前言

Author: 0x584A

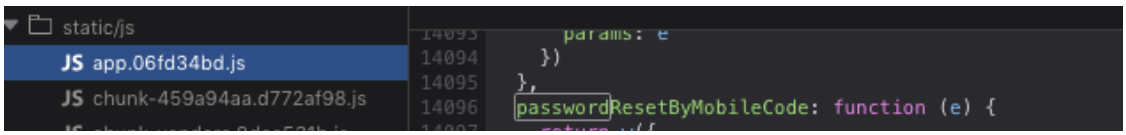
总结下在测试工作中，Web系统因前端文件导致的安全问题，通过分析前端代码找到路由配置，在对 PATH 进行调用，往往会有意想不到的结果出现。

- 在页面源代码中找注释，可能会存在敏感配置信息或功能代码说明
- JS代码中搜索关键字（setPass,password,router,path等），分析前端代码使用框架类型，查找路由配置信息及硬编码配置

示例

示例一

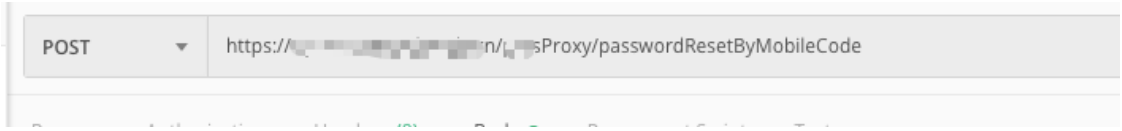
某日在渗透内部系统，在搜索 JS 代码时发现含有可疑URL接口路径，"xxxProxy/passwordResetByMobileCode"，尝试调用后提示：**请求类型错误需要POST提交、参数错误。**

