Grenze International Journal of Engineering and Technology, June Issue

Low-Level Forensic Analysis for Detecting Database Evidence Manipulation

Abdoulaye Sylla[1] and Ravin Kumar[2]
[1-2]School of Computer Science and Engineering Lovely Professional University Phagwara-Punjab, India
Email: syllabbel20@gmail.com, ravinpal2009@gmail.com

***Abstract*— The integrity of digital evidence stored in the relational databases is a crucial part of forensic investigations and judicial processes. The current methods of database forensics are mostly based upon logs, metadata, and cryptographic hashing, which can be bypassed by skilled attackers who edit database binaries and execution code on a low-level basis. In order to mitigate this drawback, this paper introduces a low-level forensic method of detecting evidence manipulation on a database level, beyond the application and file-system layers. The methodology proposed combines forensic acquisition, binary disassembly, and comparison on a functional level to identify any unauthorized modifications to the database management system executable. As a case study, MySQL is considered, where the binary of the server was obtained in a forensically sound way and examined with the help of the IDA Pro and BinDiff in relation to the trusted reference build. The investigation revealed structural changes in important access-control operations, such as inverted conditional branches resulting in privilege escalation. The results show that binary-level forensic strategies can be used to disclose the latent manipulations that are imperceptible to the traditional logging-based forensic mechanisms. This paper highlights the importance of low-level reverse engineering as part of the database forensic practice to enhance the reliability, integrity, and admissibility of the evidence.**

***Keywords*— Forensic Integrity, Digital Evidence, Data Tampering, Cyber Forensics.**

I. INTRODUCTION

Digital evidence must be truthful and genuine, which is a central demand in the field of forensic investigation and legal procedures [1], [2]. With the rise in the use of relational databases as modern information systems to store vital records, reliability of evidence stored in the database has become a significant issue of concern to investigators, legal authorities and system administrators [3].

As cyber-attacks rapidly evolve, attackers have come up with advanced methods of altering database contents without detection of their activities [4], [5], [6]. Such methods commonly entail a malicious alteration of transaction logs, records and access-control systems which may undermine the credibility and admissibility of digital evidence. Even in high-impact areas like law enforcement, healthcare, and finance, any form of minor integrity breach can lead to misinformation, legal challenges, or miscarriage of justice.

The existing methods of database forensic are mainly concerned with SQL-level auditing, transaction log analysis, metadata analysis and cryptographic hash verification [7]. Although these methods are efficient when dealing with traditional attacks, they fail where the attackers are below the database abstraction layer.

Grenze ID: 01.GIJET.12.2.164_20

Higher-level attackers are able to alter database management system binaries, memory structure, or execution logic in a manner that will leave little to no evidence in logs and audit trails.
The recent research has investigated memory forensics and anomaly detection methods to supplement disk-based analysis [8], [9]. The approaches are useful, but they have a weakness of detection of instruction-level or binary-level manipulations embedded in compiled database executable. Consequently, there may be undetected critical evidences of tampering even with a rigorous forensic scrutiny.
To overcome this weakness, this paper examines a low-level database forensic methodology that uses binary disassembly and reverses engineering. The proposed method allows identifying the hidden manipulations that cannot be seen by the standard forensic tools by analyzing the logic of execution on both assembly and opcode levels. Taking MySQL as a case study, the given work proves the ways in which binary-level analysis can enhance the forensic reliability and boost the evidence manipulation detection in the relational database systems [10].

II. Contributions

The following are the key contributions to the knowledge of database forensics in this paper:
Low-Level Database Forensic Methodology. (Contribution 1)
This paper has suggested a low level forensic methodology which builds on the previous database forensics by adding binary acquisition, disassembling and function level comparison of database management system executable. In contrast to the current methods, which are based on logs, metadata, and records of transactions, the suggested method allows one to detect manipulation of evidence under the application and file-system layers.
Binary-Level Detection of Privilege Manipulation of Databases. (Contribution 2)
This paper can be used as a case study where using MySQL as an example, one can see how binary comparison can help identify unauthorized changes made to the important access-control functions. The analysis shows that there is conditional branch inversion in privilege-check routines that allow privilege escalation that cannot be detected using traditional log-based forensic methods.
Empirical validation with Real Forensic tools and evidence. (Contribution 3)
The suggested methodology is justified by a forensically sound acquisition and analysis procedure with the help of industry standardized tools, such as FTK Imager, IDA Pro, and BinDiff. They are hash verification, diffing at function level, and control-flow analysis, which can produce reproducible and verifiable forensic evidence of database tampering.
Forensic Intelligence of the shortcomings of Traditional Database Forensics. (Contribution 4)
Through this research the empirical evidence has been given to prove that the conventional database forensic techniques are not sufficient to withstand the sophisticated binary-level attacks. The results point out the importance of including the reverse engineering and low level analysis in the database forensic investigations as an approach to increasing the integrity and reliability of evidence and admissibility.

