Assessing SQL Server User Vulnerability and Data Loss

Sergio G. Medina Ríos
Master in Computer Science
Prof. Jeffrey Duffany, Ph.D.
Electrical & Computer Engineering and Computer Science Department
Polytechnic University of Puerto Rico

Abstract *— By using the tools provided by SQL server we are going to verify some parameters on the database user accounts to determine their vulnerabilities. Run scripts to check if the users have their account passwords set to expire or if they have not changed their password in a while since this is a security vulnerability. We want to prevent a malicious user or a hacker to access out data and compromise information and suffer data loss. In this project we are going to develop a trigger that checks whether the user account expiration has been unchanged in a long time and user built-in tools like reporting services to help the system administrator to disable accounts that haven't change their password or notify if the user still active. Also, we are going to assess the recovery of data that has been erased at a row level by making use of the transaction log.*

Key Terms *— Account Vulnerability, Backup, MS-SQL, Privacy.*

INTRODUCTION

This paper presents two issues that may arise during the use of a database in a network environment, as a DBA by myself I have encountered different issues that have affected the performance of my databases, from this experience I have learned the importance of enhancing our database security.

Database security involves hackers, viruses, software vulnerabilities and lack of safety awareness and other aspects. This requires us to strengthen database security awareness and master certain security technologies. First, the database security threats, database security refers to the protection of the database to prevent illegal use of information leakage caused by alteration or destruction, the most common are:[1] (a) internal staff error, this is the most common potential risk, the most common cases include: careless operation due to accidental deletion or data leakage. Although this is not a hostile act, it is clear that this behavior will cause some unexpected data risk. (B) Social engineering techniques used by attackers because legitimate users unknowingly pass security secrets to the attacker. In this case, the user may be compromised by a website and other ways to provide information to seemingly legitimate requests, like phishing attacks. (C) The insider attacks many database attacks originate from inside of the company. Wages and interpersonal conflicts are likely to lead to dissatisfaction of employees, resulting in increased insider attacks. These internal personnel attacks that have been driven by the desire for revenge or greed, and are not affected by firewalls and intrusion prevention systems are the most dangerous attack. (d) Misconfiguration, a hacker could take advantage of the security holes like having a guest account enabled, thus bypassing the authentication method and access to sensitive information. It is most common when the people in charge of configuring the systems did not know what to do or are working with configurations beyond their knowledge. (E) An unpatched vulnerability, attacks have new evolved from public exploits to finer approach and constantly challenge traditional intrusion detection mechanisms. It is one of the hardest to manage because of the zero-day attacks. The attacker can immediately exploit code and have a fully open the door to a database.

An important part of any security solution is for accountability and compliance considerations and the ability to audit. Auditing a database is not enabled by default in SQL Server, however, you can configure to monitor certain types of activities and have a readable log that we can access to analyze or

recover data. In this project, we are going to develop triggers to do auditing tasks.

Related Research

A database is a system for the retrieval and storage of information in an easy way. The security vulnerabilities occur mainly thru loopholes as a consequence of a flaw in the system design of either the hardware or the software allowing the access of information by unauthorized users. Data breach may be conducted either by unauthorized external users, authorized or unauthorized internal users. The databases more prone to be attacked are those that are exposed outside their local networks especially those that can be accessed via a web page which can be attacked by the use of an exploit called SQL injection. A way to determine the vulnerabilities of a DB is by conducting a DB Vulnerability Analysis, this check the can be done in five steps:

1. Information Gathering – Check the database and the network where it runs to identify information assets.
2. Penetration Testing – Conduct test to determine the possibility of the database being compromised (example SQL Injection), as you can see in the graphic below here are the most common SQL injection threats:

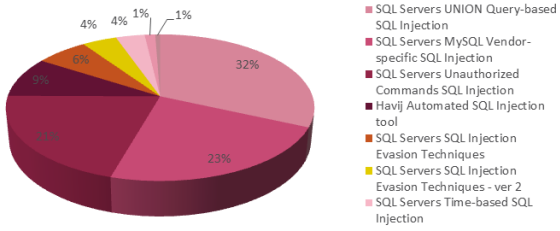


Figure 1
SQL Injection Logs Ratio from One Monitored Network in May 2015 [2]