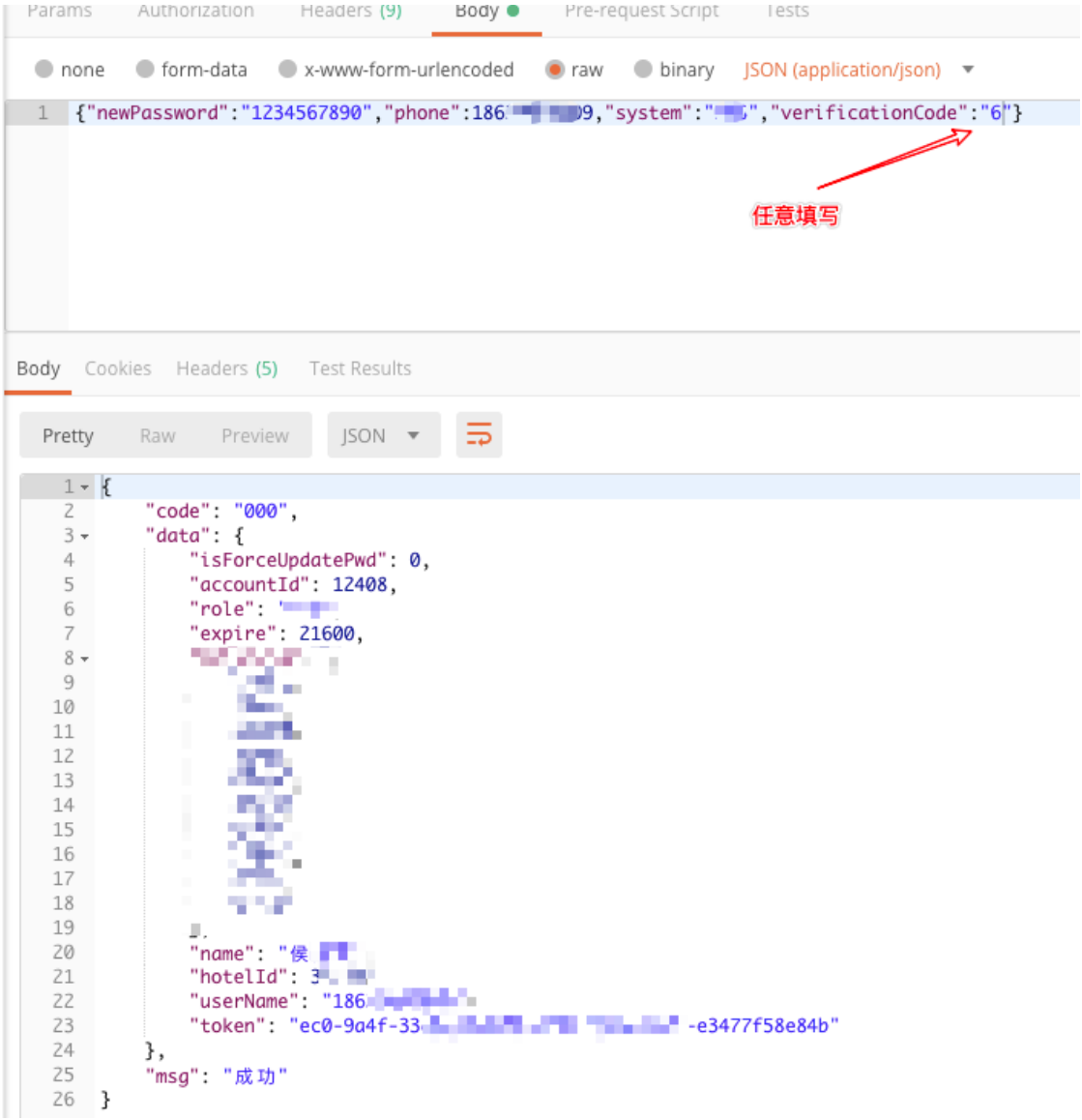


随后在找回密码页面（xxx/forgetPassword）发现加载了一个 'chunk-5df4b15a.ea1375be.js' 文件，其中存在上面看到的关键字 'passwordResetByMobileCode'，并且代码中含有疑似接口所需要的参数。

```
var t, e, n;
return l.a.wrap(function(o) {
    for (; ; )
        switch (o.prev = o.next) {
        case 0:
            if (t = {
                phone: this.$store.state.user.userName,
                verificationCode: this.$store.state.user.verificationCode,
                newPassword: this.formCustom.newPassword,
                system: "[redacted]"
            },
            e = this.$auth.login.getIsFirstLogin(),
            n = null,
            "1" === e)
                return o.next = 6,
                this.$service.user.passwordResetByToken(t);
            o.next = 9;
            break;
        case 6:
            n = o.sent,
            o.next = 12;
            break;
        case 9:
            return o.next = 11,
            this.$service.user.passwordResetByMobileCode(t);
        case 11:
            n = o.sent;
```

尝试构建所需要的参数提交，发现存在任意密码重置漏洞，关键的 **verificationCode** 参数后端无校验。





示例二

这个与上面的差不多，同样在分析前端 JS 代码中，发现一个关键字 URL：**/user/setNewPass**，浏览器访问后是设置新密码页面，填写密码后提示参数错误。

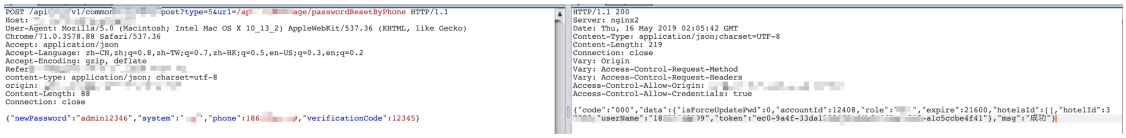
在重置密码页面 **/user/resetPass** 打开时，会加载了一个新的 JS 文件，分析代码并补全提交参数：phone、verificationCode。

```
if (!e) {
  var i = t.props,
  c = i.dispatch;
  i.login;
```