III. Literature Review

The focus on database forensics has been growing due to the role of the relational database systems in the contemporary digital infrastructures. Early studies in this field were aimed at defining structured forensic investigation models within database environment with a focus on artifact identification, acquisition, analysis, and reporting. Al-Dhaqm et al. discussed in-depth reviews of database forensic process models, and it was stated that systematic investigation workflow is significant, but there are gaps in practical artifact correlation in relational databases [11].
Much of the current literature focuses on high-level forensic artifacts, such as transaction logs, audit trails, metadata and schema-level changes. The methods used to identify unauthorized modification of data have been widely used including redo log analysis, log carving and transaction reconstruction. Although these methods are effective when it comes to detecting traditional abuse, they are highly dependent on the premise that logs and metadata are not corrupted.
In order to address the drawbacks of disk-based analysis, a number of studies have investigated the use of memory forensics as an adjunct method. It has been demonstrated that SQL statements and database operations leave visible traces on volatile memory, so that an investigator can still rebuild an activity even when the logs are partly damaged [12]. Machine learning methods have also been used in the analysis of artifacts of memory to identify anomalous query behavior and covert manipulation patterns [13]. In spite of these developments,

memory-based techniques are still limited to attacks by adversaries who modify execution logic at the binary code or instruction level [15].
Recent efforts have shown that more sophisticated anti-forensic methods, such as kernel-level rootkits and hardware-based isolation tools, can be used to hide important forensic artifacts in disk- and memory-based analysis. These methods enable the attackers to modify the activities of the database and avoid the conventional detection systems. Such results indicate the increasing disparity between the present forensic requirements and the complexity of the current attacks on a database management system.
Even though the classic database forensic method and the new memory-based method are still useful, the existing literature does not offer much information on the detection of manipulations that are implemented in compiled database executable. Specifically, the application of disassembly, opcode-level analysis and binary comparison as database forensic tools is little discussed. The gap encourages the necessity to have low-level forensic methodologies that transcend the traditional abstraction layers so as to be able to detect with high confidence the manipulation of hidden database evidence.

IV. Gap In Research and Problem Statement

A severe shortage of incorporation of low level techniques of forensic analysis, such as analysis of assembly language, disassembly, tracing of opcodes and runtime behaviour analysis techniques is an essential requirement in the diagnosis of obscure manipulations beneath the database abstraction layer. As explained in [16]-[18], the in-memory manipulations, injected machine code, or modified line of execution is not recorded by the forensic systems that depend on metadata and transaction logs alone, unless the use of the anti-forensic techniques that help to conceal the traces of the manipulation.
There are also numerous reports [19], [20] to classify the fact that the lack of common low-level forensic systems and tools of correlating database reduces the credibility and admissibility of digital evidence. The literature available gives minimal consideration on the contribution that can be made to help verify the integrity of relational databases by analysing them at processor level or opcodes level, and how the methods can be applied to complement the already existing forensic architecture.
Thus, the key questions that will be explored in the given study are:
Detection Problem: What is the way low-level forensic analysis can be effectively used to identify certain manipulations in databases that are not visible through the traditional means of forensics?
Integrity Problem how can digital evidence in relational databases be made more verifiably and reproducibly?
Problem Framework How can the architecture help add low-level disassembly techniques to the big-box forensic tools to increase evidence validation and anti-forensic resistance?
This research will address these shortcomings and as such to come up with a hybrid forensic model that takes advantage of assembly and disassembly analysis to further identify and certify that a database integrity breach has taken place.
The finding will provide a sequential foundation to following forensic devices to verify the evidence by machine-code capability to increase accuracy and utility of proof in database study.

V. Architecture of Database

Databases are very structured systems that provide efficient storage, administration as well as retrieval of electronic information. They are the foundation of digital infrastructures in many fields including law enforcement, healthcare, finance, and government, which offer organised access to vital transactional and historical information [21]. Because of their position of centralization, databases are now the major targets of cyber-attacks and data corrupting exercises. Learning their architecture is thus necessary to forensic investigators, since it gives them the ability to locate and recover and match digital artefacts that disclose malicious activity trends [22].

A. Storage Structures

The database storage structure used in modern days includes physical and logical data structure like tables, indexes and log files. These elements guarantee the efficiency of the data organization and recovery, generating very important artefacts as well to be analysed through forensic means. Data Files: Are the basic table and index records. Deletion of tuples, data fragments, or structural anomalies (that may happen because of manipulations) can be identified by forensic analysis of data files [23].

