

浅谈web身份认证类漏洞

原创 雪狼别动队 酒仙桥六号部队

2020-10-09原文

这是 酒仙桥六号部队 的第 85 篇文章。

全文共计4048个字，预计阅读时长13分钟。

前言

在渗透测试的刀光剑影中，web身份认证是我们绕不开的一个常见问题，这一类问题其实就是我们经常说的逻辑问题，同时也是我们在提交渗透测试报告或者挖SRC出现漏洞点比较多的漏洞类型，网上关于身份认证的逻辑问题的文章其实很多，但也有一些不全面，或者说对一些新出现的认证问题没有阐述，本文希望能够在前人的基础上对身份认证类逻辑漏洞进行总结并以自己渗透测试和靶机等遇到的各种认证可绕过问题进行补充达到帮助渗透测试人员少踩一些弯路的作用。其中有一些常规的内容一笔带过。

登录框中的漏洞

1. 暴力破解用户名和密码

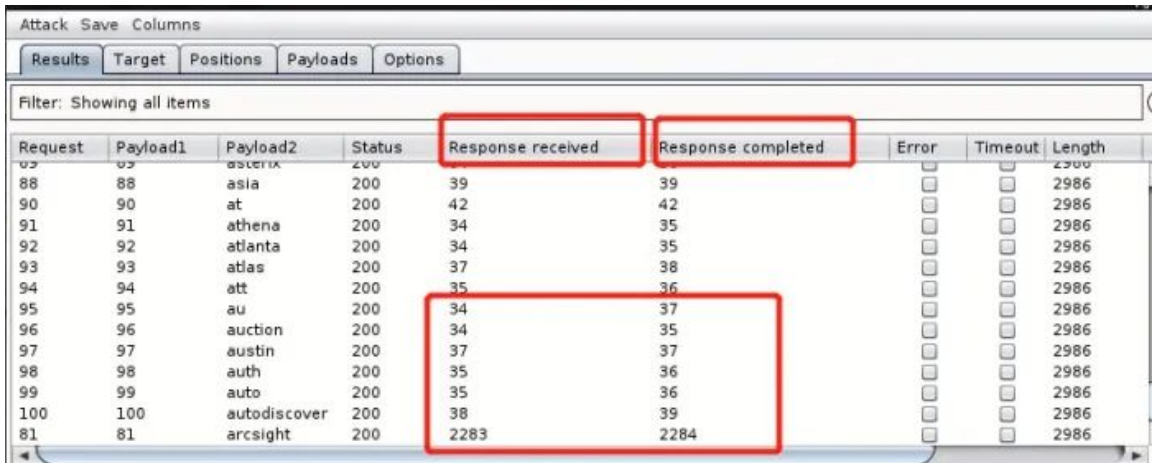
这是我们在做渗透测试中经常遇到的漏洞类型，暴力破解用户名和密码我们仅需要将burp中的intruder模块中的attack type调整到Cluster bomb即可进行暴力破解。

2. 用户名枚举、密码可爆破

这类问题也是我们在做渗透测试最喜欢的问题，可以枚举用户名，并通过特定账号暴力破解密码。

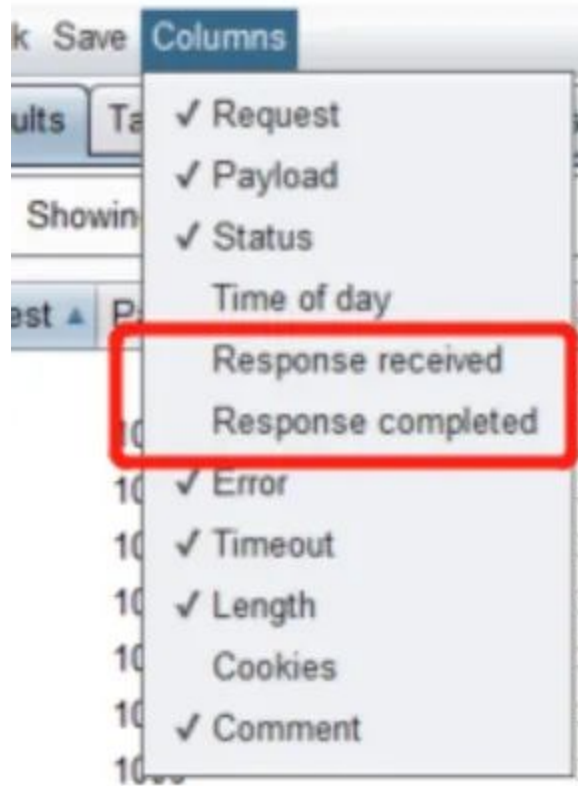
遇到用户名枚举相信我们大多数人遇到这种问题常做的就是检查返回包中的是否出现：用户名不正确等字眼或者通过其响应状态即响应码进行对比。