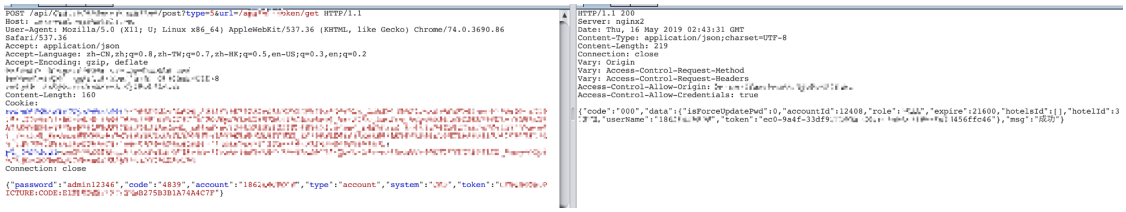
```
Object(E['g']) ({
  phone: a,
  verificationCode: o
}).then(function () {
  c({
    type: 'login/saveResetpassInfo',
    payload: {
      phone: a,
      verificationCode: o
    }
  }),
  v['a'].push({
    pathname: '/user/setNewPass',
    query: {
      type: 'forget'
    }
  })
}).catch (function (e) {
  console.log(e)
})
}
```

参数提交后发现并没有校验 **verificationCode** 的真实性，填写任意内容都可以重置成功。

账号原密 admin12345 重置成 admin12346：



随后用重制后的密码登录，提示成功：



示例三

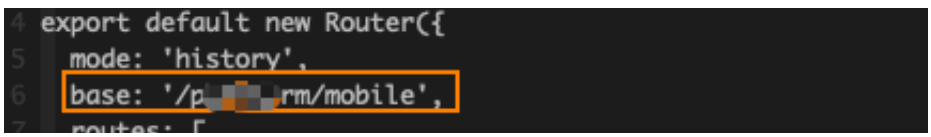
某天在 GITHUB 上发现可疑的存在生产环境域名的文件，指向某分析平台：

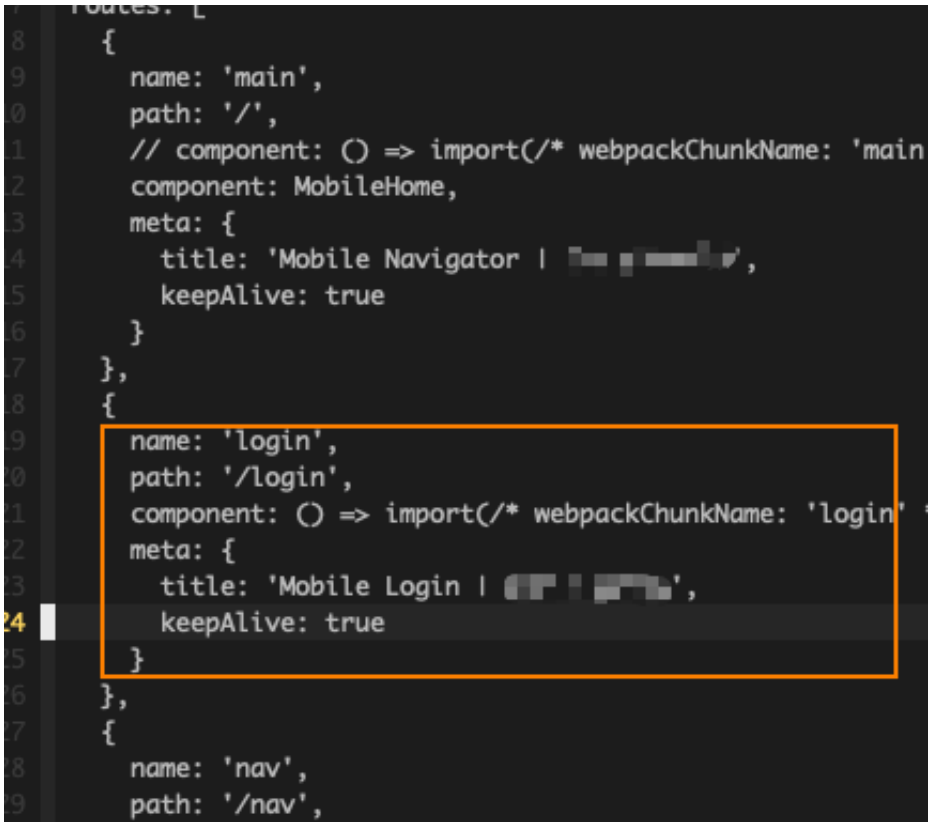
```
292  var url = encodeURIComponent(
293  // `http://[redacted]dingtalk/authCallBack?redirect_url=${redirectURL}`
```