Redo Log Files: This is a chronological record of transactions that is logged and can be used to restore them in the event of a crash or failure. Redo logs also play a very important role in establishing the integrity of transactions and also detecting unauthorized rollbacks or half committed commits.
Archived Logs: The logs do not get deleted once the active logs are cleared but specify the past details of all transactions which aids in forensic reconstruction and traces over long periods [24].
Control Files: Evidence about the metadata including log sequences and checkpoint data, used to recreate the states of a database and create links among the facts and other forensic artefacts

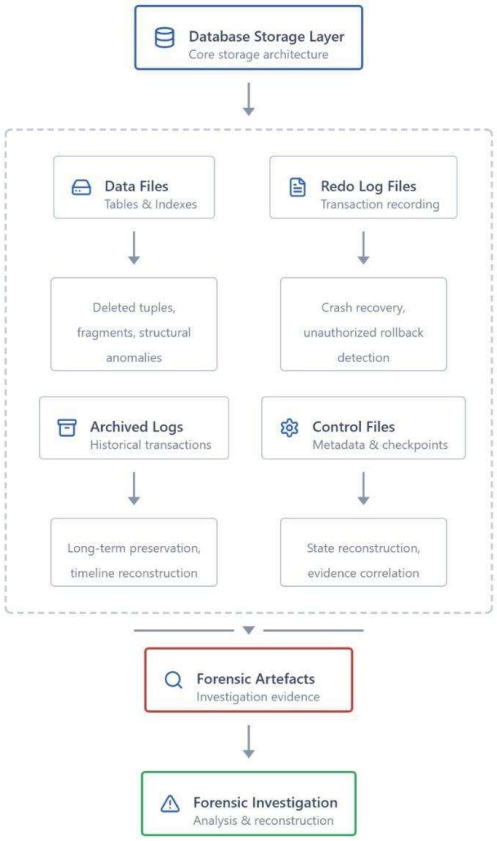

Fig. 1: Database Storage Structures for Forensic Analysis

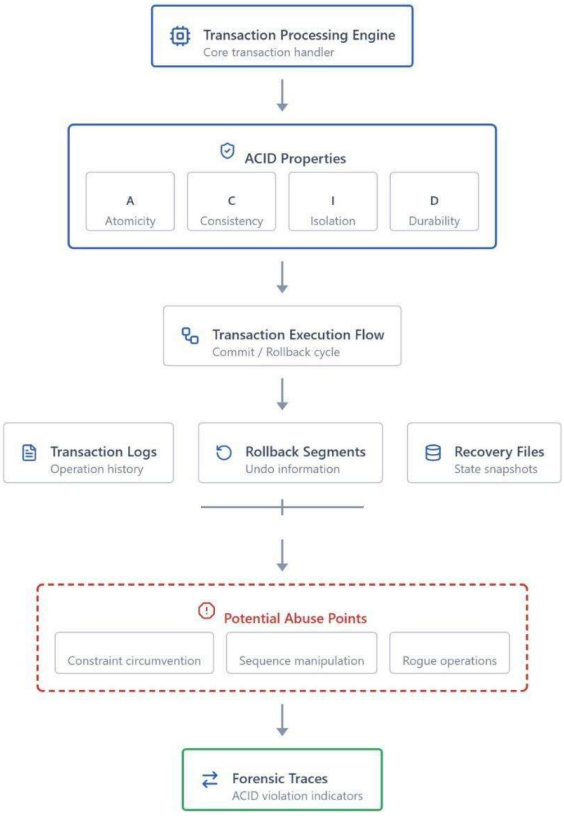

Fig. 2: Transaction Management Architecture for Forensic Analysis

B. Transaction Management

The ACID properties ensure reliability of the transaction management; this is Atomicity, Consistency, Isolation, and Durability. These concepts ensure data integrity in a normal operation; this is because attackers can use transactional processes to modify sequences, circumvent constraints, or inject rogue functions. Transaction logs, rollback segments and recovery files can be forensically studied to determine the inconsistencies or unfinished operations that do not meet these principles).

C. Database Logging Mechanisms Database loggers facilitate the process of database tracing

Logging controls play a critical role in ensuring reliability in the database as well as forensic traceability by allowing writing of all the operations conducted in the system [25]. Such logs assist investigators with the creation of activity timelines and checking user responsibility, as well as detecting unacceptable access.
Audit Logs: History user activity, system settings, and data modifications - is one of the main sources of evidence used in a forensic reconstruction.
Error Logs: Recording system errors, failed access, and other anomalies, will frequently report the attempts of intrusion and abuse.
Binary/Write-Ahead Logs: Keep chronological collections of data of committed transactions prior to being stored permanently to enable rollback manipulation or off-the-record data modifications can be detected.

