Model as a Toolkit

Lastly we have a look at how the DBMS and its data model can be used as a toolkit. This is the third reason why a clean data model state could be required. Toolkits need to ensure that the evidence collection process does not alter the evidence at the same time. The EnCase toolkit brought about a revolution in the digital forensic arena when it introduced the concept of a toolkit that mounts the bit-stream forensic images as read-only virtual devices, therefore disabling any alteration of evidence on the digital device (Casey, 2010). We argue that the data model of the DBMS can also be considered as a type of digital forensic toolkit to extract evidence from the DBMS. While an operating system forensic toolkit extracts registry entries, retrieves deleted data, or validates partitions from the operating system, the data model acts as a forensic toolkit by providing records in tables, logging information and reporting database structures. Therefore it is important that the data model is in a state where the output it delivers can be trusted.

If the data model is used as a digital forensic toolkit, it should adhere to the requirements of a digital forensic toolkit. A toolkit should be (Daniel, 2012): (1) Definable. You must be able to state the problem, describe the desired outcome, develop the algorithm and validate the process. (2) Predictable. The tool must be designed to function in a predictable manner across any usage of the tool. (3) Repeatable. The function must be repeatable within a small error range. (4) Verifiable. The output must be able to be verified in various toolkit testing environments. When we have a clean data model we can say that the data model will be definable, predictable and repeatable. We argue that the data model can be used as a digital forensic toolkit when the data model is clean.

2.2 Arguments for a Found Data Model Environment

A found environment refers to a state of the data model where the data model was in use in the DBMS when an event of forensic interest occurred. The found environment may also refer to an environment where the same machine where the DBMS was originally installed is not used, but the data model was mirrored onto another machine. It is vital to understand that the found environment is not exactly the same here as the traditional meaning of a live digital forensic environment, due to the fact that the environment may fully or partially exist on the live machine or on another machine. In this section reasons are discussed why there might be a need for a found data model environment when conducting a DBMS forensic investigation consisting of: (1) cost of forensic investigation, (2) testing a hypothesis, and (3) simple queries.

1. Cost of Forensic Investigation
A vital consideration before conducting a forensic investigation is to determine how great the cost of the forensic investigation will be. Each technique of conducting a forensic investigation has a different level of complexity and different resource cost (R. Overill *et al*. 2009). These resource costs include the hiring of the expertise to conduct the investigation, the time that the investigation might take and the tools that the experts require to complete the investigation. A found investigation can be a very cost-effective investigation where the forensic investigator can log into the database and use simple database queries to extract all the evidence that is required. The first reason why an investigation of a DBMS with a found data model can be useful is due to its lower cost.

2. Testing a Hypothesis
The second reason why a found data model might be useful for a forensic investigation is to test a hypothesis. A hypothesis is an informed supposition to explain the observations in the data gathering phase of a forensic investigation (Casey, 2010). In other words, a hypothesis is a theory which an investigator might have about what occurred and this theory needs to be tested. A hypothesis might need to be tested in the DBMS, but there is no need to change the data model environment in order to test the hypothesis. In fact it is better to test a hypothesis when we know that the data model and underlying code is in the same state when the event of forensic interest occurred. In this instance a found data model will be useful to test the hypothesis.

3. Simple Queries
The third reason why a found data model can be useful during a forensic investigation is to run simple queries that will have no noticeable effect on the operations or state of the DBMS. The results that the found data model gives as output can still not be trusted entirely, but in some cases it might not be that critical to replace or cleanse all the code of the data model. There may be a situation where the investigator needs to search for clues and not forensic evidence. In this case the database may not be the primary suspect in the investigation, but rather an

independent witness. Only a found data model can provide the forensic investigators with such quick access to data.

3 How to Achieve Various States of the Data Model

Thus far this study has discussed why there is a need to do a forensic investigation on either a clean or found data model environment. In this section we will discuss various methods on how to achieve a clean or found data model state. The practical details in this section were limited due to document length constraints, but will be released in future work.

