Transactions on Engineering and Computing Sciences - Vol. 12, No. 3
Publication Date: June 25, 2024
DOI:10.14738/tecs.123.17124.
Bassani, M. & Bagui, S. (2024). A Review of Database Attacks. Transactions on Engineering and Computing Sciences, 12(3). 101-125.



A Review of Database Attacks

Matteo Bassani
Depatment of Computer Science, University of West Florida, United States

Sikha Bagui
Depatment of Computer Science, University of West Florida, United States

ABSTRACT

In today's world, where organizations heavily rely on databases to store and access critical information, database security is a major concern. This paper provides an overview of recent trends and popular threats in database security, as well as an organized literature review of database attacks and related security methods. The primary objectives of this review are to analyze the current state of database attacks, highlight effective defense techniques, and identify research gaps and future directions. The paper categorizes the reviewed literature into various sections, including general database attacks, SQL Injection attacks, Cross-Site Scripting (XSS) attacks, Denial of Service (DoS) attacks, mixed attacks, other types of attacks, and concludes with database encryption techniques. By examining the merits and demerits of existing approaches, this review aims to contribute to a better understanding of database attacks and their related countermeasures. Finally, the findings of this paper can help researchers improve their understanding of the topic and make progress in the study of techniques to address threats to data availability, confidentiality, and integrity.

Keywords: Database, Database Security, Database Attacks, SQL Injection, Machine Learning, Survey.

INTRODUCTION

A database is an organized collection of structured data [1]. Nowadays companies and organizations strongly rely on databases as a means to store, organize, access, and manipulate data easily [2]. Typically, databases are structured in relational tables that allow for an easy modeling of different types of entities and relationships between them. These are known as relational databases and represent one of the most common and widely used types of databases [3]. Contrarily, non-relational databases do not adhere to the general relational structure, and their architecture depends strictly on the type of data that needs to be stored [4].

A user interface to a database is provided by a Database Management System (DBMS). This type of software allows an easier interaction between an authorized user or an application and the database [3]. Additionally, it helps in the fast retrieval of data by building indices on the most important tables. Moreover, it builds transaction logs to recover data in the event of a soft or hard fault and provides various methods to handle concurrency.

The significant role that data plays in the modern world is evident to everyone, as many organizations heavily rely on digitized information that can be stored and accessed easily, thanks to the low cost of memory storage and the significant development of internet infrastructure. In particular, data or database security is regarded as the process of protecting digital information stored in databases throughout its entire lifecycle to safeguard it from accidental or intentional threats such as data corruption, unauthorized access, or theft [2].

The basic requirements of data security are [5,6]:

- Data availability: Information must always be accessible to authorized users or applications.
- Data confidentiality: Information should be accessible only to authorized users and must not be disclosed by an unauthorized user.
- Data integrity: Information should be preserved without any unauthorized deletion or modification.

Database attacks are defined as any possible malicious threats that can compromise these fundamental data security requirements. While data security aims at ensuring data confidentiality, availability, and integrity in any circumstance (such as memory storage faults), a database attack specifically refers to malicious activity or unauthorized attempt to compromise these properties [7].

To classify an incident as a database attack, there must be an intention to gain unauthorized access, steal or modify data, or disrupt the normal functioning of the database. In 2022 alone, the number of data compromises in the United States reached 1802 cases. Over 422 million individuals were affected in the same year by data compromises, including data breaches, leaks, and exposures [8].

The purpose of this paper is to:

1. Present and discuss the most recent and popular threats that can affect database security and present an overview of possible solutions to these attacks.
2. Provide an organized literature review on recent trends regarding database attacks and related security methods.
3. Discuss these papers and highlight their merits/demerits in order to find the most suitable defense techniques for each type of attack.
4. Explore the gaps and suggest future research directions for this topic.

Ultimately, this review aims to analyze the current situation of database attacks and security methods in order to understand the direction of research and highlight areas that require further work. This paper is organized as follows: Section 2 presents database attack methods, discussing briefly their most popular security measures; Section 3 focusses on the main literature review; Section 3.1 covers general review papers about database attacks; Section 3.2 focuses on SQL Injection attacks; Section 3.3 focuses on Cross-Site Scripting (XSS) attacks; Section 3.4 focuses on Denial of Service (DoS) attacks; Section 3.5 focuses on mixed attacks; Section 3.6 focuses on other types of attacks; Section 3.7 focuses on database encryption

techniques; Section 4 provides a discussion about the papers, and Section 5 presents the conclusion.

ATTACK METHODS

The attack methods discussed in this work are: SQL Injection, Cross Site Scripting, Denial of Service, Excessive Privileges, DB Vulnerabilities/Bad Configuration, Weak Audit Trail and Backup Exposure.

SQL Injection

The great majority of web applications are structured in a three-tier architecture [9]:

- The presentation layer: responsible for presenting and displaying information to the user.
- The application layer: which implements the logical functionalities of the software.
- The data layer: the backend responsible for managing database access.

SQL (Structured Query Language) is the de facto standard programming language for database access and manipulation [10]. SQL Injection Attacks (SQLIA) are a classical example of attacks on a database that originates from the front-end and directly targets the back-end by exploiting knowledge of the popular SQL language and vulnerabilities in the application layer (which theoretically should properly validate any input inserted by the user). SQLI attacks consist of injecting maliciously crafted SQL commands (or parts of them) into forms used by the presentation layer to access the data [11]. The attacker needs to identify the input fields associated with the generation of the SQL queries and understand how the database tables are organized. These types of attacks can also be executed by directly injecting malicious code into the argument values of the website URL [9].

Two main forms of SQLIA are identified [12]:

- Direct attacks, which aim to insert the code into the user input variables of the SQL command in order to execute them and produce undesirable effects.
- Indirect attacks, which aim to insert permanently malicious code directly into the database table, allowing them to be executed at subsequent moments.

SQLIA is widely considered the most dangerous type of threat to a database since it can compromise the main database security services: confidentiality, integrity, authentication, and authorization [11].

In fact, a successful SQL Injection attack can [13]:

- Expose sensitive user data by displaying certain database rows or tables.
- Permanently change or delete data.
- Allow login into a protected system without any credentials.
- Alter critical data without the appropriate permissions.

Different types of SQLI attacks can be identified based on the general structure of the query. Some of the most popular ones are [11,13]:

- Tautologies: In this type of attack, the code is injected into a SQL statement with the aim of creating a condition that will always be evaluated as true. It is a basic type of SQLIA used to extract data bypassing authentication or security measures. An example could be:

SELECT * FROM Users WHERE Username='\textbf {' OR 1=1 - -}' AND Password='';

- Blind SQL Injection: The injected code asks the database true or false questions and understands the answer based on the application's response or error message.
- Union Query: This attack exploits the UNION operator to join a normal query with a malicious query.
- Piggy-based Query: This attack is one of the most dangerous as it allows the user to inject additional queries potentially different from a select statement. To execute it, the attacker only needs to identify the last user field of a query and terminate it with a \textbf {';} followed by another SQL statement (for example, a drop table). An example could be:

SELECT * FROM Users WHERE Username='user' AND Password=''\textbf{; DROP TABLE Users; - -}';

- Stored Procedures: The attacker exploits stored procedures available in the DBMS and calls them using an injected call in the SQL statement.
- Alternate Encoding: This technique aims to avoid most of IDS detections by using different encodings for strings (such as hexadecimal) to inject malicious code.
- Illegal/Incorrect Queries: This attack attempts to study the database structure by composing a deliberately wrong query to obtain useful error messages from the DBMS.