3. Security Auditing – Conduct an internal security monitoring to verify the effect of the penetration.
4. Fix scripts or patch the software – Normally the security holes occur by the wrong use of stored procedures that have unnecessarily elevated privileges, also by unpatched systems. Robert Sheldon in his article "How to Get SQL Server Security Horribly Wrong"[3] mentions the 2003 Slammer computer worm that affected 75000 servers with SQL Server and within 10 minutes of deployment the worm infected over 90% of the vulnerable server and took down thousands of databases. It is the perfect example of what could happen if we don't patch our systems often. For this we can create a patch testing computer, it can either be physical or virtual although a VM is recommended, we can test the patched on that computer that is an exact copy of the production one and test the patched before going live.
5. Report the findings.

SQL Server User Accounts Expiration

We can increase the security by using a password policy for users that access the database by using SQL Server authentication. The script developed in this project aim to further increase the security by allowing the administrator to easily identify the security status of an account, if an account has expired password or if a user that is not currently with the company and that account still active it could be prone to be exploited by an attacker. As normally end users do not understand the implications of having their passwords compromised, we have to create tools to help the assessment of those issues.

To view the user account and the expiration of the password we must first the account login properties, we can do this by first connecting to the server as an administrator and then check the user information in the security login properties from there we can enforce password policy and password expiration.

Our script will check the users for their expirations status, the expire an occur if a user does not change the password for a long time, if there is an expired account we will send a report to the system administrator to prevent and resolve security vulnerabilities. We also verify how long has passed since the last password change by using the modified

date field from the SQL_Logins table from the Master database which is the main database on SQL server, this database stores information regarding the configuration of SQL Server such as users, maintenance plans, default stored procedures etc.

By analyzing the data we can see two major problems, most of the users don't have the password expiration enable so the system will never force the user to change their password, also if there is a password policy where the user has to remember to change their password by themselves, it's clearly not working as we can see there are users that has not changed their password in over 80 days which is well above the recommended 45 days. Also the users with hashtags (##) before and after their usernames are used only for internal use, and there are "hidden" accounts used by the system. After verifying the days passed since the last change to all users I gathered the following data displayed in **Figure 3.**

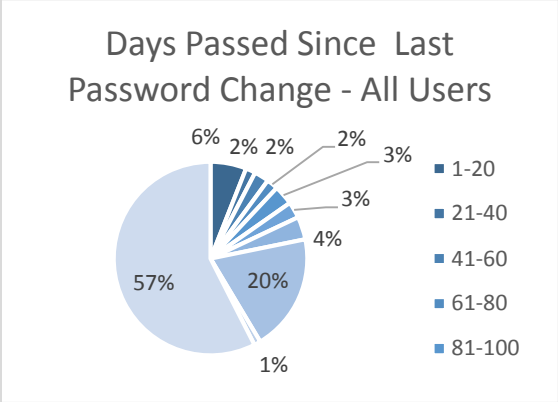


Figure 2
Distribution of Password Longevity

As we can see from this graphic is that the majority of the user have not changed their passwords in more than 6 months which is clearly a huge vulnerability problem, also only 12 accounts are disabled which it becomes a matter of investigation by the administrator to know which of these users are still with the company as there might be employees that are not working anymore, and the human resources department has not notified that information to the administrator yet.

With this we now make a report that will send this information to the administrator in order to verify if the users still with the company and if still working send a warning message to the user, also, this report helps the administrator to enhance the security of their users by activating expiration and password policy so this way the possibilities of the account being compromised be lower. For this, we use the Microsoft Reporting Services report generator platform that comes built in with SQL Server.

Data Loss Recovery

We have seen how to assess some of the most common issues regarding the user security, but what about how to manage the issue after the fact?, in this section we are going to talk about recovery of information once the attack has happened and what steps should we take into consideration in order to monitor unusual changes to the database.

