

Database Hacking: Common Cyber Attacks

**** Name ****

**** Prepared for, Date ****

1 Introduction

Databases hold and handle a tremendous amount of sensitive data interconnected to everyday processes in our World. However, they are prime targets for criminals seeking to gain unauthorized access to valuable data. Database hacking refers to the exploitation of vulnerabilities within database systems to gain unauthorized entry, extract sensitive information, manipulate data, or disrupt operations. In this article, we will delve into the world of database hacking, exploring common methods used in cyber attacks targeting databases.

1.1 Database Cyber Attacks

1. SQL Injection (SQLi): SQL injection is a widespread attack vector where an attacker manipulates input fields of a web application to inject malicious SQL statements. By exploiting poor input validation, the attacker can execute arbitrary SQL queries. This may lead to unauthorized data extraction, data modification, or even complete control over the database.
2. Cross-Site Scripting (XSS): Cross-Site Scripting attacks involve injecting malicious scripts into web pages viewed by users. When a vulnerable web application fails to properly validate or sanitize user-supplied data, the attacker can inject malicious scripts that execute within the victim's browser. XSS attacks can be used to steal session cookies, gain unauthorized access to the database, or perform other malicious actions.
3. Privilege Escalation: Privilege escalation attacks aim to elevate an attacker's access privileges within a database. By exploiting vulnerabilities or misconfigurations, attackers seek to escalate from a lower privileged user to a higher privileged user. This

allows them to gain access to more sensitive data, modify database structures, or perform other unauthorized actions.

2 Oracle Write Once Read Many(WORM)

Oracle Object Storage provides a reliable and scalable platform for storing and managing large amounts of unstructured data. The "write once read many" (WORM) feature in Oracle Object Storage allows you to write data once and prevent any subsequent modifications or deletions, ensuring data immutability for compliance and regulatory purposes. Here's a step-by-step guide on how to use Oracle Object Storage with the WORM feature:

1. Set up an Oracle Cloud Account: If you don't have an Oracle Cloud account, sign up for one at <https://www.oracle.com/cloud/>. Ensure that you have the necessary permissions and access to create and manage Object Storage resources.
2. Create an Object Storage Bucket: Log in to the Oracle Cloud Console and navigate to the Object Storage service. Create a new bucket, which acts as a container for your objects. Provide a unique name for your bucket and choose the appropriate compartment and storage tier based on your requirements.
3. Enable WORM: Once the bucket is created, select it from the bucket list. In the bucket details page, navigate to the "Management" tab and click on "Object Versioning." Enable object versioning to ensure that objects cannot be overwritten or deleted.
4. Upload Objects: To upload objects to the bucket, click on the "Upload Object" button or use the API/SDKs provided by Oracle. Select the files you want to upload from your local machine and specify the destination within the bucket. Once the upload

is complete, the objects are stored securely in the bucket and cannot be modified or deleted due to the WORM feature.

5. **Access and Retrieve Objects:** To read the objects stored in the bucket, you can use various methods such as the Oracle Cloud Console, REST APIs, or SDKs provided by Oracle. Authenticate yourself using the appropriate credentials and retrieve the objects by specifying their bucket name and object name. You can download or access the objects based on your requirements.
6. **Retention Period:** Oracle Object Storage also allows you to set a retention period for objects, specifying the duration for which the objects are immutable. This ensures that the objects cannot be modified or deleted during the specified period. To set the retention period, go to the "Management" tab in the bucket details page, click on "Object Lifecycle Policy," and define the retention period for the objects.
7. **Compliance and Governance:** The WORM feature in Oracle Object Storage helps meet compliance and governance requirements, ensuring data integrity and immutability. By preventing modifications or deletions, it ensures that stored objects remain unchanged for the defined retention period.

3 SQL Injection

3.1 Steps to execute a SQL Injection Kill Chain

1. **Reconnaissance:** The attacker gets information about the target web application at this early phase. They investigate the structure, behavior, and potential vulnerabilities of the application. This reconnaissance may include analyzing the source code of the

application, engaging with its user interface, or performing network scans.