In general, the main cause of these types of attacks is the lack of validation of user input [9]. Many security countermeasures attempt to address this type of attack by analyzing the input queries and determining if they may represent a threat to a certain web application.

Popular ways to detect or prevent these types of attacks are distinguished in two main categories:

- Static methods (White Box): These methods analyze the source code of the application to detect vulnerabilities or security faults and correct them. They are usually very specific and can be costly since they require a close study of the source code.
- Dynamic methods (Black Box): These approaches try to detect the attack while the application is running by recognizing malicious queries. Many machine learning approaches fall under this category, but they can be inconsistent in some situations if compared to classic deterministic methods.

Cross Site Scripting

Cross Site Scripting attacks are a type of client-side code injection in which a malicious script is injected into a vulnerable application and then executed in a user's browser [14].

Even though Cross Site Scripting (XSS) is typically considered an application layer attack, its origins are very similar to SQL Injections, as they both result from weak input/output validations and lack of proper sanitation routines [15]. However, while XSS is more related to the front-end, it potentially represents an attack vector for a database system by compromising especially the confidentiality or the integrity of it. Additionally, in specific situations, XSS can lead to a privilege escalation attack and consequently to the exposure of specific sections of a database.

These attacks can be differentiated into three categories based on how the malicious code is delivered to the user's browser and executed [15,16,17]:

1. Reflected XSS attack: Also known as non-persistent XSS, this is the most common type of attack. It involves the execution of a malicious script, which is automatically parsed by a vulnerable server from malicious text, usually contained in a query string of a link.
2. Stored XSS attack: Also known as persistent XSS, this is one of the most dangerous types of attack. The malicious code is stored permanently on a vulnerable server and is executed by every user's browser that visits the affected page.
3. DOM-based XSS attack: This type of attack is considered a sub-class of reflected attacks, allowing the attacker to modify the Document Object Model (DOM) of the web page through the execution of a malicious script. The DOM is used to represent web documents in a browser in a structured format.

Popular ways to detect these types of attacks are categorized into different methods [15]:

- Client-side detection methods: These techniques aim to address client vulnerabilities by using browser plugins or a proxy server as defense mechanisms. They are usually classified into three categories: static approaches, which don't involve the execution of the web app; dynamic approaches, which consider the runtime behavior of the application; and hybrid approaches, which combine both strategies.
- Server-side detection methods: These methods focus on solving the problem at its origin, meaning they address server vulnerabilities. Along with static and dynamic approaches applied to the server-side, these methods may also use anomaly-based approaches that detect outliers in queries input by the user.
- Client-Server detection methods: These techniques combine both server and client-side detection approaches to address XSS attacks.

By implementing effective detection and prevention mechanisms for XSS attacks, web applications can significantly improve their security and protect sensitive data stored in the database.

Denial of Service

Denial of Service (DoS) or Distributed Denial of Service (DDoS) are types of network attacks that aim to compromise the availability of a system. While their direct target is the network, these attacks can be considered a form of database attack since their ultimate goal is to deny access to network applications, making it impossible for users to access the related database [2].

A DoS attack is considered distributed if the sources of the attack are multiple and potentially belong to different IPs or parts of the world. The most common way these attacks are executed is by attempting to overload the network bandwidth, creating an immense inbound or outbound traffic over the target [18]. Other common denial of service technique includes data corruption, network congestion, and server resource overhead (memory, CPU, etc.) [5].

Denial of Service attacks can be differentiated into two categories [19]:

- Direct DDoS attack: In this type of attack, the attacker uses many systems to flood the target directly with a large number of network packets.
- Reflection-based DDoS attack: The attacker uses innocent servers whose purpose is to reflect and amplify the malicious packets directed to the target.

Popular countermeasures to these types of attacks include [1]:

1. Reducing the attack surface: This involves changing the size of the TCP connection queue, decreasing the connection establishment time, and employing backlog techniques to ensure that the connection queue is never empty.
2. Using an Intrusion Detection System (IDS): Implementing an IDS can automatically detect a DDoS attack and trigger appropriate responses.
3. Supervised, unsupervised, or semi-supervised Machine Learning approaches: These methods can detect DDoS attacks by analyzing network traffic patterns and identifying anomalies.

By implementing these countermeasures, organizations can enhance their system's resilience against DoS and DDoS attacks, thus improving the availability of their databases.

Excessive Privileges

Excessive privileges refer to a group of attacks that target the control mechanisms implemented in many systems to differentiate database access. This type of attack is particularly dangerous as it can compromise the security aspects of a database, including integrity, confidentiality, and availability [1].

Excessive privilege attacks can be differentiated into three categories [2]:

- Excessive Privilege Abuse: This occurs when users or applications have access privileges that exceed what they actually need to perform their specific functions. The problem arises when these users use their privileges with harmful intent [20].
- Abuse of Legitimate Privilege: In this case the users have correct privileges, but they abuse their position to compromise the integrity or confidentiality of sensitive data. Even though their privileges are legitimate, they use them for unauthorized purposes [21].
- Privilege Escalation: This allows users to exploit software vulnerabilities to transform their normal privileges into those of an administrator for harmful purposes [22].

Popular countermeasures for excessive privilege attacks include [1]:

- Access Control Policy: It is important not to grant unnecessary privileges to users. Adopting a principle of least privilege, where users are given only the minimum required access rights, can significantly reduce the risk of excessive privilege abuse.
- Audit Trail: Implementing a robust audit trail helps to spot legitimate privilege abuse. By monitoring and recording user activities, organizations can detect suspicious behaviors and unauthorized access attempts.

Implementing these countermeasures can mitigate the risks associated with excessive privileges and protect the integrity, confidentiality, and availability of databases.

DB Vulnerabilities/Bad Configuration

Many attacks on databases come from issues within the databases themselves, including software vulnerabilities and systems with default configuration parameters and accounts [1].
The popularity of database vulnerabilities is a consequence of the slow response time in applying hotfixes once these vulnerabilities become known. Implementing a fix can sometimes negatively affect a system, such as conflicting with existing code or compromising service availability for a certain period. Especially in large companies with a great number of databases, the time to address and fix a vulnerability can potentially be very slow [5].

Another significant issue related to database security is represented by the misconfiguration of the system or by the use of default settings and accounts. Attackers can exploit this fact in order to access or compromise the system. Commonly, default accounts with well-known usernames and passwords present an important security risk [1].

A temporary solution to mitigate attacks on vulnerable or misconfigured databases involves the adoption of Intrusion Detection Systems (IDS) to inspect network traffic and detect attacks targeting known vulnerabilities [2]. By addressing software vulnerabilities, properly configuring database settings, and ensuring strong access controls, companies can significantly improve the security of their databases and reduce the risk of attacks.

