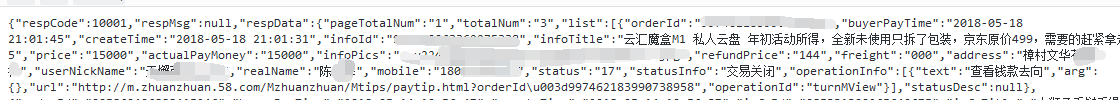
挖啊挖 没发现什么漏洞 挖不动啊 发现一处CORS也被限制死域名了

然后有一处api引起了我的注意 这个api是返回订单信息的

[https://api.\*\*\*\*\*\*\*.com](https://api.*******.com)/api/order/getSellerOrder



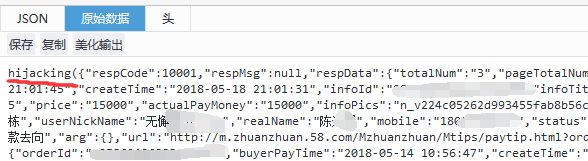
有着不少的敏感信息 想着能不能获取到这个数据呢 明显的有CORS但是被限制死了域名

返回的是json格式数据也不是jsonp

那开发人员会不会有种习惯 设计api的时候提供json和jsonp两种格式 而jsonp的函数名变量名就是callback

加多个参数 callback=hijacking试试看

果真变成jsonp了



既然如此 那么别的敏感接口是否也存在这种问题？

测试后发现不存在 继续思考

想到前辈们说过的话 主站不行 可以尝试子域名 子域名防护或许没那么好

果真子域名站点全站均存在问题jsonp劫持并且CORS限制不严格



通过jsonp劫持 能劫持到许多订单信息 然而这个订单信息中有一些是没有直接作用的，比如订单id、用户id、下单时间之类的，暂且放一边不管。

继续挖



尝试花了1块钱 租了个号 发现查看上号解锁码的请求是酱紫的



[https://aaaa.\*\*\*\*.com/zzopen/gameAccount/findOrderAccountInfo?arg={"orderId":"xx","uid":"xx"}](https://aaaa.****.com/zzopen/gameAccount/findOrderAccountInfo?arg=%7b%22orderId%22:%22xx%22,%22uid%22:%22xx%22%7d)



测试了下此处存在越权 恰好越权所需的两个值可通过上面的jsonp劫持到

这样形成了一个jsonp劫持+越权的组合拳

然后因为使用这web业务的用户少..审核询问我是否可以影响到大量用户，那要大量用户就只能app了，就去看了下app，由于见识短浅没找到什么好的利用方法，仅找到一处url跳转。

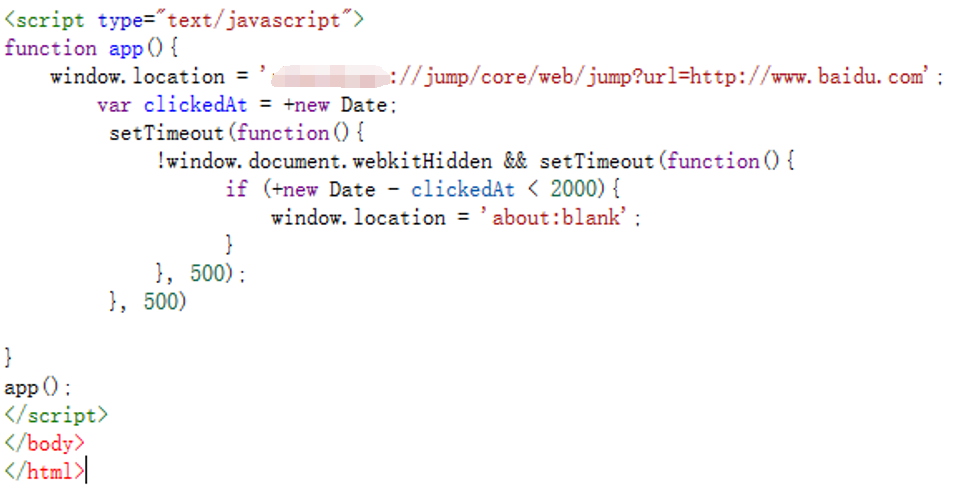
app的伪协议是 aaaa://

在APP里面 通过伪协议唤醒APP并打开指定URL是这样的

aaaa://jump/core/web/jump?url=<http://doamin.com>

由于对url参数未做校验 导致可在APP内置浏览器内跳转到任意域

poc:



打开后唤醒了app并打开了百度

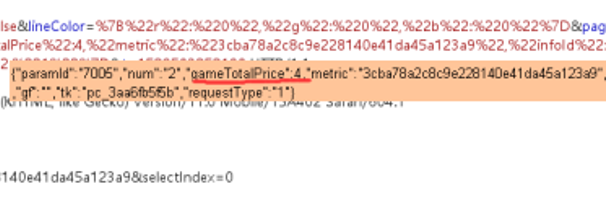


接着走..既然是交易网站 那自然就得多关注关注支付方面的问题了

浏览商品点击购买时 会弹出个二维码 提示用微信扫码下单

生成二维码的时候是这样一个请求 看到请求中传递了价格参数

果断修改成1试试看



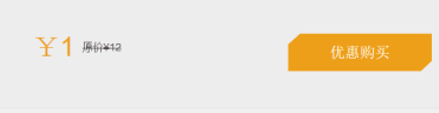
扫描修改后的二维码惊讶的发现价格真的变了！！

但是人生就是这么大起大落 点击确认下单后就被打回了原形



之后又到处点啊点 发现了一处疑似是官方出售商品的区域

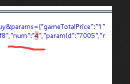
一个一个商品试啊试 发现这个1元体验4小时吃鸡



一样的是生成二维码扫码购买

这里有个num参数值为4 上面商品描述也是1元体验4小时

那这个4对应的估计就是租用时长 修改成24试试



扫码确认下单 发现价格依旧是1元 支付后发现到期时间就变成了24小时

可以畅玩吃鸡拉



在这个网页版上面 购买商品是通过

网页浏览选中商品 微信扫码购买 后端检测到用户下单网页跳转到等待支付页面 若支付成功再跳转到订单详情处

那在这个环节中 发现即使我未登录 也是可以浏览商品 依然是微信扫码购买

同样的下单后 也会跳转到等待支付处 但是这里已经从未登录状态变成了登录状态



那么我就可以通过以欺骗用户下单的方式 登录用户的账号