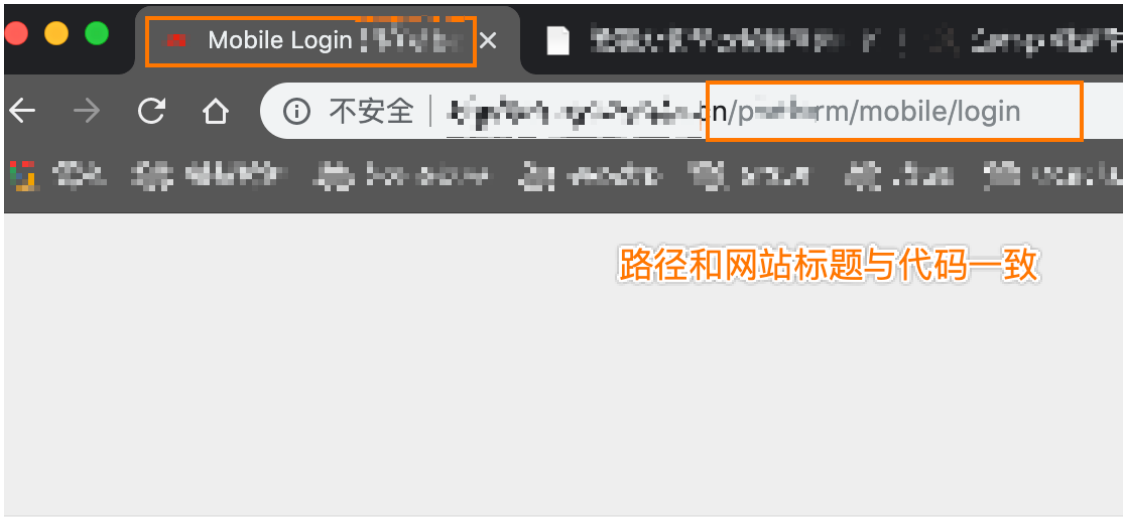
查看 github lnimpossible/xxx 项目后，确认为某分析平台前端产线代码。