In this paper we are going to concentrate on MS SQL Server but this concept is similar to other database technologies, I compared SQL Server with other types of DB technologies (MySQL and SQL Lite) and it share some important behaviors in where the information stored is not physically removed but instead is logically removed.

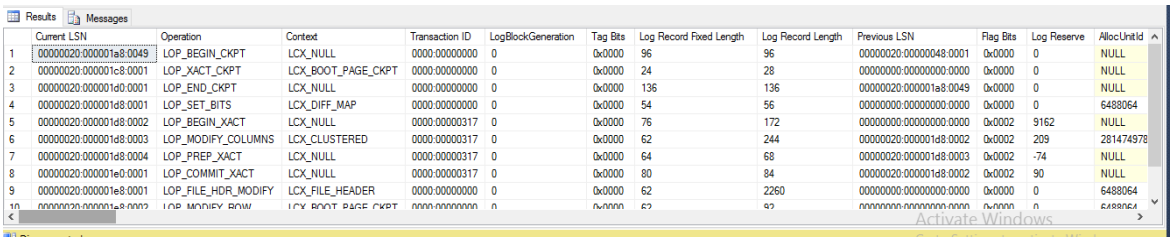
Full Recovery Mode

There are two types of set of recovery on SQL Server, simple mode y full mode, under the simple recovery mode there will be the risk of loss from the last backup was made. The full backup mode the risk of data loss is greatly reduced. We are going to concentrate on full backup mode.

In Full Recovery mode, any changes will be recorded in the database log file to give maximum protection. In full recovery mode, we may also make data recovery to any point in time within the range of the log.

In the simple recovery mode, the log is almost not managed. Every CheckPoint is likely to truncate the log so that inactive VLF (Virtual Log File) removed to reuse space. Therefore, in the simple recovery model, log space usage is almost not considered. In contrast, in full recovery mode, the log data an important part of the restoration.

In full recovery mode, CheckPoint will not truncate the log. Only backup logs will push back the



Results | Messages

	Current LSN	Operation	Context	Transaction ID	LogBlockGeneration	Tag Bits	Log Record Fixed Length	Log Record Length	Previous LSN	Flag Bits	Log Reserve	AllocUnitId
1	00000020:000001a8:0049	LOP_BEGIN_CKPT	LCX_NULL	0000:00000000	0	0x0000	96	96	00000020:00000048:0001	0x0000	0	NULL
2	00000020:000001c8:0001	LOP_XACT_CKPT	LCX_BOOT_PAGE_CKPT	0000:00000000	0	0x0000	24	28	00000000:00000000:0000	0x0000	0	NULL
3	00000020:000001d0:0001	LOP_END_CKPT	LCX_NULL	0000:00000000	0	0x0000	136	136	00000020:000001a8:0049	0x0000	0	NULL
4	00000020:000001d8:0001	LOP_SET_BITS	LCX_DIFF_MAP	0000:00000000	0	0x0000	54	56	00000000:00000000:0000	0x0000	0	6488064
5	00000020:000001d8:0002	LOP_BEGIN_XACT	LCX_NULL	0000:00000317	0	0x0000	76	172	00000000:00000000:0000	0x0002	9162	NULL
6	00000020:000001d8:0003	LOP_MODIFY_COLUMNS	LCX_CLUSTERED	0000:00000317	0	0x0000	62	244	00000020:000001d8:0002	0x0002	209	281474978
7	00000020:000001d8:0004	LOP_PREP_XACT	LCX_NULL	0000:00000317	0	0x0000	64	68	00000020:000001d8:0003	0x0002	-74	NULL
8	00000020:000001e0:0001	LOP_COMMIT_XACT	LCX_NULL	0000:00000317	0	0x0000	80	84	00000020:000001d8:0002	0x0002	90	NULL
9	00000020:000001e8:0001	LOP_FILE_HDR_MODIFY	LCX_FILE_HEADER	0000:00000000	0	0x0000	62	2260	00000000:00000000:0000	0x0000	0	6488064
10	00000020:000001e8:0002	LOP_MODIFY_ROW	LCX_BOOT_PAGE_CKPT	0000:00000000	0	0x0000	62	92	00000000:00000000:0000	0x0000	0	6488064

Figure 3
Example of a Transaction Log

MinLSN and truncate the log. Therefore, a larger volume of business systems, the growth rate of the logs will become big quickly.

