MRunal Follow

Sep 2, 2019 · 4 min read · ✨ · Listen

When i found boolean based sql Injection

i will give an explanation on a technique that belongs to the branch of SQLi Injection, in this case, it is called "Blind SQLi Boolean Based".

1.- What is a "SQLi Boolean Based Blind"?

If they can be fixed in the name, this one says them all. When we say "Boolean based" we mean that it is based on Boolean values, that is, true or false / true and false. And when I say "Blind" I mean that the Injection is blind, that is, it does not show us any sign of error.

The only way to extract data that is in the current database, would be using brute force, ie guessing.

2.- Explaining two important functions

One of the most common functions we use when exploiting a Blind Boolean Based is ascii (), with this we return some valid character from the "ASCII" table (if I am not mistaken).

Another would be substring () with this we return a substring of another substring.

3.- Checking if the site is vulnerable

And if I put it, and 1 = 0, it should give me a false / false result, since 1 is not equal to 0. An example on a website:

http://victim.com.ve/news.php?section= regional & id = 19350 'and 1 = 1 — -

As you can see, at the end put a "- -", which is a comment, the server only reads what is between the ' and the commentary — — that is, and 1 = 1 : D. Everything that goes after — — the server will not read it.

LETS TALK HOW I FOUND .....

I found a Boolean-based SQL injection, which occurs when data input by a user is interpreted as a SQL command rather than as normal data by the backend database.

This is an extremely common vulnerability and its successful exploitation can have critical implications.

That's **confirmed** the vulnerability by executing a test SQL query on the backend database. In these tests, SQL injection was not obvious, but the different responses from the page based on the injection test allowed to identify and confirm the SQL injection.

URL : http://xyz.com/****.php?id=1 OR 17–7=10

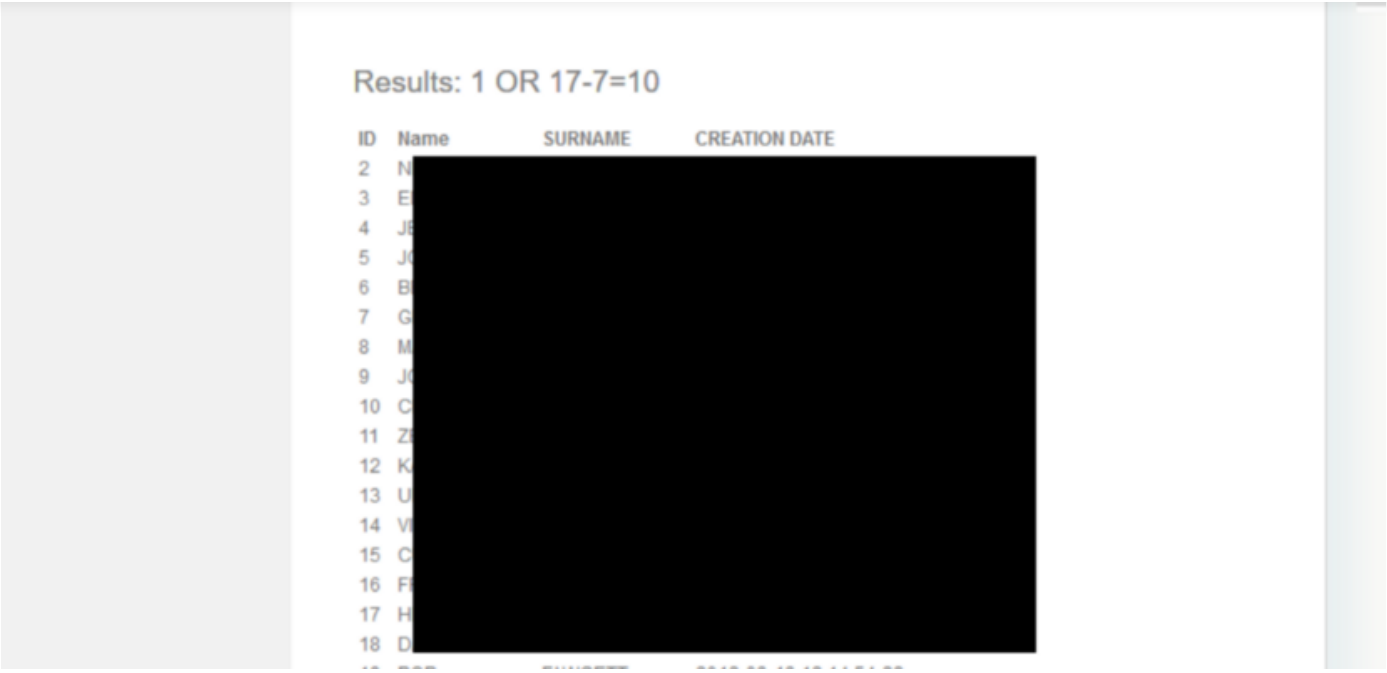
Parameter Type : GET

Payload: 1+OR+17–7%3d10

GET /****.php?id=1%20OR%2017–7%3d10 HTTP/1.1
Host: xyz.com
Cache-Control: no-cache
Connection: Keep-Alive
Accept:

Accept-Language: en-us,en;q=0.5
Cookie: PHPSESSID=ca551b1c1602c7da*********
Accept-Encoding: gzip, deflate

Photo by Caleb Jones on Unsplash



and this is Database User Has Admin Privileges bug as well .

Impact

Depending on the backend database, the database connection settings and the operating system, an attacker can mount one or more of the following type of attacks successfully:

- Reading, updating and deleting arbitrary data/tables from the database
- Executing commands on the underlying operating system

>> Gain full access to the database server.

- Gain a reverse shell to the database server and execute commands on the underlying operating system.
- Access the database with full permissions, where it may be possible to read, update or delete arbitrary data from the database.
- Depending on the platform and the database system user, an attacker might carry

2. If you are not using a database access layer (DAL), consider using one. This will help you centralize the issue. You can also use ORM (*object relational mapping*). Most of the ORM systems use only parameterized queries and this can solve the whole SQL injection problem.

3. Locate all of the dynamically generated SQL queries and convert them to parameterized queries. (*If you decide to use a DAL/ORM, change all legacy code to use these new libraries.*)

4. Use your weblogs and application logs to see if there were any previous but undetected attacks to this resource.

Remedy

The best way to protect your code against SQL injections is using parameterized queries (*prepared statements*). Almost all modern languages provide built-in libraries for this. Wherever possible, do not create dynamic SQL queries or SQL queries with string concatenation.

Create a database user with the least possible permissions for your application and connect to the database with that user. Always follow the principle of providing the least privileges for all users and applications.

Required Skills for Successful Exploitation

There are numerous freely available tools to exploit SQL injection vulnerabilities. This is a complex area with many dependencies; however, it should be noted that the numerous resources available in this area have raised both attacker awareness of the issues and their ability to discover and leverage them.

HOPE YOU LIKE IT >>>>