A Comparative Analysis of Cyber Vulnerabilities Across Relational, Document, and Cloud Databases

Shibam Karmakar
School of Computing Science and Artificial Intelligence
VIT Bhopal University,
Kothri Kalan, Sehore-466114
shibam.karmakar2020@vitbhopal.ac.in

Dr. Soma Saha
School of Computing Science and Artificial Intelligence
VIT Bhopal University,
Kothri Kalan, Sehore-466114
soma.saha@vitbhopal.ac.in

Dr. Ganeshan R
School of Computing Science and Artificial Intelligence
VIT Bhopal University,
Kothri Kalan, Sehore-466114
ganeshramasamy111@gmail.com

Abstract— - **In this comparative study, we examine the security structures of three prominent database types: relational databases, document databases, and cloud databases. Relational databases, the oldest and most widely used, rely on tables to store structured data, employing security measures like user authentication, authorization, and encryption. Document databases, designed for unstructured data, often used in web applications, employ access control lists (ACLs) and role-based access control (RBAC) to manage data access. Cloud databases, hosted by providers like AWS and Azure, offer scalability and flexibility while implementing standard security features and additional measures like network isolation and data encryption at rest. The study comprises three sections: the first detailing the security structure of relational databases, the second discussing document databases, and the third focusing on cloud databases. The study concludes by comparing the strengths and weaknesses of each database type in terms of security and offering recommendations for enhancing their security measures.**

Keywords— Relational databases, Document databases, Cloud databases, Security structures, Encryption

I. Introduction

Relational databases, document models, and cloud databases are three of the most popular database types in use today. Each type has its own strengths and weaknesses, and the best choice for a particular application will depend on a variety of factors, including the type of data being stored, the required performance and scalability, and the security requirements. The selection of a suitable database type, be it relational, document model, or cloud-based, is contingent upon various factors, encompassing data type, performance demands, scalability requirements, and security prerequisites. While relational databases exhibit a well-established security framework with features like access control, data encryption, and auditing, their complexity in configuration and security vulnerability management can be challenging. Document models, on the other hand, offer simplicity and scalability but may lack certain security attributes such as encryption and auditing, owing to their relative novelty. Cloud databases, whether relational or document-oriented, present advantages like scalability, cost-effectiveness, and streamlined management, yet they introduce fresh security complexities involving data privacy and regulatory adherence. Security analysis entails the systematic identification and evaluation of security risks within a database system, employing techniques such as vulnerability scanning, penetration testing, and risk assessments. Security comparison involves evaluating and contrasting the security attributes and capabilities of different database systems, aiding in the selection of the most appropriate database type for a given application or highlighting areas necessitating security enhancements. In an era defined by the exponential growth of data and the ever-increasing im- portance of safeguarding sensitive information, the choice of a database system has become a critical decision for organizations across the globe. As the technological landscape continues to evolve, two prominent database models have emerged as front-runners in the race for data security and management: relational databases and document model databases. Furthermore, the advent of cloud computing has introduced a new dimension to this equation, as organizations now have the option to host their chosen database systems in the cloud.
In this comparative security study, we embark on an exploration of these two database models—relational and document model—within the context of cloud databases. Our objective is to dissect the intricacies of these systems and their inherent security features, scrutinizing their strengths and vulnerabilities in an effort to assist organizations in making informed decisions that not only align with their data management needs but also fortify their defence against the growing threat landscape of the digital age. By delving into the nuances of each database model and assessing their security within the cloud, we aim to shed light on the most effective and secure data management strategies for contemporary businesses. Relational databases and document databases are two of the most popular database models used today. Relational databases have been the traditional choice for storing and managing structured data, while document databases are becoming increasingly popular for storing and managing semi- structured and unstructured data. Cloud databases are a type of database that is hosted and managed by a cloud computing provider. They offer a number of advantages over traditional on-premises databases, such as scalability, elasticity, and cost savings. Relational databases, document databases, and cloud databases all have different security structures. It is important to understand the security strengths and weaknesses of each type of database when choosing the right database for your application. This comparative study underscores the multifaceted nature of security structures in relational databases, document databases, and cloud databases. Understanding the nuances of their security features is imperative for selecting the most suitable database model based on the security requirements of a given application.

.