Weak Audit Trail

More than just an attack, weaknesses in the database audit system represent a significant security risk in terms of compliance, deterrence, detection, forensics, and recovery [1].
An audit policy in a network or a database guarantees automatic and proper recording of all the database transactions and operations [2].

A weak auditing rule may represent a risk on various levels [5]:

- It may not comply with government regulations.
- It becomes difficult to detect the attack and recover from it.
- It doesn't deter attackers since there is no way to obtain forensic information.

However, some problems of auditing systems, such as performance degradation, limited granularity, or the fact that they are specific to a server platform, may still prevent their widespread deployment.

Backup Exposure

Backups are always one of the most important aspects a company should consider and plan when data is involved. On the other hand, backup storage media could also represent a big problem when this data is not protected enough and exposed to attacks. Not only theft of backup disks and tapes (physical security is always important), but also cyber attacks to access these unsecured storages may pose significant problems, especially for the confidentiality and integrity of data [5,2].

One of the easiest ways to secure backup data storage is to encrypt them and audit their access to prevent theft or loss of backups [1].

LITERATURE REVIEW

This literature review section is divided into: Review papers, papers on SQL Injection, papers on Cross Site Scripting, papers on Denial of Service, mixed attack review papers, papers on other attacks, and papers on database encryption.

Review Papers

[5] provides an overview of the most popular database attacks and highlights general methods to prevent them. After an introduction focused on general database security definitions and methodologies, the authors describe numerous security threats that can affect many company databases. Many of these threats can be attributed to SQL Injection, privilege escalation and abuse, exploitation of database or protocol vulnerabilities, weaknesses in the audit system, denial of service, unauthorized data copying, or exposure of backup data. All of these threats could potentially violate the requirements of integrity, confidentiality, and accessibility of IT systems. To ensure these requirements are met, the paper presents and explains many good practices. These practices are categorized as general approaches to secure databases and more strategic and technical measures. In conclusion, the paper emphasizes the importance of adopting these techniques to secure users' and companies' data, which may be exposed to various threats.

[1] provides an insight into the most popular and frequent database threats by defining the type of attack and the purpose of the attacker. Countermeasures for each type of attack are then provided. Some of the explored threats include excessive privileges, SQL Injections, malware, weak audit trails, backup exposure, weak authentication, DB vulnerabilities, misconfigurations, and DoS attacks. The authors identify some control methods as prevention techniques for many of the threats mentioned above. In particular, access control methods, inference policies, user authentication, auditing, and encryption of the data sources are identified as the best countermeasures to reduce risks associated with database security.

[2] starts by pointing out the importance of securing information stored in databases and provides a literature review of the most discussed control methods and techniques. The focus then shifts to security risks to databases, and some short but effective descriptions of excessive privileges, DB vulnerabilities, SQLI, and other attacks are presented with explanatory examples. Security policies are then explained, but the main focus is on encryption techniques. The author highlights the importance of encryption to ensure confidentiality, discusses different

encryption levels, and addresses the challenges of implementing these methods. The paper concludes with an interesting literature review of popular papers about DB encryption. Two metrics are considered to evaluate and compare these approaches.

[23] consists of a long and detailed literature review about the general topic of data exfiltration. The authors attempt to identify the main attack vectors that can lead to the exfiltration of sensitive or protected data. The different attack vectors are then classified and critically analyzed. Moreover, many countermeasures for these attacks are presented according to their category: preventive, detective, or investigative. The review ends with an analysis of future research challenges. This research highlights that: 1) investigative countermeasures are usually not explored in the research, 2) many countermeasures are not able to respond in real-time to attack vectors, 3) privacy is often not considered, and 4) a standard framework to evaluate countermeasures doesn't exist yet.

Table 1 presents a summary of the review papers.

Table 1: Review papers.

Nr.	Authors	Title	Attacks Discussed	Strengths	Gaps
[5]	Simanta Sarmah et al. (2019)	Database Security – Threats and Prevention	SQLI, Excessive privileges, DB Vulnerabilities / Bad Configuration, Weak Audit Trail, DoS, Protocols vulnerabilities, Unmanaged Sensitive data, Backup Exposure.	Great review of common attacks; good practices and technical measures are distinguished.	Discussion and security methods are very generic; literature is not considered in identifying good practices.
[1]	Mubina Malik et al. (2016)	Database Security - Attacks and Control Methods	SQLI, Excessive privileges, DB Vulnerabilities / Bad Configuration, Weak Audit Trail, DoS, Malware, Weak authentication, Unmanaged Sensitive data, Backup Exposure.	Good description of many attacks; Interesting literature review.	Very generic countermeasures and control methods.
[2]	Iqra Basharat et al. (2012)	Database Security and Encryption: A Survey Study	SQLI, Excessive privileges, DB Vulnerabilities / Bad Configuration, Weak Audit Trail, DoS, Protocols vulnerabilities, Weak	Great focus on encryption techniques; Literature review with metrics.	Not many papers are considered; The discussion is not exhaustive.

			authentication, Backup Exposure.		
[23]	Faheem Ullah et al. (2018)	Data exfiltration: A review of external attack vectors and counter-measures	Data exfiltration attack vectors (physical and network based).	Data exfiltration attack vectors (physical and network based).	Mostly focused on network security rather than DB security.

Papers on SQL Injection

[24] proposes a Machine Learning approach to detect malicious SQL queries that can potentially lead to SQL Injection Attacks (SQLIAs). The goal is to offer an alternative solution to SQLI attacks that doesn't require programmers to rewrite the entire Web Application and allows users to achieve good performance using the consolidated Support Vector Machine (SVM) model to identify dangerous queries. To build an efficient detection system, the authors propose a Gap-Weighted String Subsequence Kernel function, which computes a similarity measure between query strings based on the model's training. Eventually, numeric Feature Vectors are computed and fed into the Soft Margin SVM, which determines a decision boundary to identify malicious queries. The strength of this approach lies in both the pre-processing phase (denominated Feature Manipulation Algorithm), which cleans the data before further elaboration, and the String subsequence kernel, which effectively computes the similarities between a new query and a group of queries previously analyzed in the training phase. As a probabilistic algorithm based on Machine Learning, the main disadvantage of this approach is the strong dependence between the training data and the final accuracy of the model.

[25] focuses on Machine Learning and Data Mining methods for SQL injection detection. The authors discuss SQLI attacks, considering their relevance in cyber-attacks nowadays, and point out the advantages (general methods and easy to implement) and weaknesses (black box techniques, cannot guarantee security) of adopting an ML approach to detect them. Then, the focus is moved on to presenting the pipeline commonly adopted for these techniques, which involves: data collection and pre-processing, feature extraction using a vectorizer method, and finally, the training of a classification algorithm. The authors compare five famous ML algorithms, which are Decision Tree, K-Nearest neighbors, AdaBoost, SVM, and Random Forest, to choose the most effective one for this specific task. After analyzing their accuracies and their confusion matrices, they build a "Performance analysis and Iterative optimizations of the SQLI Detection Model" (PALOSDM), whose goal is to re-train and re-test the algorithms only on the misclassified samples. While this technique could potentially lead to an improved final accuracy, the risk of overfitting could be significant if the dataset isn't sufficiently big.

