THE IMPACT OF TRIGGERS ON FORENSIC ACQUISITION AND ANALYSIS OF DATABASES

W. K. Hauger* and M. S. Olivier**

** ICSA Research Group, Department of Computer Science, Corner of University Road and Lynnwood Road, University of Pretoria, Pretoria 0002, South Africa E-mail: whauger@gmail.com*
*** Department of Computer Science, Corner of University Road and Lynnwood Road, University of Pretoria, Pretoria 0002, South Africa E-mail: molivier@cs.up.ac.za*

Abstract: An aspect of database forensics that has not received much attention in the academic research community yet is the presence of database triggers. Database triggers and their implementations have not yet been thoroughly analysed to establish what possible impact they could have on digital forensic analysis methods and processes. This paper firstly attempts to establish if triggers could be used as an anti-forensic mechanism in databases to potentially disrupt or even thwart forensic investigations. Secondly, it explores if triggers could be used to manipulate ordinary database actions for nefarious purposes and at the same time implicate innocent parties. The database triggers as defined in the SQL standard were studied together with a number of database trigger implementations. This was done in order to establish what aspects of a trigger might have an impact on digital forensic analysis. It is demonstrated in this paper that certain database forensic acquisition and analysis methods are impacted by the possible presence of non-data triggers. This is specific to databases that provide non-data trigger implementations. Furthermore, it finds that the forensic interpretation and attribution processes should be extended to include the handling and analysis of all database triggers. This is necessary to enable a more accurate attribution of actions in all databases that provide any form of trigger implementations.

Keywords: database forensics, database triggers, digital forensic analysis, methods, processes.

1. INTRODUCTION

Forensic science, or simply forensics, is today widely used by law enforcement to aid them in their investigations of crimes committed. Forensic science technicians, which are specifically trained law enforcement officials, perform a number of forensically sound steps in the execution of their duties. These steps include the identification, collection, preservation and analysis of physical artefacts and the reporting of results. One critical part is the collection and preservation of physical artefacts. The collection needs to be performed in such a manner that the artefacts are not contaminated. The artefacts then need to be preserved in such a way that their integrity is maintained. The reason why this part is so critical is so that any evidence gained from the analysis of these artefacts can not be contested. The evidence found would be used to either implicate or exonerate any involved parties. Any doubt about the integrity of the artefacts collected could lead to the evidence being dismissed or excluded from legal proceedings.

In digital forensics these steps are more commonly referred to as processes. There have been a number of process models developed to guide the digital forensic investigator [1]. The digital forensic process that matches the collection and preservation step in the physical world is the acquisition process. Traditionally, this process involves the making of exact digital copies of all relevant data media identified [19]. However, database forensics needs to be performed on information systems that are becoming increasingly complex. Several factors influence the way that data is forensically acquired and how databases are analysed. They include data context, business continuity, storage architecture, storage size and database models. These factors and their influence on database forensics are examined further in Section 2.

Database triggers are designed to perform automatic actions based on events that occur in a database. There is a wide variety of commission and omission actions that can be performed by triggers. These actions can potentially have an effect on data inside and outside of the DBMS. Thus triggers and the actions they perform are forensically important. This was already recognised by Khanuja and Adane in a framework for database forensic analysis they proposed [4].

The effect that triggers can have on data raises the concern that they could compromise the integrity of the data being investigated. Could triggers due to their nature in combination with the way databases are forensically analysed lead to the contamination of the data that is being analysed? Another concern revolves around the automatic nature of actions performed by triggers. Can the current attribution process correctly identify which party is responsible for which changes? This paper attempts to establish if these concerns around triggers are justified.

The database trigger is defined in the ISO/IEC 9075 SQL standard [5]. Triggers were first introduced in the 1999 version of the standard and subsequently updated in the

Based on: "The Role of Triggers in Database Forensics", by Werner Hauger and Martin Olivier which appeared in the Proceedings of Information Security South African (ISSA) 2014, Johannesburg, 13 & 14 August 2014.

2008 version. The specification could thus be examined to determine on a theoretical basis if there is reason for concern. However, the standard is merely used as a guideline by DBMS manufacturers and there is no requirement to conform to the standard. Certain manufacturers also use feature engineering to gain a competitive advantage in the marketplace [6]. They might implement additional triggers based on actual feature requests from high profile clients. Standard triggers might be enhanced or other additional triggers implemented based on perceived usefulness by the manufacturers. These features could be used to overcome certain limitations in their DBMS implementations. It is therefore necessary to study actual trigger implementations, rather than the standard itself.