II. IMPORTANCE

It is important to compare database models in terms of cybersecurity, cyber-attacks, and cyber threats. This information can help organizations, scholars, and cybersecurity practitioners choose the right database model for their needs, while also understanding the security risks involved. This research is focused on a comprehensive comparative analysis of database models within the context of cybersecurity, aiming to shed light on their distinct security attributes and vulnerabilities. By identifying these vulnerabilities and conducting risk assessments, it seeks to evaluate the potential impact of cyber threats on data integrity, confidentiality, and availability. Additionally, it examines the security mechanisms, encryption methodologies, user access control, and authentication systems inherent to each database model. Assessing compliance alignment, update management, integration with third-party security tools, scalability, performance, resource requirements, adaptability to evolving cyber threats, and real-world case studies are also vital components of this research. Ultimately, the goal is to equip organizations with the knowledge necessary to make well informed decisions regarding their choice of database model and to implement robust security measures for safeguarding their data.

III. Theoretical Study of threats in different model

Threats in Document Model

Threats in Document Models encompass various cyber-attacks that are distinct from those possible in relational databases. One significant threat is Poisoning attacks, where malicious data is injected into the training dataset of the document model, causing it to learn and subsequently generate incorrect or even harmful outputs. This threat is not feasible in relational databases as they do not store data in the same unstructured manner as document models. Evasion attacks are another concern, where attackers create inputs intended to deceive the document model into producing inaccurate results. These are more challenging to execute against relational databases due to their structured nature and rigid data constraints. Inference attacks exploit the document model's ability to discern intricate relationships between data points to deduce sensitive information not explicitly stored within the model, an attack vector unavailable in relational databases. Beyond these specific threats document models, by their very nature, exhibit greater susceptibility to security breaches compared to relational databases. Document models are generally more intricate and less thoroughly understood, making it more challenging to identify and mitigate security vulnerabilities within them. Threats in Relational Model- Threats to the relational model encompass a range of cyber-attacks, some of which are more commonly associated with relational databases and less likely to occur in document databases due to their structured nature. Prominent among these threats is SQL Injection (SQLi), wherein attackers exploit input validation vulnerabilities to inject malicious SQL code, potentially manipulating or extracting data. NoSQL Injection is another concern, although document databases use different query languages. Privilege escalation is more common in relational databases due to their complex access control systems. Schema-based attacks may target the predefined database structure, leading to schema object alterations or deletions. Data manipulation attacks, open executed via SQL queries, are also prevalent. Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) attacks can exploit web applications interacting with relational databases. Brute force and dictionary attacks might target weak or default credentials. Insider threats pose risks when authorized users misuse their privileges. Database misconfigurations, such as open ports or weak authentication methods, can further expose relational databases to potential vulnerability.

IV. General Discussion

Comparing different database model, typically associated with NoSQL systems, and relational databases in the context of cybersecurity entails an evaluation across multiple dimensions. Firstly, in terms of data structure, document databases like MongoDB and CouchDB adopt a semi-structured or unstructured document model (e.g., JSON or BSON), offering flexibility for managing intricate or evolving data structures, whereas relational databases, such as MySQL or PostgreSQL, adhere to structured tables with predefined schemas, affording greater control over data integrity. Secondly, with respect to security features, document databases tend to provide rudimentary security mechanisms, encompassing authentication and authorization, often relying on external safeguards like firewalls and network security tools, whereas relational databases possess well established security features, inclusive of user access controls, role-based permissions, and encryption options, typically offering more robust security measures. The schema flexibility of document databases, while accommodating agility, can pose security risks if data validation and access control are not rigorously administered. Conversely, the rigid schema enforced by relational databases enhances data integrity and mitigates certain threats, such as SQL injection, by enforcing predefined structures. Data validation is chiefly the responsibility of the application in document databases, potentially leading to data integrity lapses if inadequately managed. In relational databases, data validation is an intrinsic element, minimizing the likelihood of storing malformed data. Scaling considerations diverge as document database are commonly linked with horizontal scalability, introducing complexities in security when handling distributed data across multiple nodes, whereas relational databases typically favour vertical scalability, which can be more straight forward to manage but may curtail scaling options. Query languages for document databases, although less expressive than SQL, limit query complexity, reducing the susceptibility to specific attacks, like complex joins. Relational databases employing SQL can be susceptible to SQL injection if not diligently sanitized. Regarding backup and recovery procedures, document databases display variability across specific systems, potentially offering less mature practices compared to traditional relational databases, which are renowned for their established backup and recovery mechanisms, ensuring data resilience. Audit and compliance features in document databases may be limited, necessitating supplementary tools or bespoke solutions to full fill regulatory requirements, while relational databases frequently present more robust audit and compliance capabilities, facilitating regulatory compliance. In conclusion, both document-based and relational databases can be effectively secured, contingent on the particulars of the use case, data prerequisites, and the scope of implementable security measures. Irrespective of the database type, a robust security posture mandates diligent access control, encryption, and periodic security assessments.