[10]'s survey paper aims to go over six "traditional" SQL injection detection techniques. The first one, Variable Normalization, tries to detect injections by extracting the general structure of the query, defining an allowable list for the variables, and finally implementing an SQLBlock algorithm that checks if the query is allowed in a proxy driver before submitting it to the real database. Then, AMNESIA builds a Non-Deterministic Finite Automata (NDFA) to describe the

language of all the accepted queries. Finally, a runtime algorithm checks if the SQL statements belong to this specific language or not. The Prepared Statements method relies on a Prepared Statement Replacement Algorithm (PSRA) to convert each query to a safe one with a specified structure and some limitations for the inputs. This conversion depends on the application code and on the related SQLI vulnerabilities. SQL-DOM is strictly related to CLIs (Call level interfaces) and allows getting safe queries by setting a set of strongly typed classes related to the database. In this approach, the user is not allowed to use string manipulation that can potentially lead to SQLIA. SQLrand tries to overcome the problem by randomly encoding only the predefined SQL keywords and letting a proxy decode them. For this reason, injected keywords are not randomized and hence not recognized as a valid input. To conclude, the SQLIA prevention technique is applied when the web application makes use of stored procedures to build SQL queries. This method tries to parse the procedure and analyze it at runtime to identify malicious queries.

[26] proposes a detailed framework to real-time detect and prevent SQLIA to databases. The core of the framework makes use of Machine Learning techniques (such as Naïve Bayes) as well as Deep Learning architectures (Deep CNNs) to detect malicious queries. Despite the fact that the authors don't give an insight into these methods, they provide an interesting pipeline which can be useful in real-time servers to detect anomalies in SQL commands before submitting them to the database. In particular, the proposed method uses a Multi-Phase Algorithm to double-check every query that appears to be safe initially. If the supervised approach detects a malicious SQL command, an unsupervised method (i.e. K-means) tries to compute the similarities with other dataset queries to determine the type of attack. IP and MAC addresses are also considered to guarantee that the request is coming from a trusted source. Despite their probabilistic nature due to the training phase, the authors emphasize the strength of Machine Learning and Deep Learning approaches applied to SQLI detection, highlighting the good performance of these methods.

[12] aims to provide an in-depth explanation of different SQL injection attacks by analyzing them from a detailed perspective and giving significant examples. Firstly, it offers a short analysis of SQL injection attacks and principles by explaining the difference between direct and indirect injection attacks. Secondly, it identifies and explains seven types of attacks by highlighting their main characteristics and triggering features. Examples of these include "escape character not filtered correctly," "incorrect type handling," and "vulnerabilities in the database server," among others. Thirdly, the paper presents some practical examples of SQL injection using DVWA, a web vulnerability testing program. In conclusion, it suggests and explains some general good practices to avoid and prevent these attacks.

[11] aims to provide a general and comprehensive overview of the most common and recent approaches to SQL injection attacks and detection methods. In particular, the authors discuss the different possible sources, goals, and types of SQLIA. Injection methods typically rely on user inputs in web applications, but they can also depend on cookies, server variables, or database stored variables. Attackers, to achieve their desired results of extracting data need to investigate the database by determining its injectable parameters, its fingerprint, and possibly its schema. Depending on the goal and the database vulnerabilities, they can choose among

many types of SQLI attacks, such as tautologies, blind injections, union queries, piggy-backed queries, etc. To detect and prevent these attacks, the authors identified and reviewed different SQLIA countermeasures. Starting with Query-model-based methods, the focus then shifts to obfuscation-based methods, monitoring techniques, entropy-based approaches, and ontology-based ones. Finally, the last analysis considers machine learning methods for SQL injection detection and compares the performances of the latest methods proposed in the literature.

After introducing SQL injection attacks and their associated risks, the authors of [13] try to classify different types of classical attacks, such as tautologies, piggy-backed queries, union queries, stored procedures, etc. The focus is then shifted to modern types of attacks, describing fast-flux SQL Injection attacks, a modern approach that combines phishing with classical SQL injections. Other compounded attacks include the combination of SQLI with DDoS, DNS hijacking, XSS, etc. The document concludes by proposing a comprehensive review of different SQLI prevention techniques, the first ones effective against classical database attacks, and the latter ones useful to prevent modern types of SQLIA. A final comparison between these approaches is provided to select the best ones according to the type of attack.

Table 2 presents a summary of the papers on SQL injection.

Table 2: SQL Injection papers.

Nr.	Authors	Title	Attacks Discussed	Strengths	Gaps
[24]	Paul R. McWhirter et al. (2018)	SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel	Kernel function compute a similarity measure between query strings. Numeric Feature Vectors are extracted and fed into a Soft Margin SVM.	Can be implemented in deployed Web Application; Kernel can effectively compute the query similarities; Good performance.	Probabilistic algorithm, relies on training data; Cannot guarantee security.
[25]	Ahmed A. Ashlam et al. (2022)	A Novel Approach Exploiting Machine Learning to Detect SQLi Attacks	Framework PALOSDM to retrain and retest the ML algorithms only on the misclassified samples.	Lucid discussion on ML algorithm applied to SQLI, PALOSDM could lead to better performance.	Risk of overfitting is high.
[10]	Kasra Amirtahmasebi et al. (2009)	A Survey of SQL Injection Defense Mechanisms	Many non-ML SQL injection techniques are discussed (Variable	Great and detailed review, methods are consistent if	Harder to implement than a ML approach; Relies on

			Normalization, AMNESIA, Perpared Statements, SQL-DOM, SQLrand, SQLIA).	applied together.	programmer specifics; Often not 100% reliable if applied singularly.
[26]	Ahmed A. Ashlam et al. (2022)	Multi-Phase Algorithmic Framework to Prevent SQL Injection Attacks using Improved Machine learning and Deep learning to Enhance Database security in Real-time	Proposed Multi-Phase algorithms pipeline to detect SQLI using supervised, unsupervised methods and network checks.	Interesting combination of supervised and unsupervised approaches; Good performance.	Probabilistic approach; ML techniques are not detailed described.
[12]	Limei Ma et al. (2019)	Research on SQL Injection Attack and Prevention Technology Based on Web	Review of SQL Injection attacks with triggering conditions and real-world scenario presented.	Interesting technical examples of SQLI attacks.	New algorithms or frameworks are not presented, only good practices; Research is not exhaustive.
[11]	Ines Jemal et al. (2018)	SQL Injection Attack Detection and Prevention Techniques Using Machine Learning	Review of Query-model-based, obfuscation-based, entropy-based, ontology-based methods, monitoring and machine learning approaches.	Great analysis on SQLI attack types, sources, goals; Great countermeasures literature review.	Many methods are not straightforward to implement, there is no consistent dataset to compare ML algorithms performance.
[13]	Zainab S. Alwan et al. (2017)	Detection and Prevention of SQL Injection Attack: A Survey	Long review of different SQLI prevention techniques.	New and compounded attacks are presented (e.g combinations of SQLI with DDos, DNS hijacking, XSS etc).	Performance are not always consistent; many methods cannot generalize to address different types of SQLI attacks.