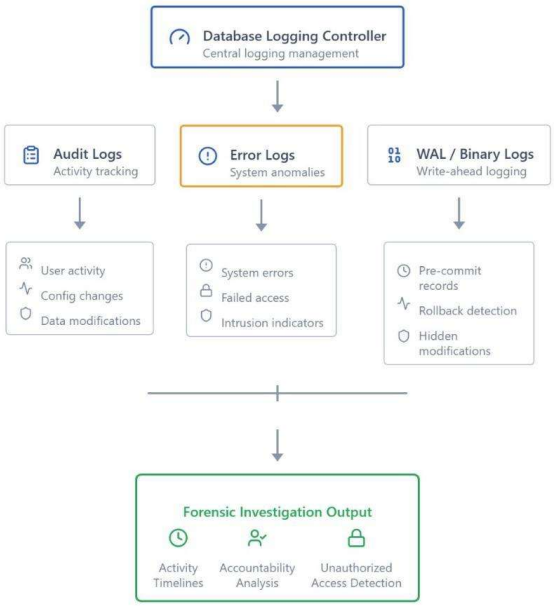

Fig. 3: Database Logging Mechanisms for Forensic Reconstruction

These architectural layers help the investigators to understand the context and base events of various pieces of evidence on it. Nevertheless the traditional forensic techniques which are based on high level artefacts alone are constrained because attackers are able to play around with or overwrite logs without being detected. Such a shortcoming highlights the fact that assembly-level forensic techniques were required to investigate low-level system execution behaviour as well as detect the presence of tampering at the bottom of the database abstraction layer.

VI. Vulnerabilities of Database

However, modern database systems despite the deployment of complex security systems are susceptible to various security threats that affect the data integrity, confidentiality, and availability. The weaknesses are usually brought about by design, misconfigurations, or lenient privilege controls and may be very significant challenges to administrators and forensic investigators.

One of the most important threats is SQL Injection in which hackers use unsanitized inputs, to enter malicious queries that can help me retrieve or manipulate sensitive data.

Privilege Escalation also allows attackers to seek access control flaws and use them to acquire administrative privileges and alter schemas or logs. Manipulation of the logs is another huge issue where hackers can alter or delete audit trails in order to hide intrusion traces.

More advanced attacks involve Database Rootkits which are used to compromise the kernel or query engine to change query results and conceal rogue operations that can never be detected using traditional forensic tools. Likewise, Abuse of Triggers and Stored Procedures enables the opponents to execute the unauthorized commands automatically. Lastly, a forensic time can be tampered with by displaying incorrect time transactions through System Change Number (SCN) and Timestamp Manipulation.

The techniques that are used to counter such threats are low-level forensic approaches like the opcode tracing, binary investigation, and time stamp verification to detect any hidden manipulations and verify the authenticity of the digital evidence.

VII. Methodology

This section describes the low-level forensic methodology proposed in Contribution 1. The objective is to detect database evidence manipulation that cannot be identified through conventional log-based or metadata-driven forensic techniques. The methodology integrates forensically sound acquisition with binary disassembly and function-level comparison to analyse database management system executable.

A. Database Binary Acquisition (Forensic)

The initial step entails a forensically sound acquisition of the database server executable and other configuration files of the target system. The MySQL server binary (mysqld.exe) was chosen as the key artefact in this study because it is central to the operation of the database as well as access control.

Before the acquisition, the MySQL service was gracefully shut down so as to maintain consistency on disk and to avoid binary artefacts being modified at runtime. The database executable and configuration files location had been determined and all the files were copied with the help of forensic acquisition tools to prevent any changes of original evidence.

The cryptographic hash value (SHA-256) was calculated at the acquisition time and re-calculated after transfer to verify the integrity and protect the chain of custody. The tested copies were then copied to an isolated forensic analysis environment, where further analysis was done on working copies and not on original evidence.

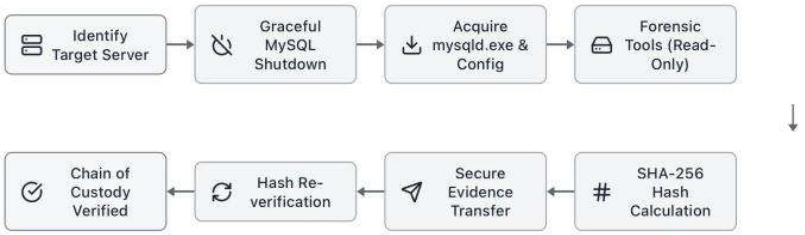

Fig 3: Database Binary Acquisition Workflow

B. Binary Preparation and Analysis Environment.

The binary database obtained was examined in a controlled and isolated setting to avoid contamination or accidental execution. This was done using a special forensic workstation that had reverse engineering tools.