V. Detailed Survey

1.Data Tampering- In today's digitally interconnected world, securing sensitive data stored in various database models has become a paramount concern. Data tampering attacks, where unauthorized entities manipulate or alter data, can compromise the integrity of information. This essay explores and compares three situations in which data tampering attacks may occur across different database models: documentoriented databases, relational databases, and cloud databases.

1.1 Document-Oriented Database: One critical vulnerability in document-oriented databases, exemplified by MongoDB, lies in the lack of robust access controls. If proper authentication and authorization mechanisms are not implemented, attackers may exploit vulnerabilities to gain unauthorized access. Once inside, these adversaries can tamper with document data, modifying or even deleting information at will. Furthermore, insecure API endpoints present another avenue for potential attacks. If these communication channels lack encryption or robust authentication, attackers could manipulate data during transit, compromising the integrity of the stored information. The absence of encryption at rest exacerbates the risk, allowing attackers with physical or unauthorized access to directly tamper with stored data.

1.2 Relational Database: In relational databases like MySQL or PostgreSQL, the threat of SQL injection attacks looms large when applications fail to validate and sanitize user inputs adequately. Successful SQL injection attacks enable unauthorized data modifications, including tampering with records and altering the database schema. Weak authentication mechanisms compound this risk, as attackers might exploit compromised credentials to gain unauthorized access. Once inside, these malicious actors can manipulate, modify, or delete critical information. Moreover, insufficient logging and monitoring practices make it challenging to detect unauthorized access promptly, allowing attackers to tamper with data unnoticed until later stages.

1.3 Cloud Database: Cloud databases, reliant on Identity and Access Management (IAM) for access control, face distinct vulnerabilities. Misconfigured IAM policies, such as overly permissive settings expose cloud databases to unauthorized access. Once access is compromised, attackers may tamper with data, alter configurations, or even delete records, posing a significant threat to data integrity. Misconfigured security groups or firewall rules in cloud environments further increase the risk of unauthorized access, providing attackers with opportunities to tamper with data or carry out other malicious activities. Additionally, insufficient encryption during data transmission (in transit) and storage (at rest) can allow attackers to intercept communication or gain access to stored data,. enabling data tampering. In conclusion, safeguarding against data tampering necessitates a multifaceted approach across different database models. Secure coding practices, robust access controls, encryption protocols, and vigilant monitoring are imperative. Regular security audits and updates play a crucial role in mitigating the risks associated with data tampering attacks. As technology evolves, the need for comprehensive security measures becomes increasingly vital to ensure the integrity and trustworthiness of stored data across diverse database environments.

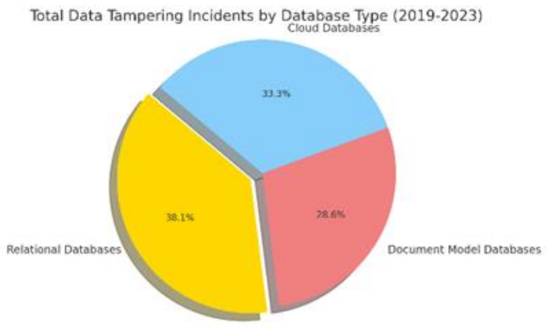

Fig1-Percentage Sharing of Data tampering among various database

2. Man in the Middle Attack The evolution of database technologies has revolutionized data management, providing efficiency and flexibility but concurrently introducing new security challenges. Among these threats, Man-in-the Middle (MitM) attacks have emerged as a significant concern. This paper delves into the exploration and comparison of MitM attack scenarios within document model databases, relational databases, and cloud databases. The goal is to equip organizations with a nuanced understanding of these scenarios, enabling the implementation of effective mitigation strategies to safeguard their invaluable data assets. 2.1.1 Document Model Database: The document model database section scrutinizes the specific risks associated with MitM attacks. Communication between clients and the database involves the exchange of JSON or BSON documents, rendering it susceptible to interception and tampering. Emphasis is placed on the potential ramifications of unauthorized changes in the database or the leakage of sensitive information. This section underscores the paramount importance of securing document transmissions to ensure the integrity and confidentiality of data.