项目为 vue.js 开发，查看配置文件发现与生产环境一致，URL含有 /pxxxform/ 路径：

寻找前端路由验证代码是否一致：



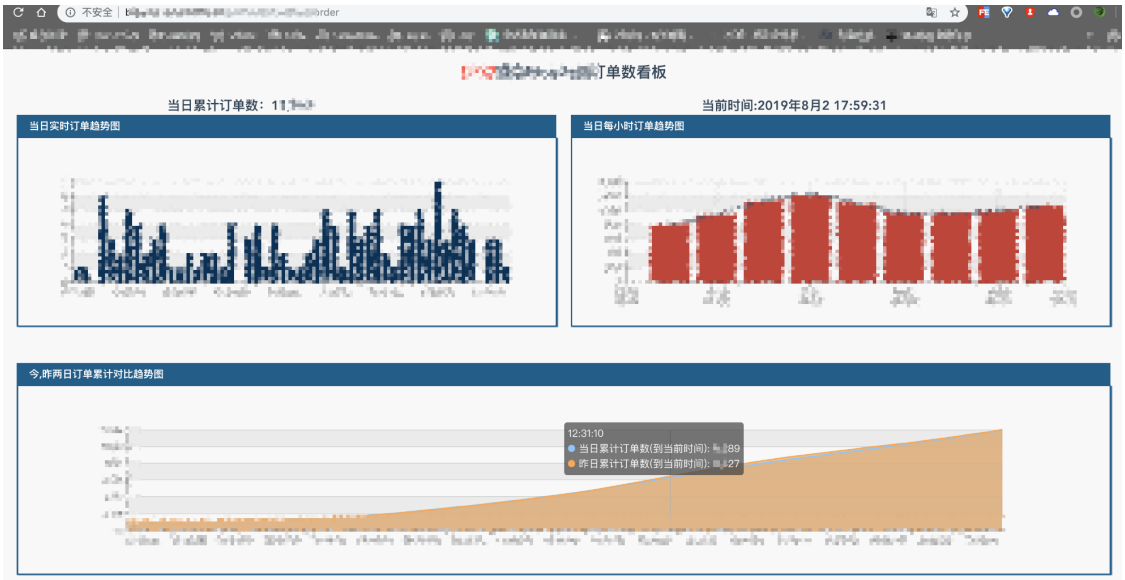




账号　　请输入手机号或姓名

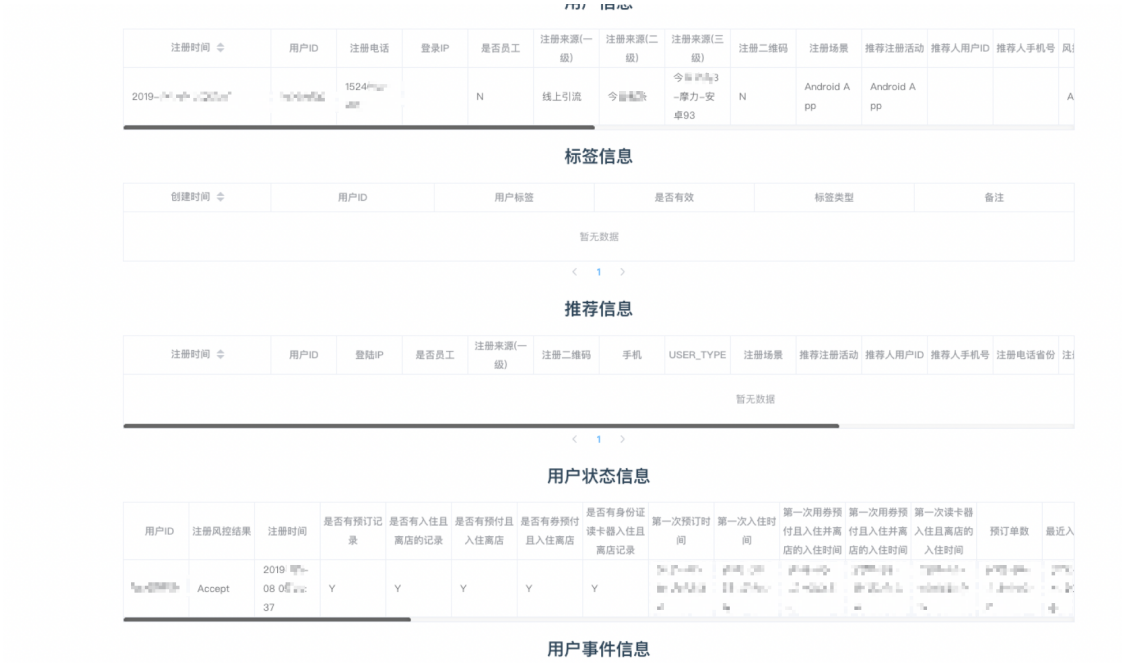
通过分析泄漏代码，汇总前端路由路径、API调用路径发现存在多个未授权访问。多个敏感页面可以在外网直接浏览（含住和离时间、联系方式及身份证信息）。

