SQL Injection in ResultSet.refreshRow() with malicious column names

Moderate davecramer published GHSA-r38f-c4h4-hqq2 on Aug 3

Package

org.postgresql:postgresql (Maven)

Affected versions	Patched versions
<42.2.26	42.2.26
<42.3.7	42.3.7
<42.4.1	42.4.1

Description

Impact

What kind of vulnerability is it? Who is impacted?

The PGJDBC implementation of the `java.sql.ResultRow.refreshRow()` method is not performing escaping of column names so a malicious column name that contains a statement terminator, e.g. `;`, could lead to SQL injection. This could lead to executing additional SQL commands as the application's JDBC user.

User applications that do not invoke the `ResultSet.refreshRow()` method are not impacted.

User application that do invoke that method are impacted if the underlying database that they are querying via their JDBC application may be under the control of an attacker. The attack requires the attacker to trick the user into executing SQL against a table name who's column names would contain the malicious SQL and subsequently invoke the `refreshRow()` method on the ResultSet.

For example:

```
CREATE TABLE refresh_row_example (
  id     int PRIMARY KEY,
  "1 FROM refresh_row_example; SELECT pg_sleep(10); SELECT * " int
);
```

This example has a table with two columns. The name of the second column is crafted to contain a statement terminator followed by additional SQL. Invoking the `ResultSet.refreshRow()` on a ResultSet that queried this table, e.g. `SELECT * FROM refresh_row`, would cause the additional SQL commands such as the `SELECT pg_sleep(10)` invocation to be executed.

As the multi statement command would contain multiple results, it would not be possible for the attacker to get data directly out of this approach as the `ResultSet.refreshRow()` method would throw an exception. However, the attacker could execute any arbitrary SQL including inserting the data into another table that could then be read or any other DML / DDL statement.

Note that the application's JDBC user and the schema owner need not be the same. A JDBC application that executes as a privileged user querying database schemas owned by potentially malicious less-privileged users would be vulnerable. In that situation it may be possible for the malicious user to craft a schema that causes the application to execute commands as the privileged user.

Patches

Has the problem been patched? What versions should users upgrade to?

Yes, versions 42.2.26, 42.3.7, and 42.4.1 have been released with a fix.

Workarounds

Is there a way for users to fix or remediate the vulnerability without upgrading?

Check that you are not using the `ResultSet.refreshRow()` method.

If you are, ensure that the code that executes that method does not connect to a database that is controlled by an unauthenticated or malicious user. If your application only connects to its own database with a fixed schema with no DDL permissions, then you will not be affected by this vulnerability as it requires a maliciously crafted schema.

Severity

Moderate 5.9 / 10

CVSS base metrics

Attack vector	Network
Attack complexity	High
Privileges required	Low
User interaction	Required
Scope	Unchanged
Confidentiality	Low
Integrity	Low
Availability	High

Availability	High

CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:H

CVE ID

CVE-2022-31197

Weaknesses

No CWEs

Credits

kato-sho