3.1 Clean Environment

To transform a data model into a clean state requires that the data model is transformed into a condition where the forensic investigator can trust the data model and be certain that the output of the DBMS is in no way corrupted by the data model. In short the data needs to remain the same, but the source code needs to be replaced or cleansed. A critical requirement for the transformation of the data model to a clean state is that the process is forensically sound and does not corrupt the data model even further. Various methods to achieve a clean data model are discussed in this section consisting of: (1) copying data files, (2) recovering the DBMS, and (3) patching the DBMS.

1. Copying Files
One of the approaches to achieve a clean state of the data model is to copy the data files from the suspect installation of the DBMS to a new installation of the DBMS, where a clean copy of the DBMS has been installed. The logic here is that a new installation of the DBMS on another machine will provide a clean data model. The data files of the new installation will be replaced with the data files of the suspect installation. A clean data model along with the data that exist on the suspect machine will then be achieved. This process has been tested with positive results on a PostgreSQL DBMS in one of our studies (Beyers *et al*. 2011). This copy process is illustrated in figure 1. This method adheres to both data model criteria. The data model will not alter the evidence and the evidence can be trusted.

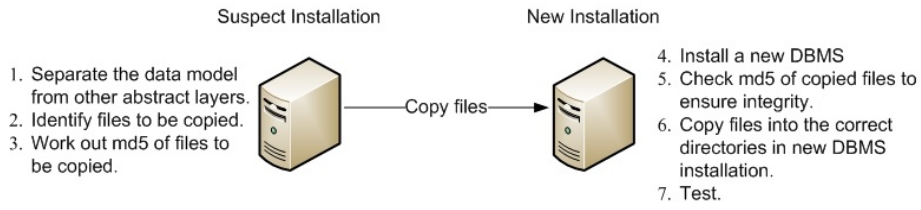

Figure 1: The copy process to achieve a clean data model is displayed.

2. Recover DBMS
Another approach to achieve a clean data model state in the DBMS is to recover the DBMS to a previous state with backup data. If we have adequate backup data of the DBMS we can recover the DBMS to a state that is similar to the state of the suspect DBMS, but with a clean data model. Old data dumps of the entire DBMS will work particularly well in this instance because data dumps will include the application schema and application data if the data dump was made before the DBMS was corrupted. Another source of recovery data is a backup server for the DBMS where it can be proven that the backup server was not compromised and we can trust the data model (Curtis Preston, 2007).

Once we have the acceptable amount of backup data we need to install a clean copy of the DBMS onto another machine. The installation should be exactly the same as the suspect DBMS and all patches and add-ons should be updated and installed. The application data and application schema can then be imported to the new DBMS installation by either entering the data through the DBMS console interface or by making use of the DBMS's data import tools. Unlike the Copying Files method, this method makes use of the clean data model to import the data into the clean DBMS installation. If the investigation requires that the data must be as similar as possible to the suspect DBMS's data it might be useful to run any scripts on the data that would have run on the suspect machine. This is a practical approach to accomplish a clean data model, because it is common practice to make regular backups of the DBMS either to a FTP server or a backup server with a DBMS installed on it. This method adheres to at least one of the two criteria because the data model can be trusted, and if the out-dated backup data has no effect on the evidence, this method will also not alter the evidence of the DBMS.

3. Patching the DBMS
Another method to achieve a clean data model is by patching the DBMS. A DBMS patch is software designed to fix problems or update the software of the DBMS (Greenwald *et al*. 2011). A patch has the potential to update the data model of the DBMS and thus provides a clean and trusted (or at least a partially trusted) data model by merely running a patch script. The patch is a trusted source of code and if this code can replace the untrusted data model code a clean or cleaner data model may be achieved effortlessly. The two scenarios when this method can be utilised is: to clean up the entire data model to acquire trusted results, or to fix odd results that are retrieved on the suspect DBMS by cleansing the data model with a patch. In the first scenario it needs to be determined if there is a patch for the relevant DBMS that will clean up the entire data model without influencing the evidence within the DBMS. Patches will specify precisely what parts of the DBMS the patch will update in the patch readme and the extent of the data model clean-up can be determined from the specification. In the second scenario the suspect DBMS gives unexpected output that could be caused by various reasons such as hidden data, malicious code, etc. A patch may be applied to attempt to eliminate the incorrect output and reveal new evidence. If changes occurred between the suspect DBMS's output and patched DBMS's output, the patch readme document will provide us with a focus area to determine where in the data model the problem occurred.