There are thousands of database implementations available and to investigate the trigger implementations of all those databases that use triggers would be prohibitive. Thus, the database trigger implementations of a few proprietary and open-source DBMSs were chosen. The DBMSs investigated were Oracle, Microsoft SQL Server, Mysql, PostgreSQL, DB2, SyBase and SQLite. These selected relational database management systems (RDBMS) are widely adopted in the industry. SQLite is particularly interesting since it is not a conventional database. SQLite has no own server or running process, but is rather a single file that is accessed via libraries in the application using it. SQLite is being promoted as a file replacement for local information storage. Some well known applications such as Adobe Reader, Adobe Integrated Runtime (AIR), Firefox and Thunderbird use SQLite for information storage. SQLite is also very compact and thus well suited for use in embedded and mobile devices. Mobile operating systems iOS and Android make use of SQLite [28,29].

The dominance of the selected RDBMSs in the market means that they would be encountered fairly often by the general digital forensic investigator. These RDBMSs are also the most popular based on the number of web pages on the Internet according to solid IT's ranking method [7]. The official documentation of these RDBMSs was used to study their trigger implementations. The latest published version of the documentation was retrieved from the manufacturer's website [8-12,25,26]. At the time of the investigation the latest versions available were as follows: Oracle 11.2g, Microsoft SQL Server 2012, Oracle Mysql5.7, PostgreSQL 9.3, IBM DB2 10, Sybase ASE 15.7 and SQLite 3.8.6.

This article is a reworked and extended version of a paper presented by the authors at the Information Security South Africa (ISSA) 2014 conference [30]. The popular databases Sybase and SQLite have been added to the investigation. The INSTEAD OF trigger which was later added to the standard is now also covered. This particular trigger raises additional challenges that are discussed under commission and omission.

Section 2 provides the database forensic background against which database triggers will be investigated. Section 3 describes the database trigger implementations investigated and is divided into four sub-sections: Firstly the triggers defined in the standard were explored. Then the implementations of the standard triggers by the selected DBMSs were examined. Thereafter, other non-standard triggers that some DBMSs have implemented were looked at. For each type of trigger the question was asked as to how the usage of that particular trigger could impact the forensic process or method. Lastly it was established on which objects triggers could be applied. Section 4 asks whether the current forensic processes would correctly identify and attribute actions if triggers were used by attackers to commit their crimes. Through the use of a few hypothetical examples as to how triggers could be used by attackers to commit their crimes, this question was investigated. Section 5 concludes this paper and contemplates further research.

2. BACKGROUND

Historically, digital forensics attempts to collect and preserve data media in a static state, which is referred to as dead acquisition [19]. Typically, this process starts with isolating any device that is interacting with a data medium by disconnecting it from all networks and power sources. Then the data medium is disconnected or removed from the device and connected via a write-blocker to a forensic workstation. The write-blocker ensures that the data medium cannot be contaminated while being connected to the forensic workstation. Software is then used to copy the contents to a similar medium or to an alternative medium with enough capacity. Hashing is also performed on the original content with a hash algorithm such as MD5 or SHA-1 [19]. The hashes are used to prove that the copies made are exact copies of the originals and have not been altered. The hashes are also used throughout the analysis process to confirm the integrity of the data being examined. Once the copies have been made, there is no more need for the preservation of the originals [2]. However, if the data being examined is to be used to gather evidence in legal proceedings, some jurisdictions may require that the originals are still available.

A different approach is to perform live acquisition. This involves the collection and preservation of both volatile data (e.g. CPU cache, RAM, network connections) and non-volatile data (e.g. data files, control files, log files). Since the acquisition is performed while the system is running, there are some risks that affect the reliability of the acquired data. These risks however can be mitigated by employing certain countermeasures [20].

In today's modern information systems there are several instances where it has become necessary to perform live acquisition. Firstly, in a permanently switched-on and connected world, the context around the imaged data may be required to perform the forensic analysis. This includes volatile items such a running processes, process memory, network connections and logged on users [19]. One area where the context gained from live acquisition is particularly useful is when dealing with possibly encrypted data. This is because the encrypted data might already be open on a running system and the encryption keys used cached in memory [21]. The increasing prevalence of encryption usage to protect data by both individuals and organisations increases the need for more live acquisitions to be performed.

Another instance where live acquisition is performed is when business continuity is required. For many organisations information systems have become a critical part of their operations. The seizure or downtime of such information systems would lead to great financial losses and damaged reputations. The shutdown of mission critical systems might even endanger human life. During forensic investigations, such important information systems can thus no longer be shutdown to perform imaging in the traditional way [19].

The complex storage architecture of today's information systems also necessitates the use of live acquisition techniques. To ensure availability, redundancy, capacity and performance, single storage disks are no longer used for important applications and databases. At least a redundant array of independent disks (RAID) or a full blown storage area network (SAN) is used. Both of these technologies group a variable number of physical storage disks together using different methodologies. They present a logical storage disk to the operating system that is accessible on the block-level.