Papers on Cross Site Scripting

[15] provides an extensive overview of a great variety of Cross Site Scripting attacks and detection approaches proposed in the literature. Firstly, the authors distinguish between reflected, stored, and DOM-based attacks. Secondly, they provide an analysis of XSS worms, which are vulnerabilities with the possibility of propagating among systems, describing their tasks. This description is enhanced by a considerable number of historical and more recent examples of XSS worms. Thirdly, the preconditions for a successful XSS attack are studied, considering the injectability, executability, and exfiltration abilities of the worm. To conclude, the paper explains and compares many detection systems. The authors distinguish between client-side and server-side detection approaches. These approaches could either conduct static or dynamic analysis, with the difference being related to whether the focus of the detection is the source code of the application or its run-time behavior. Hybrid approaches are also possible. Overall, this paper is very detailed and provides a comprehensive overview and starting point to go deeper into these types of attacks.

Starting with a general overview of XSS attacks, the authors of [27] propose a dynamic (or black-box) detection technique to address these common attacks. The purpose of this method is to design software that can automatically generate XSS attacks on a particular website and study and report its detection performance. This method helps in enhancing the security of the system by discovering multiple XSS vulnerabilities on a specific website. The approach first utilizes a crawler to extract as many URLs as possible. Then, a series of attacks is generated by combining the crawled URLs with some attack URLs and sent to the website. Finally, a result detection algorithm determines the type of vulnerability based on the server's response.

[17] proposes an interesting perspective on XSS detection, which directly focuses on machine learning techniques. The authors particularly emphasize XSS attacks on SNS (social network services) and attempt to adopt various classical ML methods to detect XSS-infected websites. To utilize and compare these methods, the first step is to identify numerical features that allow gathering as much relevant information related to the webpage as possible. Once these features are identified, pages are collected, considering both positive (XSS infected) and negative examples.

In conclusion, machine learning methods are trained on the dataset, and the final performances are evaluated to determine the best-suited approach for this particular task. Some examples of methods tested in this paper are: Random Forest, ADTree, AdaBoost, Naive Bayes, SVM, and k-NN.

Adaptive Random Testing (ART) for XSS vulnerability [16] consists of a black-box dynamic approach to detect XSS vulnerabilities in a web application. This method shares some analogies with Fuzzing, a popular black-box testing technique used to detect vulnerabilities. Its purpose is to collect a series of potentially dangerous payloads and systematically test them to uncover as many security issues as possible. To improve the efficiency of this method, the authors propose dynamic detection tools that use different techniques to more effectively identify malicious payloads in a pool. After a random ineffective payload is selected, a distance between this payload and others is calculated using the Jaccard distance as a similarity measure. To

compute a distance measure between URLs, each payload must be tokenized. In conclusion, instead of randomly searching throughout the pool, this method prioritizes payloads whose distance is computed within a certain range to the ineffective one. By tuning the distance requirements correctly, it becomes possible to select potentially dangerous payloads that could represent a vulnerability for the web application.

Table 3 presents a summary of the papers on Cross Site Scripting.

Table 3: Cross Site Scripting papers.

Nr.	Authors	Title	Attacks Discussed	Strengths	Gaps
[15]	Upasana Sarmah et al. (2018)	A survey of detection methods for XSS attacks	Comprehensive survey of XSS attack types, preconditions, detection approaches and tools.	Very detailed literature review; Good research starting point.	Tools are very specific and works only in specific platforms.
[27]	Xin-Yu Hou et al. (2018)	A Dynamic Detection Technique for XSS Vulnerabilities	Dynamic detection technique to address XSS. Automatically generated XSS attacks to a particular website to increase security	High vulnerability coverage, fast detection and flexible interaction.	Lacks detection of injection-triggered XSS vulnerabilities, not exhaustive and hard to automate.
[17]	Shailendra Rathore et al. (2017)	XSSClassifier: An Efficient XSS Attack Detection Approach Based on Machine Learning Classifier on SNSs	Comparison of classic Machine Learning approaches to detect XSS on web pages.	Good performance, strong pipeline and pre-processing of webpages.	Needs feature set enhancement and don't consider DL approaches.
[16]	Chengcheng Lv et al. (2019)	Adaptive random testing for XSS vulnerability	Dynamic approach to detect XSS vulnerabilities in web applications.	Good similarity measure of malicious payloads.	Probabilistic approach; Payloads selection could be inconsistent.

Papers on Denial of Service

[28] introduces a Convolutional Neural Network (CNN) approach to Intrusion Detection Systems (IDS). The authors aim to create an efficient machine learning system capable of detecting various types of cyber-attacks, with a particular focus on Denial of Service (DoS) and

other attacks attempting to gain access, privileges, or information about the targeted system. The dataset used to train and test the network is the KDD Cup 1999, which consists of approximately 5 million records comprising 41 features of TCP network traffic. To effectively train the model, the data needs to be pre-processed and transformed into a matrix of integers belonging to the range [0.255]. The architecture of the CNN consists of 2 convolutional layers followed by 2 fully-connected layers and an output layer. The accuracy and execution time of the model are then compared to other famous classical Machine Learning approaches (such as SVM, KNN, and Naive Bayes) trained on the same dataset. The performance shows that the CNN outperforms every other classical ML approach.

The idea behind [19] is to build an effective model to detect Denial of Service attacks using a Semi-Supervised approach. To achieve good results, the authors divide their model into different consequent parts. First, the network traffic coming from datasets such as NSL-KDD, UNB ISCX IDS 2012, and UNSW-NB15 is processed by a time-based sliding window algorithm to estimate the entropy (i.e. dispersion) of network features. If the entropy goes over or under certain threshold values, it indicates that the network traffic could be anomalous. The next step is to cluster such traffic information into 3 possible sets to reduce the amount of data to be classified and the impact of possible outliers on the classification. To distinguish anomalous from normal traffic within the selected samples, the information gain ratio is calculated to select the 2 clusters that contain the most anomalous information. Finally, this data is normalized and fed into an Extra-Trees ensemble classifier, which is an ensemble method similar to Random Forest that randomizes both attribute and cut-point choices compared to a simple decision tree. The authors claim good results obtained by this semi-supervised approach to DDoS detection.
[18] aims to develop a DDoS detection system by combining different Machine Learning Unsupervised methods into an ensemble of models. The NSL-KDD dataset is used for training and testing the approach. After preprocessing and dimensionality reduction (by feature selection), the data is fed into four different individual classifiers. These methods are considered unsupervised as their role is to detect outliers (network anomalies) among a great variety of "normal" samples, which represent normal network traffic. The methods used are One Class SVM, Local Outlier Factor, Isolation Forest, and Elliptic Envelope. To achieve better performance, the outputs of these classifiers are used to build three different ensembles, each using Majority Vote rule, Logistic Regression, and Naive Bayes, respectively.

