# SQL injection Alert

Last updated by | Soma Jagadeesh | Jan 11, 2021 at 11:19 AM PST

Contents

## Scenario

'An application generated a faulty SQL statement on database 'xxxxxxxxx'. This may indicate that the application is vulnerable to SQL injection.'

The alert indicates your application generated a faulty SQL statement in the database.

This may indicate a possible vulnerability to SQL injection attacks. There are two possible reasons for the generation of a faulty statement:

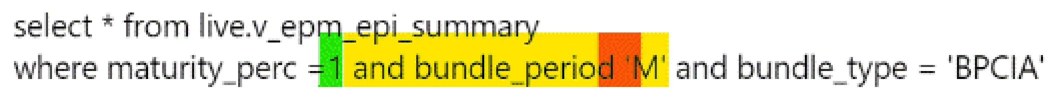
• A defect in application code that constructs the faulty SQL statement

• Application code or stored procedures don't sanitize user input when constructing the faulty SQL statement, which may be exploited for SQL Injection

The alert email that the customer get also contains link to Azure Security Center. In Azure Security Center, the customer can see details about the issue - for example, the query that triggered the alert. The customer can also examine his audit logs to get better understanding about what went wrong.

The email itself also contains details about the issue and some explanation about what happened and what can be done to mitigate issues like this. If something is not clear, there is also feedback button that gives us feedback about the alert, and we takes this feedback's seriously. I not aware of any missing data that prevent the customer from investigating the issues, but we are open for suggestions.

Example:



```
select * from live.v_epm_epi_summary
where maturity_perc =1 and bundle_period 'M' and bundle_type = 'BPCIA'
```

In the query above there's a syntax error where you're missing an equal operator (highlighted red)

The algorithm then assumes that some part of that query (for example the part highlighted in yellow, but it is impossible to know exactly without the logs) could have been injected by an attacker through a field in the application that sets the value of the maturity_perc parameter (highlighted green).

If it is indeed the case that this error can be caused by a user through your application, then your application is indeed vulnerable to SQL injection.

If it was caused by a bug/defect in the code and cannot be caused by a user then you need to:

1. Fix the code that's causing the error
2. Reset the algorithm state by disabling 'Advanced Threat Protection' in the Azure portal and then re-enabling it

## Classification

Root cause Tree - Security/User issue/error/Vulnerability assessment

**How good have you found this content?**