2. **Identify Injection Points:** Within the target online application, the attacker detects input fields or parameters where user-supplied data is not adequately sanitized or validated. Login forms, search areas, and comment sections are all common injection locations. These flaws provide opportunities for an attacker to inject malicious SQL code.
3. **Create Malicious Payload:** After identifying the injection locations, the attacker creates malicious SQL payloads to exploit the vulnerabilities. They write SQL statements that distort the original query, allowing them to evade authentication, extract sensitive data, or change the database's content.
4. **Inject Malicious Payload:** The attacker now enters the created payload into the target input fields. Without sufficient input validation, the web application accepts the inserted SQL code as legitimate and includes it into the database query.
5. **Execution of Malicious Payload:** The database server executes the altered SQL code, resulting in unforeseen consequences. The attacker's payload may take sensitive data from the database, such as usernames, passwords, or credit card information, or it may change existing data.
6. **Exploitation:** Following the successful execution of the malicious payload, the attacker uses the gained information or control over the database for illicit objectives. They may acquire unauthorized access to other sections of the system.
7. **Covering Tracks:** The attacker attempts to cover their tracks by erasing evidence of the attack in order to avoid detection. This may include removing log entries, changing

records, or concealing their activity in order to make it difficult for forensic analysts or incident response teams to track the assault back to its source.

3.2 SQL Injection Techniques

1. Blind SQL Injection: When an attacker cannot see the actual database output but may infer information from the application's behavior, this is known as blind SQL injection. The attacker creates conditional SQL queries that evaluate to true or false, allowing them to infer information from the application's response or behavior. This category includes techniques such as Boolean-based blind, time-based blind, and error-based blind SQL injection.
2. Union-Based SQL Injection: The UNION SQL operator, which combines the results of two or more SELECT queries, is used in union-based SQL injection. Attackers use the UNION operator to introduce additional queries that retrieve data from other database tables. The injected query must have the same amount of columns and be compatible with the original query's data types.
3. Function Call Injection: Some web apps allow you to run stored procedures or functions from your database. Attackers take use of this functionality by inserting malicious SQL code into input fields used to invoke database operations or functions. This can lead to arbitrary code execution or unauthorized access to sensitive data.

4 SQL Injection in Databases

4.1 SQL Injection in Oracle

BSQL Hacker available on GitHub, is open source and suitable for both experienced users and beginners seeking to automate SQL injections, particularly blind SQL injections. It allows attackers to escalate privileges within a compromised system. Attackers can execute arbitrary SQL commands, modify database entries, and gain unauthorized access to sensitive information by exploiting SQL injection vulnerabilities.

4.2 SQL Injection in MySQL

A error-based SQL injection attack takes advantage of vulnerabilities in a web application's input validation to alter SQL queries and retrieve sensitive data from the database. In an error-based attack, the attacker injects malicious SQL code that deliberately triggers database errors, revealing valuable information.

References

- [1] Lu, Dongzhe, Jinlong Fei, and Long Liu. "*A Semantic Learning-Based SQL Injection Attack Detection Technology.*" *Electronics* 12.6 (2023): 1344.
- [2] Stiawan, Deris, et al. "*An Improved LSTM-PCA Ensemble Classifier for SQL Injection and XSS Attack Detection.*" *Computer Systems Science Engineering* 46.2 (2023).
- [3] Alazmi, Suliman, and Daniel Conte de Leon. "*Customizing OWASP ZAP: A Proven Method for Detecting SQL Injection Vulnerabilities.*" 2023 IEEE 9th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS). IEEE, 2023.
- [4] Kambre, Omkar Kamleshwar, Kabir Kiran Shah, and Pankaj Dulabhai Rathod. "*SQL Injection Attacks and Defense Mechanisms.*" *International Research Journal of Innovations in Engineering and Technology* 7.2 (2023): 101.
- [5] Abdullayev, Vugar, and Alok Singh Chauhan. "*SQL Injection Attack: Quick View.*" *Mesopotamian Journal of CyberSecurity* 2023 (2023): 30-34.
- [6] Halfond, William G., Jeremy Viegas, and Alessandro Orso, "A classification of SQL-injection attacks and countermeasures" *Proceedings of the IEEE international symposium on secure software engineering.*, Vol. 1. IEEE, 2006.
- [7] Clarke, Justin *SQL injection attacks and defense.*, Elsevier, 2009.
- [8] Anley, Chris, *Advanced SQL injection in SQL server applications.*, 2002