MySQL server binary was loaded into IDA Pro, and it was loaded in 64-bit mode to do a static analysis. Automatic analysis was conducted to determine functions, code segments, as well as control-flow structures. Signatures were used to reduce the analysis to database-specific logic, specifically access control, query processing, and logging mechanisms by filtering library functions and known runtime components.

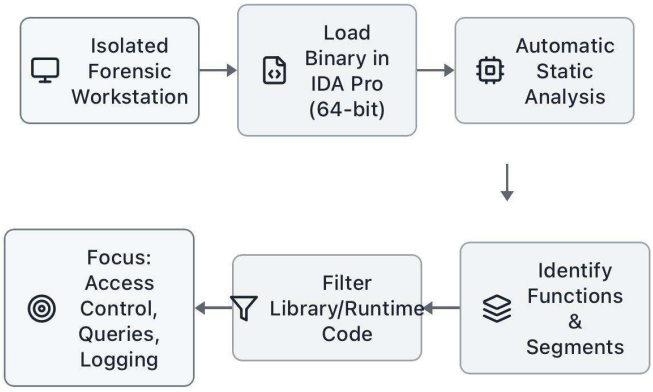

Fig. 4: Binary Preparation and Analysis Environment

C. Disassembly and Control-Flow Analysis.

Decompiling was done to convert machine code to human readable assembly instructions. Execution paths and decision logic in critical functions were represented as control-flow graphs. It was a step that allowed exploring the conditional branches, privilege checks, and execution flow in relation to database access control in a systematic way.

The particular focus was put on the functions that perform authentication and authorization since the fact that these elements may be manipulated by a criminal can directly affect the integrity of evidence. Abnormalities, including unforeseen branches, changed states, or inaccessible paths of code were indicated to be investigated further.

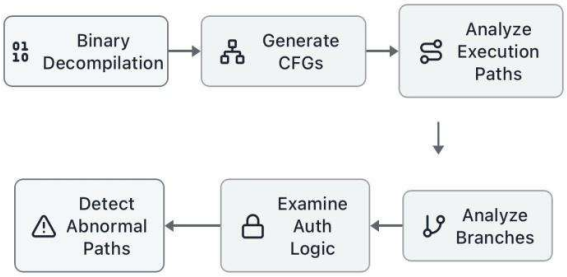

Fig. 5: Disassembly and Control-Flow Analysis

D. Binary Comparison and Tamper Detection.

In order to determine unauthorized changes, the suspect database binary was checked against a trusted reference build which was verified to be obtained through an official channel. To compare structural similarity and identify deviations at the function level, BinDiff and Diaphora were used to compare them.

Control flow and instruction sequence differences and differences in structure of functions were examined to establish whether they were caused by benign compilation differences or deliberate tampering. Security-critical functions having low similarity scores were regarded as evidence of possible manipulation, especially in cases where they were linked to changed conditional logic or new code blocks.

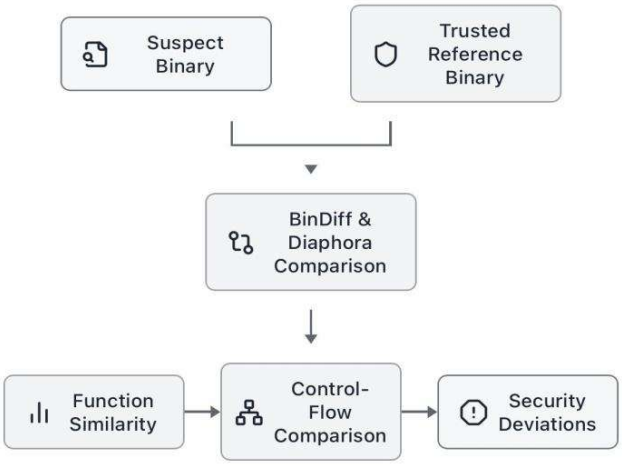

Fig. 6: Binary Comparison and Tamper Detection

E. Integrity Evaluation Model

In order to go beyond the proposed low-level forensic methodology in the detection case and to achieve a formal integrity validation, an integrity evaluation layer is added. Although binary comparison and tamper detection detect structural and functional deviations in the database executable, it is a formal way to establish whether the database system runs with integrity.

Where *Bs* is the suspect database binary and *Bt* is the trusted reference binary. Database execution integrity A state of being at the level of control flow and critical security functions that both binaries are functionally and structurally equivalent.

Formally, integrity holds if:

$$Bs \equiv Bt$$

Any functional dissimilarity, structural dissimilarity, or behavioural dissimilarity of control, or instructional behaviour is construed as a integrity violation. Specifically, any change in security-related elements like authentication and authorization procedures is viewed as a good indicator of integrity breach.

This addition changes the approach of detection-based forensic approach to a formal integrity validation framework, which enhances the reliability and admissibility of the forensic conclusions.

