Blind SQL Injection

Contributor(s): kingthorin

Description

Blind SQL (Structured Query Language) injection is a type of SQL Injection attack that asks the database true or false questions and determines the answer based on the applications response. This attack is often used when the web application is configured to show generic error messages, but has not mitigated the code that is vulnerable to SQL injection.

When an attacker exploits SQL injection, sometimes the web application displays error messages

response. This attack is often used when the web application is configured to show generic error messages, but has not mitigated the code that is vulnerable to SQL injection.

When an attacker exploits SQL injection, sometimes the web application displays error messages from the database complaining that the SQL Query's syntax is incorrect. Blind SQL injection is nearly identical to normal SQL Injection, the only difference being the way the data is retrieved from the database. When the database does not output data to the web page, an attacker is forced to steal data by asking the database a series of true or false questions. This makes exploiting the SQL Injection vulnerability more difficult, but not impossible. .

What is time-based SQL injection attack?

What are conditional time delays in SQL injection?

Simply put, **by injecting a conditional time delay in the query the attacker can ask a yes/no question to the database**. Depending if the condition is verified or not, the time delay will be executed and the server response will be abnormally long. This will allow the attacker to know if the condition was true or false.

Boolean Exploitation Technique is basically

an SQL Injection Exploitation technique where a set of Boolean operations are executed in order to extract juicy information regarding the tables of the database of an web application.

Time-Based Blind SQL Injection using Heavy Query

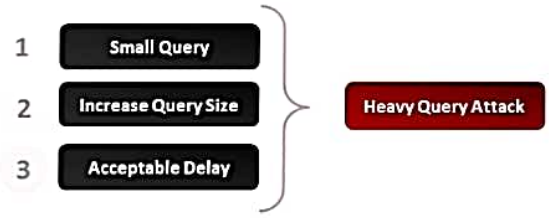
Using heavy queries instead of time delays

For different reasons, it might happen that it is impossible to use time delay functions or procedures in order to achieve a classic time delay injection. In these situations, the best option is to simulate it with a **heavy query that will take noticeable time to get executed by the database engine**. This article shows how it can be done and it presents an example for the main DBMS.

Principle

The injected query should not rely on user tables since in most cases the attacker will have no information about those yet. Queries presented in the following section rely on system tables. The execution time is essentially caused by the large number of lines returned.

Keep in mind that the time to execute each query presented in this article **can tremendously vary depending on the number of rows** contained or returned by the table (or view). This number can be influenced by many factors like: the permissions of your user, the size of the database, the server performance, etc. You should begin with a query joining 2 tables and slowly increment until you can generate an acceptable delay.



MySQL or SQL Server

Chances are low that you have to use the heavy query approach in MySQL or SQL Server since these DBMS make it relatively easy to inject classic code delays in a vulnerable field. However, it could still happen if, for example, **some function or characters are blacklisted**. Here is an example of heavy query that would work fine on SQL Server and on MySQL (version 5 or more).

If the server response takes more time, a vulnerability is probably present. Otherwise we can conclude the field is safe.

Oracle

As explained in the article about time-based attacks, in most cases you will need to use heavy queries in order to achieve this kind of SQL injection. Below is an example of query that takes a lot of time to be executed in this DBMS.

HEAVY ORACLE QUERY.

```
SELECT count(*) FROM all_users A, all_users B, all_users C
```

In my Oracle test environment the query above is executed in no time since **I have very few users**. When I grow the FROM clause to 7 tables it takes about 15 seconds. Let's now see how it could be integrated in a vulnerable field.

MALICIOUS PARAMETER (TIME-BASED INJECTION).

```
1 AND 1<SELECT count(*) FROM all_users A, all_users B, all_users C
```

QUERY GENERATED.

```
SELECT * FROM products WHERE id=1 AND 1<SELECT count(*) FROM all_users A, all_users B,
all_users C
```

Here again, if the test slows significantly the server response, you can conclude a vulnerability is present in the field.

Additionnal Information

As mentioned in the article about time-based attacks, the heavy query approach will have **noticeable impacts on CPU and server resources usage**. Whenever possible, try to inject a time delay that will not be CPU intensive and stick to standards techniques.

You must also be aware that **the injected query will most likely be executed only once**. The database optimizer will execute it, store its result and use the returned value(s) when testing the WHERE clause against each record. As you can guess, this is must faster than executing the query each time. It should be mentioned however that the query will not be executed if the optimizer detects that the WHERE clause is always false. To avoid any unexpected results **you should always try to generate a WHERE clause that will be verified for at least one record**.

If you want more information about this technique, you can find an interesting paper written by Chema Alonzo here.

Posted in : Blind SQL Injection

Tagged: Heavy Query | MySQL | Oracle | SQL Server | System Table

Related Articles

Check out the related articles below to find more of the same content.

Time-Based Blind SQL Injection Attacks
Perform tests by injecting time delays

Find Column Names for SQL Injection
Extracting column names for a given table

Find Table Names for SQL Injection
Extracting table names to achieve SQL injection

What is in-band SQL injection?

In-band SQL injection is a type of SQL injection where the attacker receives the result as a direct response using the same communication channel. For example, if the attacker performs the attack manually using a web browser, the result of the attack will be displayed in the same web browser. In-band SQL injection is also called *classic SQL injection*.

Example of in-band SQL injection

The simplest type of in-band SQL injection is when the attacker is able to modify the original query and receive the direct results of the modified query. As an example, let's assume that the following query is meant to return the personal data of the current user and display it on-screen.

Error-based SQLi is **an in-band SQL Injection technique that relies on error messages thrown by the database server to obtain information about the structure of the database**. In some cases, error-based SQL injection alone is enough for an attacker to enumerate an entire database.

Union-based SQLi is an in-band SQL injection technique that leverages the UNION SQL operator to combine the results of two or more SELECT statements into a single result which is then returned as part of the HTTP response.

What is end of line comment SQL injection?

End of Line Comment: **After injecting code into a particular field, legitimate code that follows if nullified through usage of end of line comments**: SELECT * FROM user WHERE name = 'x' AND userid IS NULL; --'; Comments in a line of code are often denoted by (--), are ignored by the query.

Piggy-Backed Queries: – **Attacker tries to inject malicious queries in the original query**. Malicious query which is injected in the original query is called piggy backed query. Inference: – In this attack an attacker modify the behavior of a database application.

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the da[redacted] server.

How illegal is SQL injection?

Yes, **using an SQL injection on someone else's website is considered illegal**. SQL injections are a type of computer attack in which malicious code is inserted into a database in order to gain access to sensitive information. 7 Feb 2017