Table 4 presents a summary of the papers on Denial of Service.

Table 4: Denial of Service papers.

Nr.	Authors	Title	Attacks Discussed	Strengths	Gaps
[28]	Sinh-Ngoc Nguyen et al. (2018)	Design and implementation of intrusion detection system using convolutional neural network for DoS detection	Convolutional Neural Network approach to Intrusion Detection Systems	Good performance if compared to traditional ML.	Hard to figure out best hyperparameters.

[19]	Mohamed Idhammad et al. (2018)	Detect Denial of Service attacks by using a Semi-Supervised approach	Semi-supervised approach to detect DDoS analyzing network traffic.	Use of clustering for better generalization; Good results.	Did not test in real world scenario; Could overfit.
[18]	Saikat Das et al. (2017)	A holistic approach for detecting ddos attacks by using ensemble unsupervised machine learning	Unsupervised ML ensemble to detect DDoS.	Good performance of ensembles; Able to detect new attacks/outliers.	Probabilistic outcome; Was not tested for online data.

Mixed Attack Papers

[29] analyzes some of the most common database attacks, which include Distributed Denial of Service, SQL Injection, and Brute force attacks. After describing these attacks, they propose various techniques to prevent these vulnerabilities in many Database and Cloud systems. To address DDoS attacks, they utilize the K-means algorithm to cluster the received packets and distinguish between normal and malicious activities. For Brute force attacks, a simple system is proposed to limit the number of times a user can enter a wrong password. Finally, to prevent SQL Injection, the idea is to save some classic attack patterns, which are used as a baseline to compare new queries and identify the malicious ones.

The focus of [9] is on SQL Injection and Cross Site Scripting attacks. The authors propose a query sanitization technique that is effective in addressing both types of attacks. Based on the type of HTTP request, the system, which functions as a middleware between the web app and the server, can detect the query type and invoke the appropriate query sanitization method. Both the SQL and XSS sanitization methods parse the URL containing the input query string and process it using regular expressions to search for specific patterns.

Table 5 presents a summary on some of the mixed papers.

Table 5: Mixed attacks papers.

Nr.	Authors	Title	Attacks Discussed	Strengths	Gaps
[29]	A. Patil et al. (2017)	A multilevel system to mitigate DDOS, brute force and SQL injection attack for cloud security	Client-side multi-level framework to detect some popular database attacks.	Integration of different attacks; Tested in real-world scenario.	Very basic methods.
[9]	M. Kawalkar et al. (2017)	An Approach for Detecting and Preventing SQL Injection and Cross Site	Query sanitization middleware to detect SQLI and XSS attacks that	Works for different types of SQLI and XSS attacks and databases.	Need a good query validation module.

		Scripting Attacks using Query sanitization with regular expression	works with http request.		

Papers on Other Attacks

[30] proposes a novel approach to ransomware detection. The authors begin by describing the differences between client-side and server-side ransomware, highlighting the high danger of an attack on a server. Not only is the data stored on a server often more important or numerous than the data on a client, but server-side ransomware is often designed as wipers, permanently deleting data. The idea of this study is to adopt Colored Petri Networks (CPNs) to classify all runtime queries according to a specified security policy. If a malicious query is detected, the system responds with an incident resolution module that can decide on the appropriate countermeasure to take.

[7] focuses on analyzing the vulnerabilities and attacks on Distributed Database Systems. The study begins with an overview of Distributed Database Systems and the potential risks of cyber-attacks. Particular attention is paid to discussing new trends in attacks that can easily compromise modern database systems. Some examples include NotPetya, Petya, Toxic Trojan, Phishing Whaling, or Man-in-the-Middle attacks. The paper then provides a description of these and many other attacks, and for each one previous attack features are presented. In conclusion, a summary of these attacks highlights their impact on the confidentiality, integrity, and availability of data.

[4] explores the vulnerabilities of NoSQL databases, which belong to the category of non-relational databases. The authors highlight that even though these databases do not use SQL to access and manipulate data, they are still susceptible to attacks. In particular, vulnerabilities in MongoDB, one of the most popular NoSQL databases, are explored. One of the most common attacks is PHP Array injections, which exploits the automatic translation from PHP to JSON to log in to a system without any credentials. Other possible attacks identified include OR injection, JavaScript injection, and integration with HTTP REST APIs. The techniques to address these threats include using encoding tools to build queries, running dynamic and static security tests, checking and limiting REST API integration, implementing access control, and preventing privilege escalation.

Table 6 presents a summary on other attacks.

Table 6: Other attacks papers.

Nr.	Authors	Title	Attacks Discussed	Strengths	Gaps
[30]	Lucas Ifflander et al. (2019)	Hands Off my Database: Ransomware Detection in Databases	DIMAQS, a novel anti-ransomware solution that performs runtime monitoring of	Tries to address server-side ransomwares; Good	Need to hand craft the policy.

		through Dynamic Analysis of Query Sequences	incoming queries and pattern matching using Colored Petri Nets.	performance; Easy to deploy	
[7]	A. M. Gamundani et al. (2018)	A Review of New Trends in Cyber Attacks: A Zoom into Distributed Database Systems	Review of new trends in cyber-attacks for distributed database systems.	Deals with new trends in attacks and analyze the difference with old attacks; Interesting study of impact of these attacks.	Not exhaustive, does not provide any insight into security methods.
[4]	Aviv Ron et al. (2015)	No SQL, No Injection? Examining NoSQL Security	Explores the vulnerabilities of the NoSQL databases, that belongs to the category of non-relational databases.	Deals specifically with MongoDB vulnerabilities and security methods; provide good examples.	Doesn't go deep into the attack mitigation techniques.

Papers on Database Encryption

[31] provides an interesting literature review regarding the most recent techniques in Database Encryption. After introducing the topic of database protection and explaining the need for encrypting data, the authors describe the commonly used methods to achieve this encryption. In particular, they focus on the differences between encryption and hashing and provide examples of algorithms related to these different methods of data protection. The last section of the paper presents a literature survey in which the authors compare 12 papers on specific encryption techniques. These techniques are categorized as Standalone Encryption Approaches, Hybrid, and Hashing algorithms. To conclude, a table is provided to summarize each paper and highlight their merits/demerits.

[6] introduces a novel approach to database cryptography using genetic algorithms. The first sections of the document discuss some popular database threats and attacks, such as SQL injection, Insider threat, Denial of Service, weak authentication, etc. Then, some database security techniques to address these issues are presented, including Access Control, User Identification, Encryption, etc. The authors explain genetic algorithms by describing the operators like selection, crossover, and mutation. To conclude, their proposed genetic algorithm for Database encryption is presented. The purpose is to encrypt every query, which is then compared against the values of the database. Even if the login credentials are exploited, the security of the database is guaranteed by the genetic encryption algorithm, making it impossible for attackers to decipher the content.

Table 7 presents a summary on database encryption.

Table 7: Database Encryption papers.