In such a storage configuration a write-blocker can no longer be efficiently used. There simply may be too many disks in the RAID configuration to make it cost and time effective to image them all [19]. In the case of a SAN, the actual physical disks holding the particular logical disk might not be known, or might be shared among multiple logical disks. These other logical disks may form part of other systems that are unrelated to the application or database system and should preferably not be affected. Attaching the disks in a RAID configuration to another controller with the same configuration can make the data appear corrupt and impossible to access. RAID controller and server manufacturers only support RAID migration between specific hardware families and firmware versions. The same would hold true for the imaged disks as well.

While it is still technically possible to image the logical disk the same way as a physical disk, it may not be feasible to do so either. Firstly the size of the logical disk may be bigger than the disk capacity available to the forensic investigator [24]. Secondly the logical disk may hold a lot of other unrelated data, especially in a virtualised environment. Lastly organisations may be running a huge single application or database server containing many different applications and databases. Due to hardware, electricity and licensing costs, the organisation may prefer this to having multiple smaller application or database servers.

Lastly, database systems have their own complexities that affect digital forensic investigations. The models used by the database manufacturers are tightly integrated into their database management systems (DBMS) and are many times of a proprietary nature. Reverse engineering is purposely being made difficult to prevent their intellectual property being used by a competitor. Sometimes reverse engineering is explicitly prohibited in the licensing agreements of the usage of the DBMSs. To forensically analyse the raw data directly is thus not very easy, cost-effective or always possible. The data also needs to be analysed in conjunction with the metadata because the metadata not only describes how to interpret the data, but can also influence the actual seen information [3]. The usage of the DBMS itself, and by extension the model it contains, has become the necessary approach to forensically analyse databases.

The database analysis can be performed in two ways: an analysis on site or an analysis in a clean laboratory environment. On site the analysis is performed on the actual system running the data base. In the laboratory a clean copy of the DBMS with the exact same model as used in the original system is used to analyse the data and metadata acquired [3]. Both ways can be categorised as live analysis due to being performed on a running system. In the first instance the real system is used, while in the second a resuscitated system in a more controlled environment is used e.g. single user, no network connection.

Due to all these complexities associated with applications and particularly databases, live acquisition is the favoured approach when dealing with an information system of a particular size and importance. Fowler documents such a live acquisition in a real world forensic investigation he performed on a Microsoft SQL Server 2005 database [23]. It should be noted that both the operating system and the DBMS are used to access and acquire data after being authenticated. To preserve the integrity of the acquired data, he uses his own clean tools that are stored on a read-only medium [20]. However, the mere accessing of the system will already cause changes to the data, thus effectively contaminating it before it can be copied. Since all the operations performed during the acquisition are documented, they can be accounted for during a subsequent analysis. Hence, this kind of contamination is acceptable as it can be negated during analysis.

Against this background of how forensic acquisition and analysis is performed on a database system, triggers are examined.

3. TRIGGER IMPLEMENTATIONS

This section firstly examines what types of triggers are defined in the standard and how they have been implemented in the DBMSs surveyed. It then looks at other types of triggers that some DBMSs have implemented. Lastly, the database objects that triggers can be applied to, are examined. Throughout the section, the possible impact on database forensics is explored.

3.1 Definition

The ISO/IEC 9075 standard part 2: Foundation defines a trigger as an action or multiple actions taking place as a result of an operation being performed on a certain object. The operations are defined as being changes made to rows by inserting, updating or deleting them. Therefore three trigger types are being defined: the insert trigger, the delete trigger and the update trigger. The action can take place immediately before the operation, instead of the operation or immediately after the operation. A trigger is thus defined as a BEFORE trigger, an INSTEAD OF trigger or an AFTER trigger. The action can take place only once, or it can occur for every row that the operation manipulates. The trigger is thus further defined as a statement-level trigger or as a row-level trigger.

3.2 Standard triggers

The first aspect that was looked at was the conformance to the ISO/IEC 9075 SQL standard regarding the type of triggers. All DBMSs surveyed implement the three types of data manipulation language (DML) triggers defined. The only implementations that match the specification exactly in terms of trigger types are those of Oracle and PostgreSQL. They have implemented all combinations of BEFORE/AFTER/INSTEAD OF/Statement-level/Row-level triggers. The others either place restrictions on the combinations or implement only a subset of the definition from the specification. DB2 has no BEFORE statement-level trigger, but all the other combinations are implemented. SQL Server and Sybase do not implement BEFORE triggers at all. Mysql and SQLite do not have any statement-level triggers.

PostgreSQL goes one step further and differentiates between the DELETE and TRUNCATE operation. Because the standard only specifies the DELETE operation, most databases will not execute the DELETE triggers when a TRUNCATE operation is performed. Depending on the viewpoint, this can be advantageous or problematic. It allows for the quick clearing of data from a table without having to perform possibly time consuming trigger actions. However, if a DELETE trigger was placed on a table to clean up data in other tables first, a TRUNCATE operation on that table might fail due to referential integrity constraints. The linked tables will have to be truncated in the correct order to be successfully cleared. PostgreSQL allows additional TRUNCATE triggers to be placed on such linked tables, facilitating easy truncation of related tables.