还有一种就是通过响应时间，这种类型经常是返回包中出现：用户名或密码不正确等字眼，这时我们也可以进行枚举用户。在我们将密码设置为很长的一段之后（最好这样做），通过枚举用户时所产生的响应时间的不同，达到枚举用户名的目的。



| Request | Payload1 | Payload2 | Status | Response received | Response completed | Error | Timeout | Length |
|---------|----------|--------------|--------|-------------------|--------------------|--------------------------|--------------------------|--------|
| 87 | 87 | asia | 200 | 39 | 39 | <input type="checkbox"/> | <input type="checkbox"/> | 2986 |
| 88 | 88 | asia | 200 | 39 | 39 | <input type="checkbox"/> | <input type="checkbox"/> | 2986 |
| 90 | 90 | at | 200 | 42 | 42 | <input type="checkbox"/> | <input type="checkbox"/> | 2986 |
| 91 | 91 | athena | 200 | 34 | 35 | <input type="checkbox"/> | <input type="checkbox"/> | 2986 |
| 92 | 92 | atlanta | 200 | 34 | 35 | <input type="checkbox"/> | <input type="checkbox"/> | 2986 |
| 93 | 93 | atlas | 200 | 37 | 38 | <input type="checkbox"/> | <input type="checkbox"/> | 2986 |
| 94 | 94 | att | 200 | 35 | 36 | <input type="checkbox"/> | <input type="checkbox"/> | 2986 |
| 95 | 95 | au | 200 | 34 | 37 | <input type="checkbox"/> | <input type="checkbox"/> | 2986 |
| 96 | 96 | auction | 200 | 34 | 35 | <input type="checkbox"/> | <input type="checkbox"/> | 2986 |
| 97 | 97 | austin | 200 | 37 | 37 | <input type="checkbox"/> | <input type="checkbox"/> | 2986 |
| 98 | 98 | auth | 200 | 35 | 36 | <input type="checkbox"/> | <input type="checkbox"/> | 2986 |
| 99 | 99 | auto | 200 | 35 | 36 | <input type="checkbox"/> | <input type="checkbox"/> | 2986 |
| 100 | 100 | autodiscover | 200 | 38 | 39 | <input type="checkbox"/> | <input type="checkbox"/> | 2986 |
| 81 | 81 | arcsight | 200 | 2283 | 2284 | <input type="checkbox"/> | <input type="checkbox"/> | 2986 |

Response received 和 Response completed可在columns中找到。



找到我们响应完成时间与其他用户名有很大差距的就是我们所枚举出来的用户名，接着可以用此用户名暴力破解密码。

tips:防暴力破解的设计逻辑问题

(1)、当遇到封IP的情况

You have made too many incorrect login attempts. Please try again in 1 minute(s).

这是在暴力破解的时候经常会遇到的情况，许多人遇到这种情况会跳过继续寻找下一个漏洞点，然而这个地方也是可以绕过的。

这边说三个可能会用到的方法：

X-Forwarded-For 绕过

这边简单提一下XFF、XFH（凑字数）

X-Forwarded-

For: 用于通过HTTP代理或负载均衡器识别连接到web服务器的客户端的原始IP地址的事实上的标准报头。当客户端和服务端之间的通信被截获时，服务器访问日志仅包含代理或负载均衡器的IP地址。要查看客户端的原始IP地址，X-Forwarded-For请使用请求标头。

X-Forwarded-

Host报头是用于识别由客户机在所要求的原始主机一个事实上的标准报头Host的HTTP请求报头。反向代理的主机名和端口（负载均衡器，CDN）可能与处理请求的原始服务器不同，在这种情况下，X-Forwarded-Host标头可用于确定最初使用的主机。

这种绕过方法最为简单，只需在请求包中增加XFF报头即可，并将其也设置为一个intruder payload即可。

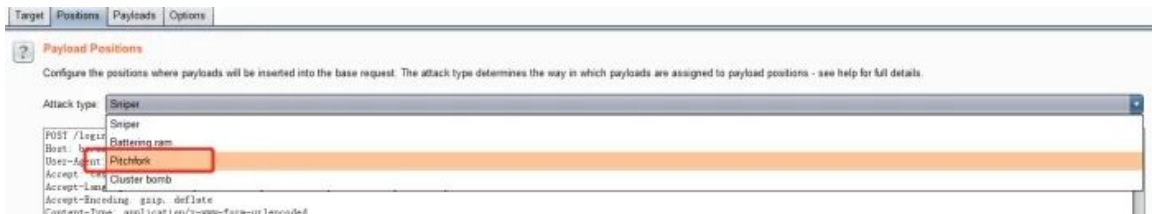
正确用户与需要暴力破解的用户交替出现（或者错误账号稍微多几次）

这种情况的出现是可能是因为服务端只对失败次数进行了校验，当登录成功之后失败次数清0。

生成用户名字典：baymax为已知密码的用户名，wil为需要破解的用户名。

baymax
wil
wil
baymax
wil
wil
baymax
wil
wil
baymax

将burp的爆破模式调为：



生成的密码也要一一对应，比如以下这样（123456为正确用户名的密码，其余的是wil需要爆破的密码）：

```
123456  
qewerw  
sadf  
123456  
fdghjh  
ewfrdgfh  
123456  
wergtd|
```