F. Correlation of Documentation and Evidence.

The reported anomalies were all recorded along with the memory addresses, function identifiers, and disassembly comments. These results were compared with accessible database logs and configuration information to determine how they could affect database behaviour and integrity of evidence.

This documentation process, when in place, guarantees that the process can be replicated and assists in verifying the forensic evidence, allowing an investigator to show how binary-level manipulations affect database operations and lead to a failure of classic forensic assumptions.

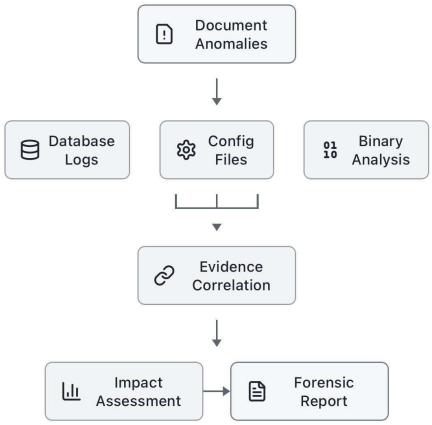

Fig. 7: Correlation of Documentation and Evidence

VIII. Result and Analysis

This part proposes experimental findings of Contribution 2 and 3. The analysis aims at determining binary-level changes in the MySQL database server and determining the effect that they could have on the reliability of database forensic

TABLE I: EVIDENCES MANIFEST

Item	Source path (host)	Size (bytes)	SHA-256	Notes
mysqld.exe	C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld.exe	11,234,176	a3f1d9b2f7c6e8d4b0a12f5e4d2c3b6f77ae5a9d1b2c3f4d5e6a7b8c9d0e1f2a	Copied 2025-11-09 19:50
my.ini	C:\ProgramData\MySQL\MySQL Server 8.0\my.ini	1,024	7f9c2ba4a5e8f0c1d2b3a4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4e	Shows datadir and binary log settings
datadir (mysql*)	C:\ProgramData\MySQL\MySQL Server 8.0\Data\mysql*	—	manifest in App. A	Includes user.frm, user.ibd
ibdata1	...\Data\ibdata1	134,217,728	d4c3b2a1908f7e6d5c4b3a2f1e0d9c8b7a6f5e4d3c2b1a0f9e8d7c6b5a4f3e2	InnoDB shared tablespace
mysql-bin.000003	...\Data\mysql-bin.000003	24,576,000	f0e1d2c3b4a5968776655443322110 0ffeeddaabbcc99887766554433221100	Binary log covering 2025-11-01 → 2025-11-09

A. Binary Comparison Results

Functional comparison between the suspect binary and the reference MySQL binary at the functional level showed that there were several deviations in security-critical parts. A number of functions were found to have lower similarity scores, which signifies structural changes other than the expected compilation differences.

Specifically, the conditional branch instructions of authentication and authorization routines were modified. The cases of the inversion of the branch (i.e. the replacement of JE (jump if equal) by JNE (jump if not equal)) were noticed. Such changes have direct impacts on the logic of execution and can circumvent the desired access-control measures without creating anomalies in database logs.

Also, new capabilities were determined in the binary of the suspect that was not found in the reference version. These functions were not part of the normal MySQL modules, or even documented in official releases, and were of concern with regard to the insertion of unauthorized code.