The DBMS must be installed on another machine with the same DBMS version and add-ons as the suspect DBMS to make sure the installation creates all required configuration on the new machine. Then the files of the entire original DBMS needs to be copied onto the new machine. The hash must be calculated to ensure that no data has changed in the copy process from the original machine to the new machine. The copied files of the original DBMS must now replace the files in the new installation. Finally the selected patches should be applied to the new DBMS installation. The new DBMS copy can now be tested and the investigation can start on the clean DBMS. An advantage of this method is that we can find an almost exact copy of the suspect DBMS, because we do not have to revert to a backup. A disadvantage of this method is the fact that it might be difficult to prove that the patch will clean up the entire data model. This method adheres to both the criteria. The data model can be trusted and the evidence will not be altered.

3.2 Found Environment

The previous section discussed the various methods that can be used to achieve a clean state of the data model which will be acceptable for a forensic investigation. In this section the discussion will include methods to achieve a found data model environment which is acceptable to conduct a forensic investigation in. A found data model involves that the original data model of the DBMS is utilised to conduct a forensic investigation. The various methods to conduct an investigation on a found data model consist of: (1) an on-site investigation, (2) replacing fragments of the DBMS, and (3) backing up the DBMS.

1. On-site Investigation
The on-site investigation involves doing an analysis on the original DBMS where an event of forensic interest has occurred. The on-site analysis provides the investigator with access to the DBMS without doing much preparation of an environment. Traditional live investigation practices need to be applied here and the investigator needs to be mindful not to corrupt or change the evidence. It is good practice to back up the DBMS before starting the investigation and make a copy of the log before and after the investigation to record what the investigator has done during the analysis. This approach gives the investigator quick access to potential evidence and does not require a list of tasks and specialised expertise to prepare the correct investigative environment. This method is useful when information needs to be extracted from the DBMS and the DBMS is not a suspect, but rather an impartial witness. This method may adhere to one of the criteria if care is taken and evidence is not altered.

2. Replace fragments of the DBMS
Another approach to conduct a forensic investigation of a DBMS with a found data model is to replace fragments of the DBMS installation on the suspect machine in order to realise a more trustworthy data model. In this context, replacing fragments refers to the replacing of executable files or other files that belong to the suspect data model with a similar but trusted file. If tests can be conducted to determine which part of the data model cannot be trusted, it is possible to replace that fragment of the data model with this method. Firstly, we need to determine which part of the data

model has been compromised. An md5 hash comparison between the suspect DBMS and clean DBMS may expose alterations in the data model. If a file can be linked to the data model and the replacement of the file will have no effect on the data or schema, the file can be approved for replacement. Once the investigator has found files to replace the corrupt files from a trustworthy data model, the investigator first needs to back up the current DBMS so the DBMS can be rolled back if something goes wrong. Thereafter the files can be replaced by the new clean files. If the corrupt activities of the data model can be narrowed down to a couple of files it may be easy to swap the files with clean files from the source code of the DBMS. This approach adheres to both the criteria if applied correctly. The data model can be trusted and the evidence will remain unchanged. Note that this fragment replacement method is a hybrid between achieving a clean or found data model. The live installation of the suspect DBMS is used, but the data model is cleansed by replacing fragments.

3. Backup the DBMS
It is standard practice to back up the DBMS to protect the company's revenue from a DBMS failure. When conducting a forensic investigation a situation may exist where the original DBMS has been compromised, but the backup DBMS on another machine or medium is still functioning normally and was not affected by the compromise. The original DBMS has a compromised data model that cannot be trusted which points to all our data, and the backup DBMS has a trusted data model with all the same data. When using the backup DBMS for forensic evidence it must be proved that the processes used to make the backup copy of the DBMS are just as forensically correct than making a copy of the original DBMS in a forensically correct way and taking the evidence into the lab. It should also be proved that no user edited the data on the backup server. This method adheres to one of the criteria, because we can trust the data model.