Nr.	Authors	Title	Attacks Discussed	Strengths	Gaps
[31]	P. Singh et al. (2015)	Database security using encryption	Literature review regarding the most recent techniques on Database Encryption.	Good starting point.	Literature review doesn't go deep.
[6]	S. R. Devara et al. (2022)	Improved database security using cryptography with genetic operators	Novel approach to database cryptography using genetic algorithms along with traditional authentication.	Security of the database is enforced by the genetic encryption algorithm to make hard for the attackers to decipher.	Could be not straightfor ward to understand and implement.

DISCUSSION

The discussion section is divided into discussion on review papers, discussion on SQL Injection papers, discussion on Cross Site Scripting papers, discussion on Denial of Service papers, discussion on mixed attack papers, discussion on other attack papers, and a discussion on encryption papers.

Discussion on Review Papers

When it comes to generic review papers, the current trend is to provide a detailed analysis of various types of database attacks and best practices to prevent them. These include common attacks like DoS or SQL injection, as well as less frequent ones like excessive privileges. This can be seen in papers such as Sarmah et al. (2019) and Malik (2016) [5,1].

However, both of these papers mainly offer general countermeasures rather than specific security algorithms. The countermeasures they suggest are mostly a list of good practices rather than concrete security methods. While these papers effectively describe attacks and cite relevant literature sources, they don't explore the security measures available in current literature.

Basharat et al. (2012) and Ullah (2018) [2,23] provide more detailed explanations of security methods by referencing existing literature. However, they focus only on the most common attack trends and don't cover many recent developments.

Overall, these review papers serve as a good starting point for learning about database attacks and security methods. But they don't offer significant insights into the latest trends in this topic.

Discussion on SQL Injection

Among all the different types of database attacks discussed in research papers, SQL Injection is the most commonly studied. An interesting trend in recent research is the use of Machine Learning techniques to enhance the security measures against SQLI. Researchers have been developing middleware solutions that can be integrated into web applications without requiring significant resources. These solutions utilize ML algorithms to analyze SQL queries,

extract features, and determine whether a query is normal or malicious. Several papers, such as McWhirter et al. (2018), Ashlam (2022a, 2022b), and Jemal et al. (2020), explore supervised, unsupervised, and semi-supervised ML approaches for building SQLI detection mechanisms [24,25,26,11].

However, ML-based approaches have certain limitations. They heavily rely on the data they are trained on, and there is currently no standard dataset available for comparing the performance of different algorithms. Moreover, these methods have a probabilistic nature and can overfit easily. It is challenging to create rules that can generalize well, and even slight variations in dangerous queries can lead to false negatives, where malicious queries are not detected. For these reasons, "traditional" methods, although more complex to implement, remain more reliable, especially when used in combination. According to Amirtahmasebi (2009), the combination of multiple techniques is necessary for effective database security [10].

Additionally, many recent and complex attack scenarios, such as combining SQLI with DDoS, DNS hijacking, or XSS, are not extensively covered in most research papers [13].

Discussion on Cross Site Scripting Papers

Cross Site Scripting attacks are more complicated and harder to execute compared to SQL Injection. However, they are also more dangerous because in some specific situations, for example when they can lead to a privilege escalation attack. Machine Learning has been helpful in addressing these attacks too, but it faces the same problems seen in discussing SQL injection, as explained by Rathore et al. (2017) and Lv et al. (2019) [17,16]. The main issues are that Machine Learning has a random outcome and relies heavily on the training data, which limits its general use. Additionally, processing millions of real-time queries is difficult, so having an algorithm that can select specific harmful parts, sessions, or traffic would be desirable.

Dynamic approaches, which analyze web application behavior in real-time, are often popular. However, for many web applications, it is more important to analyze the code itself. This helps identify vulnerabilities that could make a website susceptible to XSS attacks that can compromise its database security.

Discussion on Denial of Service Papers

Denial of Service attacks are not as dangerous as SQL Injection or Cross-Site Scripting attacks, but they can still be a problem since they can compromise the availability of a system and potentially lead to more significant attacks. To handle DoS, the latest trend is to use Machine Learning (ML) techniques like supervised, unsupervised, or semi-supervised methods. However, ML methods have some common problems, like difficulty in adapting to different situations, the risk of producing uncertain outcomes, and limited real-world testing. It can also be challenging to convert TCP or UDP traffic into numeric features that ML models can understand. Nevertheless, these ML-based methods have potential and most of them give good performance. Additionally, ML is often more effective at detecting DoS and DdoS attacks as it is at detecting SQLI and XSS, which typically require more specific contextual information.

Discussion on Mixed Attack Papers

Protecting against different types of attacks simultaneously, except for similar ones like SQL Injection and Cross-Site Scripting, can be quite challenging. In the paper by Patil et al. (2017), they try to create a framework to handle simultaneously three types of attacks [29]. However, the methods they propose are often basic and do not address more complex and dangerous attacks. In contrast, Kawalkar et al. (2017) suggest a unified approach using regular expressions for validating queries [9]. The advantage of this system is that it can work well for various databases and defend against SQLI and XSS attacks. A significant drawback is that the regular expression validation part needs to be manually designed.

Discussion on Other Attack Papers

SQL Injection, Cross Site Scripting, and Distributed Denial of Service attacks are not the only ones that can afflict modern databases. There are other common threats like Ransomware, NoSQL injections, Phishing, Man-in-the-Middle attacks, Peyta, and NotPeyta. Surprisingly, these attacks are often not considered when it comes to securing databases according to the literature. The few papers that do examine these complex attacks often lack detailed ways to detect and handle them properly.

It's worth mentioning that NoSQL databases are commonly used, especially in social networks and beyond [4]. Distributed database attacks and server-side attacks are also real and significant concerns [7,30]. To effectively address these attacks, more research is needed to discover better and more efficient ways to defend against them.

Discussion on Database Encryption Papers

Encryption has always been considered one of the most secure ways to protect data. In the context of databases, encryption is commonly used to ensure the confidentiality and integrity of data. The literature provides a good starting point with various approaches such as Standalone Encryption, Hybrid methods, and Hashing algorithms [31]. A more interesting perspective is exploring the use of genetic algorithms to enhance the security of encrypted databases [6]. However, more research is needed in this direction to find better ways to protect data and implement encryption methods that can also help prevent attacks like SQL Injection.

CONCLUSION

This paper provides an overview of the current state of literature on database attacks and the corresponding countermeasures, in order to highlight the sector in which more research needs to be done. Database security has been and still is one of the main concerns in cybersecurity in our times. Exposing sensitive or valuable data to attacks that can compromise their confidentiality, integrity or availability could represent a significant risk.

The recent trends to secure databases is to build and train Machine Learning models which are able to detect most of the dangerous queries or malicious network traffics. While these techniques have some advantages (for example they are easy to deploy, they require low hand tuning and they are usually lightweight for a real time detection), they still need further improvements. Many methods heavily rely on the train and validation database, and their probabilistic nature makes them not suitable for applications in which the security of data is

the main concern (for example medical, governmental or political databases). For these reasons, many Intrusion Detection Systems (IDS) or security frameworks should consider traditional hand-crafted methods to detect dangerous queries or malicious http requests.