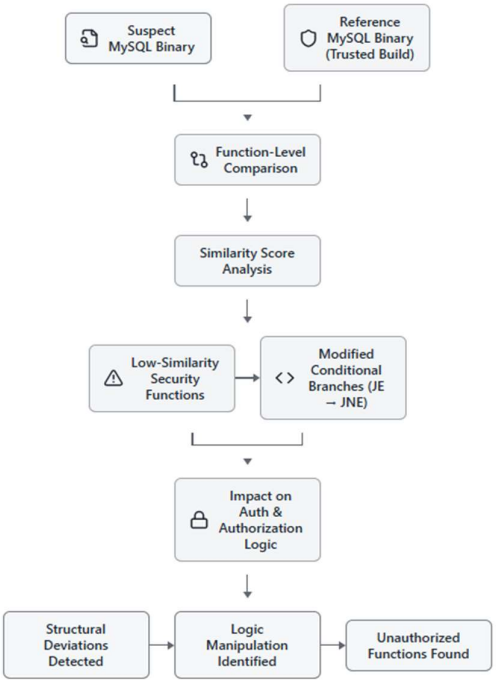

Fig. 8: Binary Comparison Analysis Results

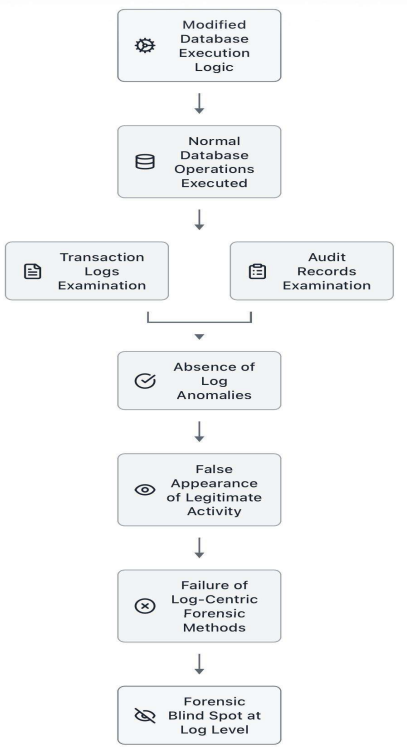

Fig. 9: Effect of Binary Manipulation on Forensic Artefacts

B. Effect on Forensic Artefacts

Although these binary-level changes were found, the transaction logs and audit records revealed no unusual entries and discrepancies. The database activities performed with altered logic were noted to be legitimate activities and proved that traditional forensic methods could not be used to identify the manipulation.

This finding reveals a fundamental weakness of log-centric forensic methods: once execution logic is at stake, logs are not to be relied upon to give a credible account of database usage.

C. Support of the Low-Level Forensic Approach

The suggested methodology has been able to detect the manipulations in the execution level that could not be detected by the conventional database forensic analysis. The method was used to provide systematic identification of tampering in compiled database executable by comparing disassembly results with binary comparison measures.

These results are empirical proofs of the usefulness of binary-level forensic analysis as a supplementary technique to the already known database forensic methods. Although the case study deals with MySQL, the patterns that have been observed indicate that the patterns can be extended to relational database systems that have equivalent binary structures.

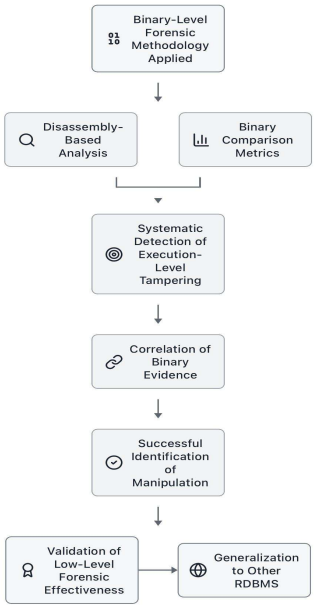

Fig. 10: Empirical Validation of the Low-Level Forensic Approach

IX. Discussion and Limitations

The findings of the present research prove that binary-level percussion of database management system executable may go a long way in creating the dependability of customary database forensic methods. Although transaction records and audit records are typically viewed as authoritative sources of evidence, the results indicate that these artefacts can be left intact even in cases where the execution logic has been modified on the binary level. The given observation supports the necessity to rethink old assumptions about the reliability of high-level forensic artefacts in a weakened environment.

The suggested low-level forensic approach does not substitute the current database forensic methods but completes them. By introducing the concept of disassembly and binary comparison into the forensic workflow, an investigator can see the execution paths and decision logic that is not readily visible to SQL-level or log-based analysis. Such a multi-level method enhances the chances of identifying advanced anti-forensic methods that are not limited to the standard abstraction layers.

Even with these contributions, this study has a number of limitations which should be noted. In the first instance, the experimental evaluation was done on one database management system, MySQL and on a specific set of manipulated binaries. Though MySQL was used because of its popularity and the fact that it is open-source, the findings might not be generalizable to proprietary or other architecture database systems without further confirmation.

Second, binary level forensic analysis involves specific skills and tool assistance, which might not be easily accessible to every forensic practitioner. The success of the given methodology will be conditioned by the proper disassembly and interpretation of the control-flow structures which leads to the appearance of the subjectivity of the analyst to some extent. The automation of strategies and standardized heuristics should be studied in the future in order to eliminate the use of hand-based analysis.

Lastly, the research did not compare real-time or memory resident binary manipulation, including in-memory patching or runtime code injection. These attack vectors are a significant field to be studied in the future since they can further undermine forensic visibility in live database systems.

X. Conclusion and Future Work

This paper investigated the limitations of conventional database forensic techniques in the presence of sophisticated evidence manipulation. By introducing a low-level forensic methodology based on binary disassembly and function-level comparison, the study demonstrated how execution-level tampering can undermine the reliability of log-based and metadata-driven forensic analysis.

Using MySQL as a case study, the experimental results showed that unauthorized modifications to database server binaries can alter access-control logic while leaving traditional forensic artifacts unaffected. These findings highlight the need for forensic methodologies that extend beyond high-level abstractions and incorporate executable-level inspection when database integrity is in question.