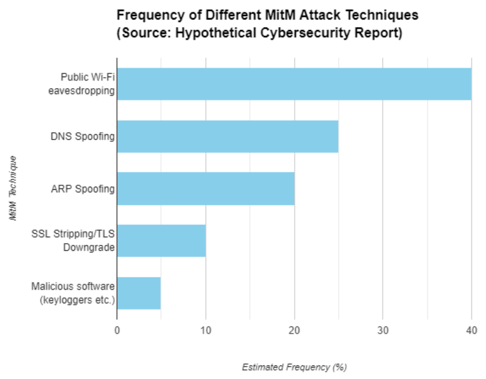

Fig 2- Frequency of Different MitM Attack.

2.1.2 Relational Database: In the realm of relational databases, the focus shifts to the vulnerability of SQL queries to MitM attacks. As SQL queries serve as a fundamental means of communication, attackers can intercept and modify them. The section explores potential outcomes, including unauthorized changes to the database structure and the exposure of confidential information. Mitigation strategies center around bolstering query security to counter these threats and maintain the integrity of relational database systems.

2.1.3 Cloud Database: The cloud database section addresses the distinctive challenges posed by distributed systems and remote server storage. MitM attacks in this environment can manifest through the interception of cloud service communication and unauthorized access to cloud credentials. The paper delves into the implications of these attacks, stressing the critical need to secure communication channels and access credentials in cloud-based database services. By doing so, organizations can fortify their defence against sophisticated MitM threats in the cloud.

Year	**Relational Databases (%)**	**Document Model Databases (%)**	**Cloud Databases (%)**
2019	25	15	40
2020	30	18	42
2021	27	20	45
2022	29	19	48
2023	35	22	50

Fig 3 – Percentage sharing of MitM cases across different database types.

2.2 Mitigation Strategies: Irrespective of database type, the paper proposes a set of general mitigation strategies to effectively counter MitM attacks. Encryption emerges as a key strategy for protecting data in transit, while robust authentication mechanisms are advocated to verify the identities of both clients and servers. The adoption of secure communication protocols with regular updates serves to address vulnerabilities systematically. Additionally, comprehensive monitoring and auditing practices are essential for detecting and responding to unusual patterns or unauthorized access, forming a holistic defence against evolving MitM threats.

3.Brute Force Attacks Brute force attacks can target various types of systems, including document models, relational databases, and cloud databases. Below, we will compare three situations where a brute force attack might occur in each of these contexts:

3.1. Brute Force Attack in Document Model: A document model database, such as MongoDB, is used to store unstructured data in JSON-like documents. An attacker attempts to gain unauthorized access to the document database by repeatedly trying different username and password combinations. The attacker may exploit weak or default credentials, attempting to guess the correct ones through brute force. Mitigation Measures: Implement strong password policies. Enforce account lockouts aver a certain number of failed login attempts. Regularly audit and update access controls and credentials.

3.2 Brute Force Attack in Relational Database: A relational database, like MySQL or PostgreSQL, is employed to store structured data in tables with predefined relationships. Data Retrieval: An attacker tries to extract sensitive information from the database by repeatedly attempting to guess SQL queries or exploiting vulnerabilities to extract data. Mitigation Measures: Apply parameterized queries to prevent SQL injection attacks. Regularly update and patch the database management system to fix any known vulnerabilities. Implement strong access controls to restrict unauthorized data accessed.

3.3 Brute Force Attack in Cloud Database: A cloud-based database, such as Amazon RDS or Azure SQL Database, is utilized to store and manage data in a scalable and flexible manner. Possible Brute Force Attack-Resource Hijacking: An attacker attempts to gain control of the cloud database by brute forcing access to the cloud provider's management console or API. The attacker may try to manipulate access controls, launch unauthorized instances, or modify configurations. Mitigation Measures: Use multi-factor authentication (MFA) for cloud provider accounts. Regularly monitor and audit cloud resources for any unusual activity. Employ identity and access management (IAM) policies to restrict access based on the principle of least privilege.