The Role of the SQL Server Transaction Log

Each SQL Server database will be by its modified data (insert, update, delete) the order of the corresponding log records to a log file. SQL Server use the Write-Ahead Logging technology to guarantee the performance and durability of the transaction log, in fact, not only SQL Server, basically mainstream relational database include Oracle, MySQL, db2 are used WAL technology. And this technology also greatly reduces the I/O operation. The WAL core idea us that before the data is written to the database it is first written to the transaction log, then in the next checkpoint it will be written to the database, so when a record is deleted it will be marked as a ghost that will tell the database engine to hide it from future queries even though the underlying data still resides within the data page. A garbage cleanup process runs periodically within SLQ Server to physically remove the ghost records within the data page so space can be reused.

The form SQL Server modify data, it will be divided into the following steps:

1. Write "Begin Tran" in the log buffer in SQL Server
2. Write the information you want to modify the log buffer page of SQL Server
3. Write in data pages the SQL Server buffer the data to be modified
4. Write "Commit" to record in the log buffer in SQL Server
5. Place the log buffer to the log file
6. Send a confirmation message (ACK) to the client (SMSS, ODBC, etc.)

The above steps can be seen, even if the transaction has come to commit stage, just log pages are written to the log buffer, and no data is written to the database. It will be when modified data pages are written to the database [4].

The page buffer is written to disk by one of two processes to achieve:

These two processes are [5]:

1. Lazy Writer- Lazy Writer purpose is to manage the buffer. When the buffer reaches a certain critical value, Lazy Writer will store in a disk file in the buffer zone whereas the unmodified pages are released and resources recycled.
2. CheckPoint – The functions of Checkpoint is to reduce the recovery time of the system. CheckPoint like his name, as indicated, is an archive point. CheckPoint occur regularly. The "dirty" page buffer is written to disk. In addition to the automatic CheckPoint, CheckPoint will happen when the Alter DataBase and shut down SQL Server commands are executed.

Any buffer is modified pages will be marked as "dirty" page. Dirty pages are written to disk is CheckPoint or Lazy Writer jobs.

To illustrate some important fields (See Fig. 3):

- CurrentLSN: Current LSN number, the transaction log for each record by a unique log sequence number (LSN).
- Operation: Operation current LSN made.
- Context: Context operation.
- TransactoinID: Transaction ID Number.
- The share of fixed-length record LSN virtual log file: Log Record Fixed Length.
- Previous LSN: previous LSN number.

- AllocUnitID: modified piece of data relevant to the allocation unit ID.
- AllocUnitName: Modify the table data.
- Slot ID: data where the first few pages of data recording.

In my times as a DBA this was one of the first issues I encountered as the company I work had a problem with the transaction log that it grew so big that the hard drive that hold the TLog became full and the DB crashed, the size of the TLog had grown up to 80GB, the problem was that they were not performing the log backup correctly

Therefore, the purpose of the log backup into the following two:

- Reduce the size of the activity log
- Reduce the risk of damage to the log

You can see from the chart by the excerpt from the MSDN:

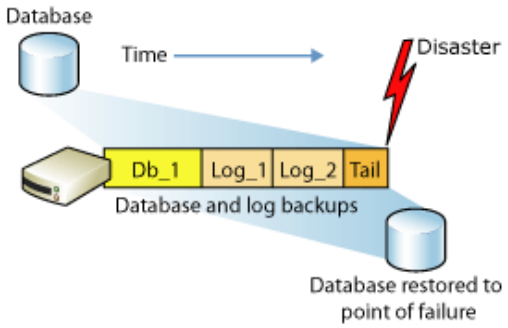


Figure 4
Backup Strategy under Full Recovery Model

In this case, DB_1 has a complete backup, and there is two log backup (Log_1 and Log_2), soon after the backup of Log_2 a loss of data happens. Then if the log file is intact, you can back up the active log (Tail of log), began to recover from DB_1, followed by recovery Log_1, Log_2, and then restore the database to a point in time when a disaster occurs. Theoretically the data loss is zero. The difference to recover from a full or differential backup is great as normally the full backups occur every 24 hours if the DBA do it like is recommended [6].