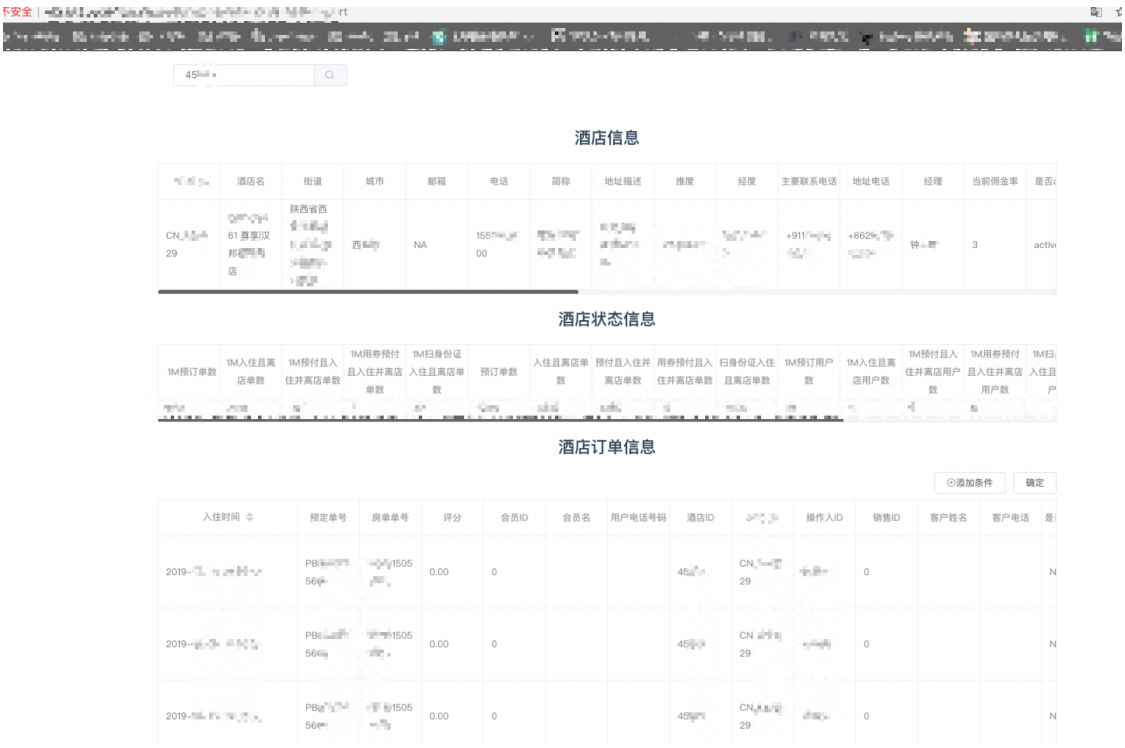

KPI Dashboard
ALL
BRN
URN OCC(%)
OverAll
42
East
44.69
South
42.5
SouthCentral
37.46
SouthWest
38.11
North
38.67
SRN
URN
BRN
西南区 BRN 108
brn
occ%
69
15
34


不安全
订单数看板
当日累计订单数：
当前时间:2019年8月2 17:59:31
当日实时订单趋势图
当日每小时订单趋势图
今,昨两日订单累计对比趋势图
12:31:10
当日累计订单数(到当前时间):
昨日累计订单数(到当前时间):
提示：
订单数对应的是Mobile端订单总数
每隔10秒更新每个报表


开始日期
至
结束日期

用户信息





联系到项目所属人为离职员工，已经联系其删除了该项目。

总结

在前后端分离开发时，常出现接口或页面的未授权访问、越权等问题。后端不能完全信任用户提交过来的参数，必须对参数及业务流程进行严格校验。