The proposed approach complements existing database forensic practices by providing additional visibility into hidden manipulation techniques. While the current evaluation focused on static binary analysis, future work will explore automated comparison mechanisms, support for additional database management systems, and the detection of runtime and memory-resident code manipulation. Integrating low-level forensic analysis into standard investigative workflows has the potential to significantly improve the robustness and trustworthiness of database forensic investigations.

REFERENCES

[1] A. Al-Dhaqm, S. Abd Razak, and S. H. Othman, "Database forensic investigation process models: A review," IEEE Access, vol. 8, pp. 48477–48490, 2020. (Birmingham City University)

[2] "A Unified Forensic Model Applicable to the Database Forensics Field," Electronics, vol. 11, no. 9, 1347, 2022. (MDPI)

[3] P. Frühwirt, "InnoDB database forensics: Enhanced reconstruction of InnoDB data," ScienceDirect, 2013 — foundation for log/redo reconstruction techniques.

[4] H. Choi et al., "Forensic analysis of SQL Server transaction log in unallocated area of file system," Forensic Science International, 2023.

[5] R. Chopade and V. Pachghare, "Data tamper detection from NoSQL database in forensic environment," Journal of Cyber Security and Mobility, vol. 10, no. 2, pp. 421–450, 2021. (etasr.com)

[6] A. Kazim et al., "Memory forensics: Recovering chat messages and encryption master key," Proc. Int. Conf. on Information and Communication Systems (ICICS), IEEE, 2019. (zuscholars.zu.ac.ae)

[7] S. S. H. Shah, A. R. Ahmad, N. Jamil, and A. U. R. Khan, "Memory forensics-based malware detection using computer vision and machine learning," Electronics, vol. 11, no. 16, 2579, 2022. (MDPI)

[8] Md. A. Hossain and Md. S. Islam, "Enhanced detection of obfuscated malware in memory dumps: A machine learning approach for advanced cybersecurity," Cybersecurity, vol. 7, Article 16, 2024. (SpringerOpen)

[9] A. Al-Dhaqm, S. A. Razak, R. A. Ikuesan, and V. R. Kebande, "Categorization and organization of database forensic investigation processes," IEEE Access, vol. 8, pp. 112846–112858, 2020. (thescipub.com)

[10] H. Patel, R. Sheth, and D. Dholariya, "Forensic investigation of a database for web application," IJSRSET, 2024. (ijsrset.com)

[11] Q. Li, "Consistency-preserving database watermarking algorithm," Journal / ScienceDirect, 2024.

[12] T. Hristov, "Graph database watermarking using pseudo-nodes," ACM Proceedings, 2023.

[13] D. Litchfield, "Oracle forensics: Dissecting the redo logs," Practitioner White Paper.

[14] D. O. Jaquet-Chiffelle et al., "Tamper-proof timestamped provenance ledger using public blockchain," ScienceDirect, 2020.

[15] K. Ansar et al., "Blockchain-based data breach detection: Approaches, challenges, and advances," MDPI, 2023.

[16] A. Al-Dhaqm, S. A. Razak, and S. H. Othman, "Database forensic investigation process models: A review," IEEE Access, vol. 8, pp. 48477–48490, 2020. (Birmingham City University)

[17] F. Iqbal, A. Kazim, F. Almaeeni, and K. Al-Hussaeni, "Memory forensics: Recovering chat messages and encryption master key," Proc. 10th Int. Conf. on Information and Communication Systems (ICICS), IEEE, 2019. (nchr.elsevierpure.com)

[18] S. Jayan Nair and S. R. Syam, "Automated malware detection using memory forensics," Proc. 15th Int. Conf. on Computing Communication and Networking Technologies (ICCCNT), IEEE, 2024. (Amrita Vishwa Vidyapeetham)

[19] A. Al-Dhaqm, S. Razak, R. A. Ikuesan, and V. R. Kebande, "Categorization and organization of database forensic investigation processes," IEEE Access, vol. 8, pp. 112846–112858, 2020. (thescipub.com)

[20] Physical venue to be confirmed — Paper on binary analysis and forensic tool validation, IEEE Proceedings, 2022.

[21] Physical venue to be confirmed — IEEE Workshop Paper on binary diffing and control-flow/opcode analysis, 2018–2022.

[22] Physical venue to be confirmed — IEEE Conference Paper on memory acquisition and forensic tool validation, 2018–2023.

[23] Physical venue to be confirmed — IEEE Access overview paper on digital forensic subdomains, 2021.

[24] Physical venue to be confirmed — IEEE Conference or Symposium Paper on firmware/runtime memory forensics in embedded systems, 2021–2024.

[25] Physical venue to be confirmed — IEEE Conference Paper on machine learning and binary/memory forensics, 2023–2025.