The transaction log is a file of extension .ldf and it contains all the data necessary to recover the database. From the inside of this there are virtual log files (VLF), this is a place of separation on the TLOG, it happens when the transactions are not active, when this occurs, and the log file does not contain any more active transactions then is truncated to make up space for new transactions. It happens because SQL Server storage engine works this way so that the transaction log management is more effective. VLFS number and size cannot be set by the configuration, but is managed by SQL Server. When the Create or Alter database, SQL Server through ldf file size determine the size and number of VLFS. When the log file grows, SQL Server will be re-planning of the number of VLFS.

Note: According to this principle is not difficult to read, if you set the log file is too small increments, it will produce too much VLFS, which is the log file fragmentation, excessive log file fragmentation will hit SQL Server performance.

The segments of the VLF can be divided into the following four categories:

1. Active- VLF stored on any LSN that is active, the VLF was active, even if a tiny part of the VLF contains an LSN.
2. Recoverable (Recoverable) - VLF is inactive, the VLF does not include activities LSN but has not yet been cut off (truncated).
3. Reusable (Reusable) - VLF is inactive, the VLF does not include activities LSN, it has been cut off (truncated), can be reused.
4. Unused (Unused) - VLF is inactive and has not been used.

Logical Organizational Structure of the Transaction Log

Before saving any modifications made to the database objects in the database, the corresponding log will first be recorded in the log file. This record will be recorded in accordance with the order of logic to the end of the log file, and assign a globally unique log sequence number (log sequence number, referred LSN), the serial number is entirely in accordance with the order, and if the log sequence number two LSN2> LSN1, then the occurrence of after LSN2 where LSN1.

It can be seen from consideration in the log file is divided into a plurality of files in addition to disk space. Image data that cannot be completely parallel access, so the log file is divided into a plurality will not have performance improvement.

LSN number can be seen as a link to a log file and record data between the LSN of each log is not only numbers, as well as the corresponding transaction log [7].

Many types of operations are recorded in the transaction log. These operations include:

- The beginning and end of each transaction.
- Every data modification (insert, update, or delete). It includes system stored procedures or data definition language (DDL) statements made to any table, including system tables, including changes.
- Each allocation or release area and page.
- Create or delete a table or index.

If we need to restore data we can simply restore the main database and then restore the transactions logs, however for the purpose of this project we are going to restore specifics rows from the transaction log. The purpose of this could be for forensic investigation or to restore a very important transaction on the database that was there at a certain point in time without changing anything else in the current system but before we restore those rows we need to know when the row was changed or deleted and for this we need to have the necessary auditing tools available, we can use the SQL Server auditing features but this may have an impact in the server performance depending the configuration, for the purpose of this project we analyze the need of a company who needed to record all the payment transactions that have been deleted, this deletion can occur if the user made a mistake with the transaction and needed to redo it again but also it could mean that the transaction was in purpose deleted, in order to catch the data in the moment of the transaction occurs we made an Trigger that activates when a specific action is done against the table, in this case a delete, in where we capture the user, the transaction number and the date inserting that information in a user defined audit-table.

With this information, we can proceed to recover the data from that point in time. NOTE: It is important to note that this process will not work if we don't have the exact date and time of when the record was modified, if we don't have this data, recovery at row level will not be possible, that is why is very important to have the auditing tools like the ones built-in on SQL Server or the custom made like the one on this experiment also this is having into account that the DB is backed up every 24 hours.