4 Data Model Forensics in Context the Other Abstract Layers of the DBMS

Problems or anomalies will almost always be detected in the lower abstract layers (like the application data and application schema) of the DBMS, because these layers are used on a daily basis and are understood by more people than the higher layers (like the source code) of the DBMS. The problem can be detected in either data results after running a query or by detecting a fault in the DBMS structures when attempting to use that structure. Some investigations might be conducted and concluded in the lower layers only, but when the attacker attempted to clear his tracks or when the output of the DBMS cannot be explained by the lower levels, the higher levels of the DBMS should be considered. This study explains how the data model can be prepared for a forensic investigation when the higher abstract layer (the data model) of the DBMS has been compromised.

5 Conclusion

Because the data model cannot be trusted to deliver correct evidence, the state of the data model should be considered when conducting a forensic investigation on a

DBMS. This paper systematically presented the methods to transform the data model into a forensically sound state. There are various forensic methods that could be used in various circumstances to conduct a data model sensitive investigation. Depending on the forensic environment, the DBMS investigator should understand the arguments for either a clean or found data model environment, and consequently select a method that is appropriate for the investigation, keeping in mind how much of the criteria the method adheres to. This study argues that no one method is adequate for every forensic investigation. The methods in this paper should be tested against a list of DBMS forensic scenarios to determine the practical feasibility of the methods. Future work includes the practical implementation of a database rootkit and an in-depth study about the rest of the abstract layers of the DBMS.

6 References

ANSI/X3/SPARC Study Group (1975), 'Data Base Management Systems: Interim Report', *ACM SIGMOD bulletin*, vol. 7, no. 2.

Beyers, HQ, Olivier, MS & Hancke, GP (2011), 'Assembling the Metadata for a Database Forensic Investigation', in *Advances in Digital Forensics: Proceedings of the 7th IFIP International Conference on Digital Forensics, 4 February 2011*, Springer, New York, ISBN: 1868-4238.

Casey, E (2010), *Handbook of Digital Forensics and Investigations*, Elsevier, California, ISBN: 978-0-12-374267-4.

Curtis Preston, W (2007), *Backup and Recovery*, O`Reilly, Sebastopol, ISBN: 0596102461.

Daniel, L (2012), *Digital Forensics for Legal Professionals: Understanding Digital Evidence from the Warrant to the Courtroom*, Elsevier, Waltham, ISBN: 159749643X.

Dyche, J 2000, *e-Data: Turning Data into Information with Data Warehousing,* Addison-Wesley, New Jersey, ISBN: 0201657805.

Greenwald, R, Stackowiak, R, Alam, M & Bhuller, M (2011), *Achieving Extreme Performance with Oracle Exadata: Best Practices for Deployments in Enterprise Datacentres*, McGraw Hill, ISBN: 0071752595.

Jakobsson, M & Ramzan, Z (2008), *Crimeware: Understanding New Attacks and Defences*. Pearson Education, Boston, ISBN: 0321501950.

Kornbrust, A (2005), *Database Rootkits*: *Presented at Black Hat Europe, 29 March 2005*, viewed 4 November, 2011, <http://www.red-database-security.com/whitepaper/presentations.html>

Litchfield, D, Anley, C, Heasman, J & Grindlay, B (2005), *The Database Hacker's Handbook: Defending Database Servers*, John Wiley and Sons, Indianapolis, ISBN: 0764578014.

Olivier, MS (2009), 'On Metadata Context in Database Forensics', *Digital Investigations*, vol. 5, pp. 115-123.

Overill, R, Kwan, M, Chow, K, Lai, P & Law, F (2009), 'A Cost-Effective Model for Digital Forensic Investigations', *Advances in Digital Forensics: Proceedings of the 5th Annual IFIP WG 11.9 International Conference*, *28 January 2009*, Springer, New York, ISBN: 1868-4238.

Patzakis, J (2002), 'The Encase Process' in E Casey (ed.), *Handbook of Computer Crime Investigation: Forensic Tools and Technology*. Elsevier, California, ISBN: 0-12-163103-6.