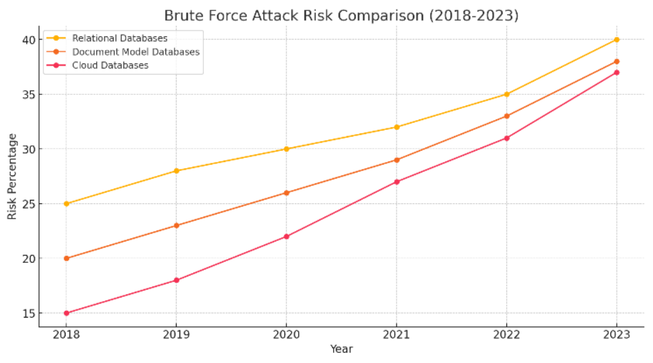

Fig 4 - Graphical data of Brute Force incident hiking from 2018 to 2023

4. Cross-Site Forgery (CSRF) attacks Cross-Site Forgery (CSRF) attacks can occur in various scenarios, regardless of whether the application is using a document model, a relational database, or a cloud database. However, the attack vectors and potential risks may differ based on the underlying technology. Let's explore three situations in each context.

4.1 Document Model Cross-Site Forgery (CSRF) attacks pose a threat across various application architectures, be it utilizing a document model like MongoDB, a relational database such as MySQL or PostgreSQL, or a cloud database like Amazon DynamoDB or Google Cloud Fire store. In the context of a document model, scenarios for CSRF vulnerabilities include inadequate anti-CSRF measures, where the absence of proper safeguards allows attackers to manipulate requests and trick authenticated users into unintended actions. Similarly, insecure JavaScript implementations may open avenues for malicious script injections, especially if client-side JavaScript heavily interacts with the document database. Cross origin Resource Sharing (CORS) misconfigurations can further exacerbate risks by enabling unauthorized domains to make requests on behalf of users, leading to unintended changes in the document database.

4.2 Relational Database In relational databases, CSRF threats may emerge due to a lack of tokenization and validation, allowing attackers to forge requests and manipulate data without proper authenticity verification. Weak session management is another vulnerability, allowing unauthorized users to hijack active sessions and perform unauthorized actions in the relational database. Cross-Site Scripting (XSS) vulnerabilities in the application introduce additional risks, as injected malicious scripts can initiate CSRF attacks, manipulating the relational database on behalf of authenticated users.

4.3 Cloud Database In the context of cloud databases, insecurely configured API endpoints without appropriate authentication and authorization checks create opportunities for attackers to exploit vulnerabilities, including CSRF attacks. Weak Identity and Access Management (IAM) policies pose a significant risk, allowing unauthorized entities to access and modify cloud database resources, potentially leading to CSRF threats. Additionally, insufficient Transport Layer Security (TLS) measures can expose communication channels, enabling attackers to intercept and modify requests between the application and the cloud database, facilitating CSRF attacks. To mitigate these risks, it is imperative to implement robust security measures such as anti-CSRF tokens, input validation, secure session management, and access controls across all application architectures. Regularly updating and patching software components, conducting security audits, and staying informed about emerging threats are essential practices to enhance overall security posture.