Now, we are going to backup the database when nothing was stored:

```
backup database Database_Demo
to disk='C:\bkp\Database_Demo.bak'
```

Then, we backup the log file of current state.

```
backup log Database_Demo
to disk='C:\bkp\Database_Demo_LOG1'
```

Then, we disable database access to network users by setting up in Single Mode.

```
ALTER DATABASE Database_Demo
SET SINGLE_USER WITH
ROLLBACK immediate
```

Then, we restore the last full backup of the database before the change was made.

```
Restore database Database_Demo
from disk='C:\bkp\Database_Demo.bak'
with replace,norecovery;
```

Then, we restore the Transaction log of the date the change was made, this script will tell SQL Server to restore the entire TLog backup without the modified transaction so this way the change is not committed to the DB.

```
Restore log Database_Demo
from disk='C:\bkp\Database_Demo_LOG1'
with recovery, stopat = 'Sep 11, 2015 05:20:00 PM'
go
```

Finally, depending on how far back the transactions were modified we have to restore each subsequent TLog to the present time.

This procedure helps companies restore critical information without affecting other transactions.

CONCLUSION

Database security is the main focus of a DB Administrator to prevent leakage of information from and entity or from an individual user. Database security vulnerabilities are mostly because of human error or software bugs. If a malicious user gets access to that information, it can cause great harm to the company. The user account expiration vulnerability check made in this project helps to reduce one these vulnerabilities. These features can be further strengthened and help effectively protect the information of the user by preventing one of the main end user issues in a computer environment that is changing their passwords.

This paper analyze the logical and physical architecture of the transaction log as a way to understand how to use SQL Server log to ensure the basic durability and data backup and recovery. We also introduced the CheckPoint and Lazy Writer, for the understanding of these concepts is fundamental to understanding SQL Server DBA work.

We have proven that we can recover information in a secure way after being deleted at a row level as long as we have the dated of the change and the necessary backup files. There is now way to stop data breach or loss, that's why we must have all the tools available to help combat those threats, some companies does not know that they have some of those tools available for free built in the DB, as most of computer systems, there is always a way to recover data that has been compromised in any way if we have an organized backup and restore policy. This way we can reduce the possibilities of losing important information

FUTURE WORK

This project has helped me to understand some concepts of database systems that I did not know, after this I'm planning to make a research about auditing databases using the built-in tools without having a significant impact on the server performance. Also, I will try to replicate the user accounts vulnerabilities by using SQL injection.

REFERENCES

[1] C. Osborne. (2013). *The top ten most common database security vulnerabilities | ZDNet* [Online]. Available: http://www.zdnet.com/article/the-top-ten-most-common-database-security-vulnerabilities/. [Accessed: 5-Oct-2015].

[2] R. Sheldon. (2015). *How to Get SQL Server Security Horribly Wrong, Simple-talk.com* [Online]. Available: https://www.simple-talk.com/sql/database-administration/how-to-get-sql-server-security-horribly-wrong/. [Accessed: 5-Oct-2015].

[3] A. Kaploun. (2015). *The Latest SQL Injection Trends | Check Point Blog* [Online]. Available: http://blog.checkpoint.com/2015/05/07/latest-sql-injection-trends/. [Accessed: 5-Oct-2015].

[4] Technet.microsoft.com. (2015). *SQL Server Transaction Log Architecture and Management* [Online]. Available: https://technet.microsoft.com/en-us/library/jj835093%28v=sql.110%29.aspx. [Accessed: 5-Oct-2015].

[5] Technet.microsoft.com. (2015). *Checkpoints and the Active Portion of the Log* [Online]. Available: https://technet.microsoft.com/en-US/library/ms189573%28v=sql.90%29.aspx. [Accessed: 05-Oct-2015].

[6] Technet.microsoft.com. (2015). *Backup under the full recovery model* [Online]. Available: https://technet.microsoft.com/en-us/library/ms190217%28v=sql.105%29.aspx. [Accessed: 04-Oct-2015].

[7] Technet.microsoft.com. (2015). *Transaction Log Logical Architecture* [Online]. Available: https://technet.microsoft.com/en-us/library/ms180892%28v=sql.105%29.aspx. [Accessed: 03-Oct-2015].