Since all three types of DML triggers defined by the standard rely on changes of data taking place i.e. either the insertion of new data or the changing or removal of existing data, the standard methods employed by the forensic analyst are not impacted. These methods are specifically chosen because they do not cause any changes and can be used to create proof that in fact no changes have occurred.

Some members of the development community forums have expressed the need for a select trigger [13]. A select trigger would be a trigger that fires when a select operation takes place on the object on which it is defined. None of the DBMSs surveyed implement such a select trigger. Microsoft however is working on such a trigger and its researchers have presented their work already [14]. Oracle on the other hand has created another construct that can be used to perform one of the tasks that the developers want to perform with select triggers: manipulate SQL queries that are executed. The construct Oracle has created is called a group policy. It transparently applies the output from a user function to the SQL executed on the defined object for a certain user group. The function can be triggered by selecting, inserting, updating or deleting data. The good news for the forensic analyst is that these functions will not be invoked for users with system privileges. So as long as the forensic analyst uses a database user with the highest privileges, the group policies will not interfere with his investigations.

The existence of a select trigger would have greatly impacted on the standard methods used by the database forensic analyst. One of the methods used to gather data and metadata for analysis is the execution of SQL select statements on system and user database objects such as tables and views. This would have meant that an attacker could have used such a trigger to hide or even worse destroy data. A hacker could use select triggers to booby-tap his root kit. By placing select triggers on sensitive tables used by him, he could initiate the cleanup of incriminating data or even the complete removal of his root kit should somebody become curious about those tables and start investigating.

3.3 Non-standard triggers

The second aspect that was investigated was the additional types of triggers that some DBMSs define. The main reason for the existence of such extra trigger types is to allow developers to build additional and more specialised auditing and authentication functionality, than what is supplied by the DBMS. However that is not the only application area and triggers can be used for a variety of other purposes. For example instead of having an external application monitoring the state of certain elements of the database and performing an action once certain conditions become true, the database itself can initiate these actions.

The non-standard triggers can be categorised into two groups: data definition language (DDL) triggers and other non-data triggers. From the DBMSs investigated, only Oracle and SQL Server provide non-standard triggers.

DDL triggers: The first group of non-standard triggers are the DDL triggers. These are triggers that fire on changes made to the data dictionary with DDL SQL statements e.g. create, drop, alter etc. Different DBMSs define different DDL SQL statements that can trigger actions. SQL Server has a short list that contains just the basic DDL SQL statements. Oracle has a more extensive list and also a special DDL indicator that refers to all of them combined. Since DDL SQL statements can be applied to different types of objects in the data dictionary, these triggers are no longer defined on specific objects. They are rather defined on a global level firing on any occurrence of the event irrespective of the object being changed. Both SQL Server and Oracle allow the scope to be set to a specific schema or the whole database.

These triggers once again rely on data changes being made in the database to fire and thus pose no problem of interference during the forensic investigation.

Non-data triggers: The second group of non-standard triggers are non-data triggers. These are triggers that fire on events that occur during the normal running and usage of a database. Since these triggers do not need any data changes to fire, they potentially have the biggest impact on the methods employed by the forensic analyst. Fortunately the impact is isolated because only a few DBMSs have implemented such triggers.

SQL Server, Oracle and Sybase define a login trigger. This trigger fires when a user logs into the database. SQL Server's login trigger can be defined to perform an action either before or after the login. Authentication however will be performed first in both cases, meaning only authenticated users can activate the trigger. That means the login trigger can be used to perform conditional login or even completely block all logins. An attacker could use this trigger to easily perform a denial of service (DoS) attack. Many applications today use some kind of database connection pool that dynamically grows or shrinks depending on the load of the application. Installing a trigger that prevents further logins to the database would cripple the application during high load. It would be especially bad after an idle period where the application would have reduced its connections to the minimum pool size.

Oracle's login trigger only performs its action after successful login. Unfortunately that distinction does not make a significant difference and this trigger can also be used to perform conditional login or completely prevent any login. That is because the content of the trigger is executed in the same transaction as the triggering action [16]. Should any error occur in either the triggering action or the trigger itself, then the whole transaction will be rolled back. So simply raising an explicit error in the login trigger will reverse the successful login.

Sybase distinguishes between two different kinds of login triggers. The first is the login-specific login trigger. The trigger action is directly linked to a specific user account. This kind of trigger is analogous to the facility some operating systems provide, which can execute tasks on login. The second kind is the global login trigger. Here the trigger action will be performed for all valid user accounts. Sybase allows both kinds of login triggers to be present simultaneously. In this case the global login trigger is executed first and then the login-specific trigger [27].