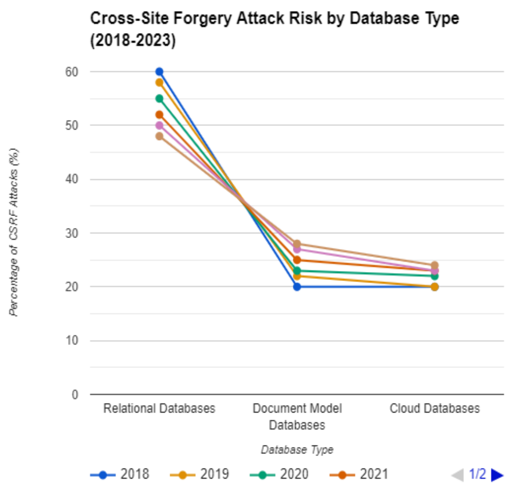

Fig 5 - CSF attack incidents from 2018 to 2023

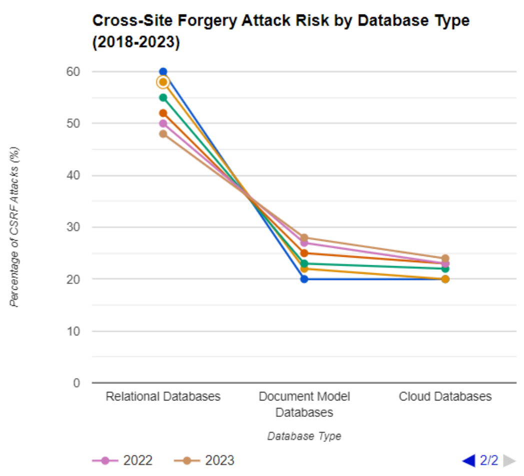

Fig 6 - CSF attack incidents of 2022 & 2023

Source-https://github.com/HamzaLatif02/HRDataset_Visualisation

VI. Prevention and Detection Strategies

Preventing and detecting various database attacks is a multifaceted endeavour critical for maintaining the confidentiality, integrity, and availability of data. Cross-Site Forgery (CSRF) prevention involves not only implementing CSRF tokens and secure coding practices but also ensuring robust validation of user input and employing mechanisms to authenticate and authorize users. Detection requires continuous monitoring of web server logs for suspicious activity, analyzing HTTP requests for anomalies, and leveraging sophisticated intrusion detection systems (IDS) capable of identifying CSRF attempts amidst the vast amount of web traffic. Session hijacking prevention strategies extend beyond implementing HTTPS encryption and strong session management techniques to include regular session monitoring and the implementation of session expiration policies. Detecting session hijacking attempts necessitates vigilant monitoring of session activity for irregularities, logging and analyzing session data for unusual patterns, and employing intrusion detection systems capable of identifying unauthorized access or session manipulation. To mitigate Distributed Denial of Service (DDoS) attacks, organizations must deploy a combination of DDoS mitigation solutions such as rate limiting, traffic filtering, and utilizing content delivery networks (CDNs) to distribute incoming traffic effectively. Detecting DDoS attacks involves monitoring network traffic for sudden spikes or patterns consistent with DDoS activity, analyzing server performance metrics to identify anomalies, and promptly responding with appropriate mitigation measures to mitigate the impact on service availability. Brute force attack prevention encompasses enforcing stringent password policies, implementing account lockout mechanisms, and promoting the use of multi-factor authentication to bolster security. Detection requires monitoring login attempts for multiple failed logins within a short one frame, analyzing authentication logs for suspicious patterns indicative of brute force attempts, and employing intrusion detection systems capable of flagging and responding to unauthorized login attempts promptly. Mitigating Man-in-the-Middle (MitM) attacks involves implementing HTTPS encryption with robust SSL/TLS configurations, verifying server certificates to prevent spoofing, and employing techniques like certificate pinning to enhance security. Detection requires continuous monitoring of network traffic for signs of unauthorized interception or tampering, analyzing SSL/TLS handshake details for anomalies, and utilizing network intrusion detection systems (NIDS)to identify and respond to MitM attack attempts promptly. Finally, preventing data tampering necessitates implementing comprehensive access controls and permissions, encrypting sensitive data at rest and in transit using strong cryptographic algorithms, and implementing integrity checks such as cryptographic hashing to detect unauthorized alterations to data. Detection involves monitoring for unauthorized changes to data, implementing file integrity monitoring systems capable of detecting and alerting administrators to unauthorized modifications, and conducting regular audits to ensure data integrity and compliance with security policies. By adopting a proactive approach to prevention and detection, organizations can effectively mitigate the risk of database attacks and safeguard their critical assets against malicious actors

VII. Conclusion

In conclusion, this research paper has provided a comprehensive examination of security threats and mitigation strategies across diverse database environments, including document-oriented, relational, and cloud databases. Through detailed analysis and comparison, we have highlighted the vulnerabilities inherent in each database model and proposed effective measures to safeguard against data tampering, Man-in-the-Middle attacks, brute force attacks, cross-site forgery attacks, distributed denial of service attacks, and session hijacking. It is evident that the evolution of database technologies has introduced new security challenges, necessitating a multi-faceted approach to data protection. Secure coding practices, robust access controls, encryption protocols, and vigilant monitoring are imperative in mitigating the risks associated with modern- day cyber threats. Furthermore, regular security audits, updates, and proactive measures are essential to maintaining the integrity and trustworthiness of stored data across diverse database environments. As technology continues to advance, the need for comprehensive security measures becomes increasingly vital to safeguard sensitive information and uphold user trust. Organizations must remain vigilant and proactive in addressing emerging threats, adopting best practices, and staying informed about evolving security trends. By implementing the recommendations outlined in this paper, organizations can enhance their overall security posture and mitigate the risks posed by malicious actors in the everchanging landscape of database security.