Additionally, more research needs to be done to develop static strategies which are able to detect DB and web applications vulnerabilities just observing at the code. Especially for SQL Injection and Cross Site Scripting, more effort needs to be put in find good strategies to validate input of the users. To conclude, many modern systems are completely unable to handle big and complex attacks, which could also represent a significant problem especially for major companies.

ACKNOWLEDGEMENT

This work has been partially supported by the Askew Fellowship at The University of West Florida.

References

[1]. M. Malik and T. Patel, *Database security-attacks and control methods*. International Journal of Information, 2016. 6(1/2): p. 175–183.

[2]. I. Basharat, F. Azam, and A. W. Muzaffar, *Database security and encryption: A survey study.* International Journal of Computer Applications, 2012. 47(12).

[3]. S. Kulkarni and S. Urolagin, *Review of attacks on databases and database security techniques.* International Journal of Emerging Technology and Advanced Engineering, 2012. 2(11): p. 253–263.

[4]. A. Ron, A. Shulman-Peleg, and E. Bronshtein, *No sql, no injection? examining nosql security.* arXiv (Cornell University), 2015.

[5]. S. S. Sarmah, *Database security–threats & prevention.* International Journal of Computer Trends and Technology, 2019. 67(5): p. 46–53.

[6]. S. R. Devara and C. Azad, *Improved database security using cryptography with genetic operators*. Research Square, 2022.

[7]. A. M. Gamundani and L. M. Nekare, *A review of new trends in cyber attacks: A zoom into distributed database systems.* 2018 IST-Africa Week Conference (IST-Africa) Proceedings, 2018.

[8]. Petrosyan, A., *Number of data breaches and victims in U.S*. *2005-2022* Statistica, 2023. https://www.statista.com/statistics/273550/data-breaches-recorded-in-the-united-states-by-number-of-breaches-and-records-exposed/.

[9]. M. Kawalkar and P. Butey, *An approach for detecting and preventing sql injection and cross site scripting attacks using query sanitization with regular expression.* International Journal of Computer Trends and Technology, 2017. 49(4): p. 237–245.

[10]. K. Amirtahmasebi, S. R. *Jalalinia, and S. Khadem, A survey of sql injection defense mechanisms.* International Conference for Internet Technology and Secured Transactions, 2009. p. 1–8.

[11]. I. Jemal, O. Cheikhrouhou, H. Hamam, and A. Mahfoudhi, *Sql injection attack detection and prevention techniques using machine learning.* International Journal of Applied Engineering Research, 2020. 15(6): p. 569–580.

[12]. L. Ma, D. Zhao, Y. Gao, and C. Zhao, *Research on sql injection attack and prevention technology based on web.* 2019 International Conference on Computer Network, Electronic and Automation (ICCNEA), 2019. p. 176–179.

[13]. Z. S. Alwan and M. F. Younis, *Detection and prevention of sql injection attack: a survey.* International Journal of Computer Science and Mobile Computing, 2017. 6(8): p. 5–17.

[14]. V. Nithya, S. L. Pandian, and C. Malarvizhi, *A survey on detection and prevention of cross-site scripting attack.* International Journal of Security and Its Applications, 2015. 9(3): p. 139–152.

[15]. U. Sarmah, D. Bhattacharyya, and J. K. Kalita, *A survey of detection methods for xss attacks.* Journal of Network and Computer Applications, 2018. 118: p. 113–143.

[16]. C. Lv, L. Zhang, F. Zeng, and J. Zhang, *Adaptive random testing for xss vulnerability*. 2019 26th Asia-Pacific Software Engineering Conference (APSEC), 2019. p. 63–69.

[17]. S. Rathore, P. K. Sharma, and J. H. Park, *Xssclassifier: An efficient xss attack detection approach based on machine learning classifier on snss.* Journal of Information Processing Systems, 2017. 13(4).

[18]. S. Das, D. Venugopal, and S. Shiva, *A holistic approach for detecting ddos attacks by using ensemble unsupervised machine learning.* Advances in Information and Communication: Proceedings of the 2020 Future of Information and Communication Conference (FICC), Volume 2, 2020. p. 721–738.

[19]. M. Idhammad, K. Afdel, and M. Belouch, *Semi-supervised machine learning approach for ddos detection.* Applied Intelligence, 2018. 48: p. 3193–3208.

[20]. E. Bertino, S. Jajodia, and P. Samarati, *Database security: Research and practice.* Information systems, 1995. 20(7): p. 537–556.

[21]. Houben, A.J.H.M., et al., *Quantitative analysis of retinal vascular changes in essential and renovascular hypertension.* Journal of hypertension, 1995. 13(12).

[22]. G. Pernul, *Database security.* Advances in Computers, 1994. p. 1–72.

[23]. T. F. Lunt and E. B. Fernandez, *Database security*. ACM Sigmod Record, 1990. 19(4): p. 90–97.

[24]. F. Ullah, M. Edwards, R. Ramdhany, R. Chitchyan, M. A. Babar, and A. Rashid, *Data exfiltration: A review of external attack vectors and countermeasures.* Journal of Network and Computer Applications, 2018. 101: p. 18–54.

[25]. P. R. McWhirter, K. Kifayat, Q. Shi, and B. Askwith, *Sql injection attack classification through the feature extraction of sql query strings using a gap-weighted string subsequence kernel.* Journal of information security and applications, 2018. 40: p. 199–216.

[26]. A. A. Ashlam, A. Badii, and F. Stahl, *A novel approach exploiting machine learning to detect sqli attacks.* in 2022 5th International Conference on Advanced Systems and Emergent Technologies (IC ASET), 2022. p. 513–517.

[27]. A. A. Ashlam, A. Badii, and F. Stahl, *Multi-phase algorithmic framework to prevent sql injection attacks using improved machine learning and deep learning to enhance database security in real-time.* 2022 15th International Conference on Security of Information and Networks (SIN), 2022. p. 1–4.

[28]. X.-Y. Hou, X.-L. Zhao, M.-J. Wu, R. Ma, and Y.-P. Chen, *A dynamic detection technique for xss vulnerabilities.* 2018 4th Annual International Conference on Network and Information Systems for Computers (ICNISC), 2018. p. 34–43.

[29]. S.-N. Nguyen, V.-Q. Nguyen, J. Choi, and K. Kim, *Design and implementation of intrusion detection system using convolutional neural network for dos detection.* ICMLSC '18: Proceedings of the 2nd International Conference on Machine Learning and Soft Computing, 2018. pp. 34–38.

[30]. A. Patil, A. Laturkar, S. Athawale, R. Takale, and P. Tathawade, *A multilevel system to mitigate ddos, brute force and sql injection attack for cloud security.* 2017 International Conference on Information, Communication, Instrumentation and Control (ICICIC), 2017. p. 1–7.

[31]. L. Iffl¨ander, A. Dmitrienko, C. Hagen, M. Jobst, and S. Kounev, *Hands off my database: Ransomware detection in databases through dynamic analysis of query sequences.* arXiv (Cornell University), 2019.

[32]. P. Singh and K. Kaur, *Database security using encryption.* 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), 2015. p. 353–358.