Both kinds of login triggers are not created with the standard Sybase SQL trigger syntax. Instead a two-step process is used. First a normal stored procedure is created, that contains the intended action of the trigger. Then this stored procedure is either linked to an individual user account or made applicable to all user accounts with built-in system procedures. Like with Oracle, the action procedure is executed after successful login, but within the same transaction. Thus it can be similarly misused to perform a DoS attack.

Microsoft has considered the possibility of complete account lockout and subsequently created a special method to login to a database that bypasses all triggers. Oracle on the other hand has made the complete transaction rollback not applicable to user accounts with system privileges or the owners of the schemas to prevent a complete lockout. Additionally, both SQL Server and Oracle have a special kind of single-user mode the database can be put into, which will also disable all triggers [15,16]. Sybase on the other hand has no easy workaround and the database needs to be started with a special flag to disable global login triggers [27].

A hacker could use this trigger to check if a user with system privileges, that has the ability to look past the root kits attempts to hide itself, has logged in. Should such a user log in, he can remove the root kit almost completely, making everything seem normal to the user even on deeper inspection. He can then use Oracle's BEFORE LOGOFF trigger to re-insert the root kit, or use a scheduled task [17] that the root kit hides to re-insert itself after the user with system privileges has logged off.

Another non-data trigger defined by Oracle is the server error trigger. This trigger fires when non-critical server errors occur and could be used to send notifications or perform actions that attempt to solve the indicated error.

The final non-data triggers defined by Oracle only have a database scope due to their nature: the database role change trigger, the database startup trigger and the database shutdown trigger. The role change trigger refers to Oracle's proprietary Data Guard product that provides high availability by using multiple database nodes. This trigger could be used to send notifications or to perform configuration changes relating to the node failure and subsequent switch over.

The database startup trigger fires when the database is opened after successfully starting up. This trigger could be used to perform certain initialisation tasks that do not persist and subsequently do not survive a database restart. The database shutdown trigger fires before the database is shut down and could be used to perform cleanup tasks before shutting down. These last two triggers can be similarly exploited as the login and logoff triggers by a hacker to manage and protect his root kit.

3.4 Trigger objects

The third aspect that was investigated was which database objects the DBMSs allowed to have database triggers. The standard generically defines that triggers should operate on objects, but implies that the objects have rows. It was found that all DBMSs allow triggers to be applied to database tables. Additionally, most DBMSs allow triggers to be applied to database views with certain varying restrictions. Only Mysql restricts triggers to be applied to tables only.

None of the DBMSs allow triggers to be applied to system tables and views. Triggers are strictly available only on user tables and views. Additionally, there are restrictions to the kind of user table and user views that triggers can be applied to.

This is good news for forensic investigators, since they are very interested in the internal objects that form part of the data dictionary. However, there is a move by some DBMSs to provide system procedures and views to display the data from the internal tables [22]. To protect these views and procedures from possible user changes they have been made part of the data dictionary. The ultimate goal seems to be to completely remove direct access to internal tables of the data dictionary.

This might be unsettling news for forensic investigators as they prefer to access any data as directly as possible to ensure the integrity of the data. It will then become important to not only use a clean DBMS, but also a clean data dictionary (at least the system parts). Alternatively the forensic investigator first needs to show that the data dictionary is uncompromised by comparing it to a known clean copy [3]. Only then can he use the functions and procedures provided by the data dictionary.

4. IDENTITY AND ATTRIBUTION

The login trigger example brings up another interesting problem. Once the forensic investigator has pieced together all the actions that occurred at the time when the user with system privileges was logged in, the attribution of those actions can be performed. Since the forensic investigator can now make the assumption that the picture of the events that took place is complete, he attributes all the actions to this specific user. This is because all the individual actions can be traced to this user by the audit information. Without looking at triggers, the investigator will miss, that the particular user was completely unaware of certain actions that happened, even though they were triggered and executed with his credentials.

These actions can be categorised into two groups: commission actions and omission actions. The BEFORE/AFTER trigger can be used to commission additional actions before or after the original operation is performed. Since the original operation is still performed unchanged, no omission actions can be performed. The outcome of the original operation can still be changed or completely reversed by actions performed in an AFTER trigger, but those actions are still commission actions. The INSTEAD OF trigger on the other hand can be used to perform actions in both groups. Normally this trigger is intended to commission alternative actions to the original operation requested. Like the BEFORE/AFTER trigger, it can also be used to commission actions in addition to the original operation. But importantly, it provides the ability to modify the original operation and its values. This ability also makes it possible to either remove some values or remove the operation completely. Operations that were requested simply never happen and values that were meant to be used or stored disappear. These removal actions therefore fall into the omission action group.