References

[1] R. B. Bobba, K. M. Rogers, Q. Wang, and H. Khurana, “Detecting false data injection attacks on DC state estimation,” Proceedings of the First Workshop on Secure Control Systems, 2010.

[2] T. T. Kim, and H. V. Poor, “Strategic Protection Against Data Injection Attacks on Power Grids,” IEEE Transactions on Smart Grid, vol.2, no.2, pp.326,333, June 2011.

[3] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt, Spread Spectrum Communications Handbook,revised ed. New York: McGraw Hill, 1994.

[4] Gade, Nikhita Reddy Reddy, Ugander. (2014). A Study Of Cyber Security Challenges And Its Emerging Trends On Latest Technologies.

[5] Conti, Mauro Dragoni, Nicola Lesyk, Viktor. (2016). A Survey of Man in the Middle Attacks. IEEE Communications Surveys Tutorials. 18. 1-1. 10.1109/COMST.2016.2548426.

[6] S. Ariyapperuma and C. J. Mitchell, ”Security vulnerabilities in DNS and DNSSEC,” The Second International Conference on Availability, Reliability and Security (ARES’07), Vienna, 2007, pp. 335- 342, doi: 10.1109/ARES.2007.139.

[7] R. Morris and K. Thompson, “Password Security: A Case History,” Com- mun. ACM, vol. 22, no. 11, pp. 594–597, Nov. 1979.

[8] D. C. Feldmeier and P. R. Karn, “UNIX password security - ten years later,” 9th Annual International Cryptology Conference on Advances in Cryp- tology, pp. 44-63, 2012.

[9] D. Klein, “Foiling the cracker: A survey of, and improvements to, password security,” Proceedings of the 2nd USENIX Security Workshop, pp. 5-14, 2011.

[10] Song, D., and Perrig, A. (2001). Advanced and authenticated marking schemes for IP traceback. Proceedings of IEEE INFOCOM. IEEE Press, New York.

[11] Staniford, S., Paxson, V., and Weaver, N. (2002). How to own the internet in your spare time. Proceedings of the 11th USENIX Security Symposium, 207-213. USENIX Press, Berkeley, CA.

[12] Thomas, R., Mark, B., Johnson, T., and Croall, J. (2003) NetBouncer: Client-legitimacybased High-performance DDoS Filtering. Proceedings of DARPA Information Survivability Conference and Exposition, Vol. I, pp. 14.

[13] T. Schreiber. Session Riding: A Widespread Vulnerability in Today’s Web Applications. http://www.securenet.de/papers/ SessionRiding.pdf, 2004.

[14] C. Shiflett. Security Corner: Cross-Site Request Forgeries. http://shiflett. org/articles/cross-site-requestforgeries, Dec 2004. [15] C. Shiflett. The cross-

[15] domain.xml Witch Hunt. http://shiflett.org/blog/2006/oct/ the-crossdomain.xml- witch-hunt, Oct 2006.

[16] Sivakorn, Suphannee, Iasonas Polakis, and Angelos D. Keromytis. ”The cracked cookie jar: HTTP cookie hijacking and the exposure of private infor- mation.” Security and Privacy (SP), 2016 IEEE Symposium on. IEEE, 2016.

[17] Burgers, Willem, Roel Verdult, and Marko Van Eekelen. ”Prevent session hijacking by binding the session to the cryptographic network credentials.” Nordic Conference on Secure IT Systems. Springer, Berlin, Heidelberg, 2013.

[18] Letsoalo, Enos, and Sunday Ojo. ”Survey of Media Access Control address spoofing attacks detection and prevention techniques in wirele

[19] ES GSR, R Ganeshan, IDJ Jingle, JP. Ananth- Knowledge-Based Systems 261, 110132. FACVO-DNFN: Deep learning-based feature fusion and Distributed Denial of Service attack detection in cloud computing.

[20] R Ganeshan Wireless Personal Communications 109 (396), 1-25Crow-AFL: Crow Based Adaptive Fractional Lion Optimization Approach for the Intrusion Detection.

[21] R Ganeshan, T Daniya, IOP Conference Series: Materials Science and Engineering 981 (2), 022010A systematic review on anomaly based intrusion detection system

[22] RJ Baijusha, R Ganeshan, Cyber-physical system security using decoy system

[23] S Khapre, R Ganeshan, Automated Malware Analysis in Internet of Things based Systems: A Deep Learning Approach