Consider a database in a medical system that contains patient medical information. An additional information table is used to store optional information such as organ donor consent, allergies etc. in nullable columns. This system is used among other things to capture the information of new patients being admitted to a hospital. The admissions clerk carefully enters all the information from a form that is completed by the patient or his admitting partner. The form of a specific patient clearly indicates that he is allergic to penicillin. This information is dutifully entered into the system by the admissions clerk.

However an attacker has placed an INSTEAD OF trigger on the additional information table that changes the allergy value to null before executing the original insert. After admission, the medical system is then used to print the patient's chart. A medical doctor then orders the administration of penicillin as part of a treatment plan after consulting the chart, which indicates no allergies. This action ultimately leads to the death of the patient due to an allergic reaction. An investigation is performed to determine the liability of the hospital after the cause of death has been established. The investigation finds that the allergy was disclosed on the admissions form, but not entered into the medical system. The admissions clerk that entered the information of the patient that died is determined and questioned. The admissions clerk however insists that he did enter the allergy information on the form and the system indicated that the entry was successful. However, without any proof substantiating this, the admissions clerk will be found negligent.

Depending on the logging performed by the particular database, there might be no record in the database that can prove that the admissions clerk was not negligent. The application used to capture the information might however contain a log that shows a disparity between the data captured and the data stored. Without such a log there will possibly be only evidence to the contrary, implying gross negligence on the part of the admissions clerk. This could ultimately lead to the admissions clerk being charged with having performed an act of omission. However, should triggers be examined as part of a forensic investigation, they could provide a different perspective. In this example the presence of the trigger can as a minimum cast doubts on the evidence and possibly provide actual evidence to confirm the version of events as related by the admissions clerk.

The next example shows commission actions by using a trigger to implement the salami attack technique. An insurance company pays its brokers commission for each active policy they have sold. The commission amount is calculated according to some formula and the result stored in a commission table with five decimal precision. At the end of the month, a payment process adds all the individual commission amounts together per broker and stores the total amount rounded to two decimals in a payment table. The data from the payment table is then used to create payment instructions for the bank.

Now an attacker could add an INSTEAD OF trigger on the insert/update/delete operations of the commission table which would get executed instead of the insert/update/delete operation that was requested. In the trigger, the attacker could truncate the commission amount to two digits, write the truncated portion into the payment table against a dormant broker and the two decimal truncated amount together with the other original values into the commission table. The banking details of the dormant broker would be changed to an account the attacker controlled and the contact information removed or changed to something invalid so that the real broker would not receive any notification of the payment.

When the forensic investigator gets called in after the fraudulent bank instruction gets discovered, he will find either of two scenarios: The company has an application that uses database user accounts for authentication or an application that has its own built-in authentication mechanism and uses a single database account for all database connections. In the first case, he will discover from the audit logs that possibly all users that have access in the application to manage broker commissions, have at some point updated the fraudulent bank instruction. Surely not all employees are working together to defraud the company. In the second case, the audit logs will attribute all updates to the fraudulent bank instruction to the single account the application uses.

In both cases it would now be worthwhile to query the data dictionary for any triggers that have content that directly or indirectly refers to the payment table. Both Oracle and SQL Server have audit tables that log trigger events. If the trigger events correlate with the updates of the payment table as indicated in the log files, the investigator will have proof that the trigger in fact performed the fraudulent payment instruction updates. He can now move on to determine when and by whom the trigger was created. Should no trigger be found, the investigator can move on to examining the application and its interaction with the database.

Another more prevalent crime that gets a lot of media attention is the stealing of banking details of customers of large companies [18]. The most frequent approach is the breach of the IT infrastructure of the company and the large scale download of customer information including banking details. This normally takes place as a single big operation that gets discovered soon afterwards. A more stealthy approach would be the continuous leaking of small amounts of customer information over a long period.

Triggers could be used quite easily to achieve that at the insurance company in our previous example. The attacker can add an AFTER trigger on the insert/update operations of the banking details table. The trigger takes the new or updated banking information and writes it to another table. There might already be such a trigger on the banking details table for auditing purposes and so the attacker simply has to add his part. To prevent any object count auditing picking up his activities, the attacker can use an existing unused table. There is a good chance he will find such a table, because there are always features of the application that the database was designed to have, that simply were not implemented and might never be. This is due to the nature of the dynamic business environment the companies operate in.

Suppose every evening a scheduled task runs that takes all the information stored in the table, puts it in an email and clears the table. There is a possibility that some form of email notification method has already been setup for the database administrator's own auditing process. The attacker simply needs to piggy back on this process and as long as he maintains the same conventions, it will not stand out from the other audit process. Otherwise, he can invoke operating system commands from the trigger to transmit the information to the outside. He can connect directly to a server on the Internet and upload the information if the database server has Internet connectivity. Otherwise, he can use the email infrastructure of the company to email the information to a mailbox he controls.

The forensic analyst that investigates this data theft will find the same two scenarios as in the previous example. The audit information will point to either of the following: All the staff members are stealing the banking information together or somebody is using the business application to steal the banking details with a malicious piece of functionality. Only by investigating triggers and any interaction with the table that contains the banking information, will he be able to identify the correct party responsible for the data leak.

The actual breach of the IT infrastructure and the subsequent manipulation of the database could have happened weeks or months ago. This creates a problem for the forensic investigator that tries to establish who compromised the database. Some log files he would normally use might no longer be available on the system because they have been archived due to space constraints. If the compromise was very far back, some archives might also no longer be available because for example the backup tapes have already been rotated through and reused. The fact that a trigger was used in this example is very useful to the forensic investigator. The creation date and time of a trigger can give him a possible beginning for the timeline and more importantly the time window in which the IT infrastructure breach occurred. He can then use the log information he can still get for that time window to determine who is responsible for the data theft.

5. CONCLUSION

Two concerns were raised around the presence of database triggers during forensic investigations. Can triggers cause the contamination of the data being analysed and can the actions performed by triggers be correctly identified and attributed without analysing triggers?

A contribution of this paper is a thorough survey of all trigger types found in the most widely used relational databases. The research found that database triggers are generally defined to perform actions based on changes in the database, be it on the data level or the data definition level. This will normally not affect the work of a forensic analyst, since he is primarily viewing information (be it data or metadata) without making any changes.

5.1 Results

In contrast, the research also showed that some DBMS's allow triggers to be set on the accessing of information. If the forensic analyst works with an Oracle or SQL Server database, he needs to consider the non-data triggers. He should take great care in how he connects to the database to prevent unintended changes from happening and thus potentially having to do time consuming reconstruction to get back to the initial state of the database.

Furthermore, the research demonstrated that triggers can be used to facilitate malicious actions on the back of normal application or operational actions on the database. These changes would be executed in the context of the initial change and the standard audit material would attribute all changes to the same user. It is therefore necessary to examine database triggers as part of the forensic interpretation and attribution processes. All types of triggers should be examined for out of the ordinary and suspicious actions that relate to the compromised data. This is needed to separate the user actions from the automatic trigger actions.

5.2 Future Work

The current research being conducted is focused on determining how to best analyse the different kinds of triggers. A database under investigation may contain several triggers. Many of those triggers, if not all of them, may bear no relevance to the investigation. So a possible starting point would be the ability to identify if any of those triggers played a part in the specific data being analysed. This can be accomplished by searching the content of all triggers for the occurrence of database objects that are being analysed. A paper proposing an algorithmic approach to achieve this is to be presented at the 2015 IFIP Working Group (WG) 11.9 conference on Digital Forensics [31].

Attention also needs to be given to the fact that some DMBSs allow the obfuscation of the trigger content. This would make it difficult to determine what actions a specific trigger performs and what database operations would initiate them. It also makes the searching of the content for database objects impossible. However, some Oracle and SQL Server database versions have obfuscation weaknesses that make it possible to retrieve the clear text content from an obfuscated trigger.

Also further research needs to be conducted to determine how to best perform forensic acquisition and analysis when the database being investigated supports login triggers. Since the login trigger is non-standard, the implementations will differ between different databases. Hence it will not be easy or even possible to establish a common process. Any process that can be followed to neutralise or circumvent any potentially interfering logon triggers would be very database specific.

An aspect that has not been addressed in this paper is what impact triggers have when the forensic investigator does make intentional changes on a copy of the data. The investigator could be testing a hypothesis, performing data reduction, reconstructing deleted data or simply be storing his results in a temporary table.

6. REFERENCES

[1] M.M. Pollitt: "An Ad Hoc Review of Digital Forensic Models", *Proceedings of the Second International Workshop on Systematic Approaches to Digital Forensic Engineering*, Seattle, pp. 43-54, April 2007.

[2] F. Cohen: *Digital Forensic Evidence Examination*, Fred Cohen & Associates, Livermore, CA., 4th edition, Chapter 2, p. 45, 2009.

[3] M.S. Olivier: "On metadata context in Database Forensics", *Digital Investigation*, Vol. 5 No. 3-4, pp. 115–123, 2009.

[4] H.K. Khanuja and D.S. Adane: "A framework for database forensic analysis", *Computer Science & Engineering*, Vol. 2 No. 3, June 2012.

[5] ISO/IEC 9075-2, "Information technology - Database languages - SQL - Part 2:Foundation (SQL/Foundation) ", 2011.

[6] C.R. Turner, A. Fuggetta, L. Lavazza and A.L. Wolf: "A conceptual basis for feature engineering", *The Journal of Systems and Software*, Vol. 49 No. 1, pp. 3-15, 1999.

[7] "DB-Engines Ranking of Relational DBMS". Internet: http://db-engines.com/en/ranking/relational+dbms, [1 May 2014].

[8] "CREATE TRIGGER Statement", Oracle® Database PL/SQL Language Reference 11g Release 2 (11.2). Internet: http://docs.oracle.com/cd/E11882_01/appdev.112/e17126/create_trigger.htm, [2 May 2014].

[9] "CREATE TRIGGER", Data Definition Language (DDL) Statements. Internet: http://msdn.microsoft.com/en-us/library/ms189799.aspx, [2 May 2014].

[10] "CREATE TRIGGER Syntax", MySQL 5.7 Reference Manual. Internet: http://dev.mysql.com/doc/refman/5.7/en/create-trigger.html, [2 May 2014].

[11] "CREATE TRIGGER", PostgreSQL 9.3.4 Documentation. Internet: http://www.postgresql.org/docs/9.3/static/sql-createtrigger.html, [2 May 2014].

[12] "CREATE TRIGGER", DB2 reference information. Internet: http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2z10.doc.sqlref/src/tpc/db2z_sql_createtrigger.htm, [2 May 2014].

[13] "Oracle Community". Internet: https://community.oracle.com/community/developer/search.jspa?peopleEnabled=true&userID=&containerType=&container=&q=select+trigger, [2 May 2014].

[14] D. Fabbri, R. Ramamurthy and R. Kaushik: "SELECT triggers for data auditing", *Proceedings of the 29th International Conference on Data Engineering*, Brisbane, pp. 1141-1152, April 2013.

[15] "Logon Triggers", Database Engine Instances (SQL Server). Internet: http://technet.microsoft.com/en-us/library/bb326598.aspx, [2 May 2014].

[16] "PL/SQL Triggers", Oracle® Database PL/SQL Language Reference 11g Release 2 (11.2). Internet: http://docs.oracle.com/cd/E11882_01/appdev.112/e17126/triggers.htm, [2 May 2014].

[17] A. Kornbrust: "Database rootkits", *Black Hat Europe*, April 2005. Internet: http://www.red-database-security.com/wp/db_rootkits_us.pdf, [1 May 2014].

[18] C. Osborne: "How hackers stole millions of credit card records from Target", *ZDNet* (13 February 2014). Internet: http://www.zdnet.com/how-hackers-stole-millions-of-credit-card-records-from-target-7000026299/, [5 May 2014].

[19] F. Adelstein: "Live forensics: diagnosing your system without killing it first", *Communications of the ACM*, Vol. 49 No. 2, pp. 63-66, February 2006.

[20] B.D. Carrier: "Risks of live digital forensic analysis", *Communications of the ACM*, Vol. 49 No. 2, pp. 56-61, February 2006.

[21] C. Hargreaves and H. Chivers: "Recovery of Encryption Keys from Memory Using a Linear Scan", *Proceedings of the Third International Conference on Availability, Reliability and Security*, Barcelona, pp. 1369-1376, March 2008.

[22] M. Lee and G. Bieker: *Mastering SQL Server 2008*, Wiley Publishing Inc., Indianapolis, Indiana, Chapter 2, pp. 48-51, 2009.

[23] K. Fowler: "A real world scenario of a SQL Server 2005 database forensics investigation", *Black Hat USA*, 2007. Internet: https://www.blackhat.com/presentations/bh-usa-07/Fowler/Whitepaper/bh-usa-07-fowler-WP.pdf, [3 July 2014].

[24] S.L. Garfinkel: "Digital forensics research: The next 10 years", *Digital Investigation*, Vol. 7 Supplement, pp. S64-S73, 2010.

[25] "Create Trigger", Adaptive Server Enterprise 15.7: Reference Manual - Commands. Internet: http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc36272.1570/html/commands/X19955.htm, [1 September 2014].

[26] "CREATE TRIGGER", SQL As Understood By SQLite. Internet: http://www.sqlite.org/lang_createtrigger.html, [1 September 2014].

[27] "Login triggers in ASE 12.5+", Login triggers. Internet: http://www.sypron.nl/logtrig.html, [1 September 2014].

[28] "Appropriate Uses For SQLite", Categorical Index Of SQLite Documents. Internet: http://www.sqlite.org/whentouse.html [1 September 2014].

[29] "Well-Known Users of SQLite", Categorical Index Of SQLite Documents. Internet: http://www.sqlite.org/famous.html [1 September 2014].

[30] W.K. Hauger and M.S. Olivier: "The role of triggers in database forensics", *Proceedings of the 2014 Information Security for South Africa Conference*, Johannesburg, August 2014.

[31] W.K. Hauger and M.S. Olivier: "Determining trigger involvement during forensic attribution in databases", *2015 IFIP Working Group 11.9 Conference on Digital Forensics*. Accepted for presentation.