短时间请求次数太多封锁IP，但当出现以json格式的请求包的情况可以这样绕过限制：

Request

Raw Params Headers Hex

```
10 Content-Length: 86
11 Connection: close
12 Cookie: session=hashGANHtlun6FalFDC4oLCDWAmwFard
13
14 {
    "csrf": "aeU0hGR98V4QRtMfV0oTo4PI6bQctJeD",
    "username": "s0mm3r",
    "username": "carlos",
15    "password": [
16        "123456",
17        "password",
18        "qwerty",
19        "123456789",
20        "12345",
21        "1234",
22        "111111",
23        "1234567",
24        "dragon",
25        "123123",
26        "baseball",
27        "abc123",
28        "football",
29        "monkey".
```

(2)、锁定账号

同一用户登录次数过多，将账户封锁一段时间，比如出现以下情况：

Request Response

Raw Headers Hex Render

```
49 <section>
50 <p class=is-warning>
    You have made too many incorrect login attempts [redacted] in 1 minute(s).
51 </p>
52 <form class=login-form method=POST>
53 <input required type="hidden" name="[redacted]" value="1q560MBaSh">
    <label>
        Username
    </label>
```

0 matches In Pretty

可通过建立用户名并确定一个很小的出现频率最高的密码字典，选择的密码数量不得超过允许的登录尝试次数。例如，如果您确定限制为3次尝试，则最多选择3个密码。自己有一次做渗透测试便是通过这种方法拿到密码的。

3. SQL注入、XSS、CSRF

这个相信我也不需要多说，在登录框这边可能存在SQL注入、XSS和CSRF的。

4. 抓包把password字段修改成空值发送

这个比较少见，我也只是从其他人博客上见到过。

5. 参数的FUZZ

1) 任意后台地址，手动添加参数admin=1之后，发现返回了cookie，成功登陆。

2) 普通用户时，可手动添加一些参数，说不定权限会提高。

6. 信息泄露

这个基本上是老生常谈的问题，有时在源码中会泄露一些信息，也有时在JS文件中找到令人惊喜的点。比如自己曾经碰到一个网站，将密码加密算法ase加密信息写在了JS文件中，直接将整个密码字典加密后爆破。更有甚者可以通过JS暴露的接口未授权访问。

二因素 (2FA) 身份认证中的漏洞

2FA又称为双因子认证、二因素，是指结合密码以及实物（信用卡、手机、令牌或指纹等生物标志）两种条件对用户进行认证的方法，他的优点就是比单纯的密码登录安全得多。就算密码泄露，只要手机或其他实物等还在，账户就是安全的。不过也有缺点，费时且麻烦，且账户恢复也是双因素认证最大的麻烦问题。

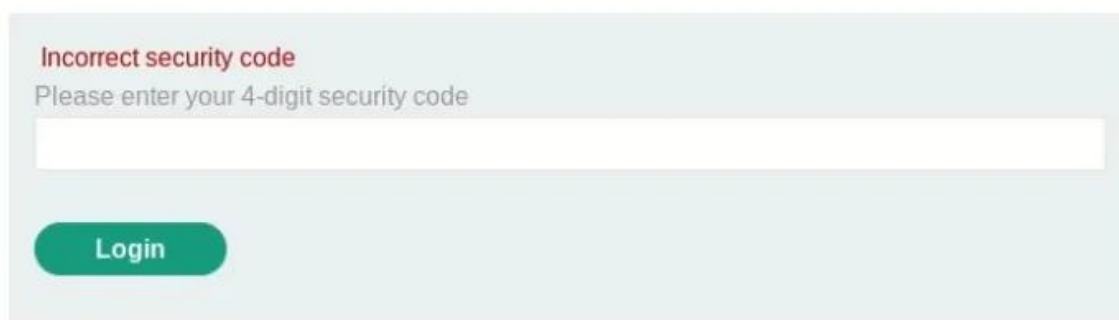
1. 虚假的2FA验证漏洞

当我们使用账户密码登录、出现输入安全口令时，这个时候其实我们已经“登录”进来了，只需要开启万能的F12大法（简言之便是绕过客户端校验类型）便可绕过

2. 有缺陷的2FA验证逻辑

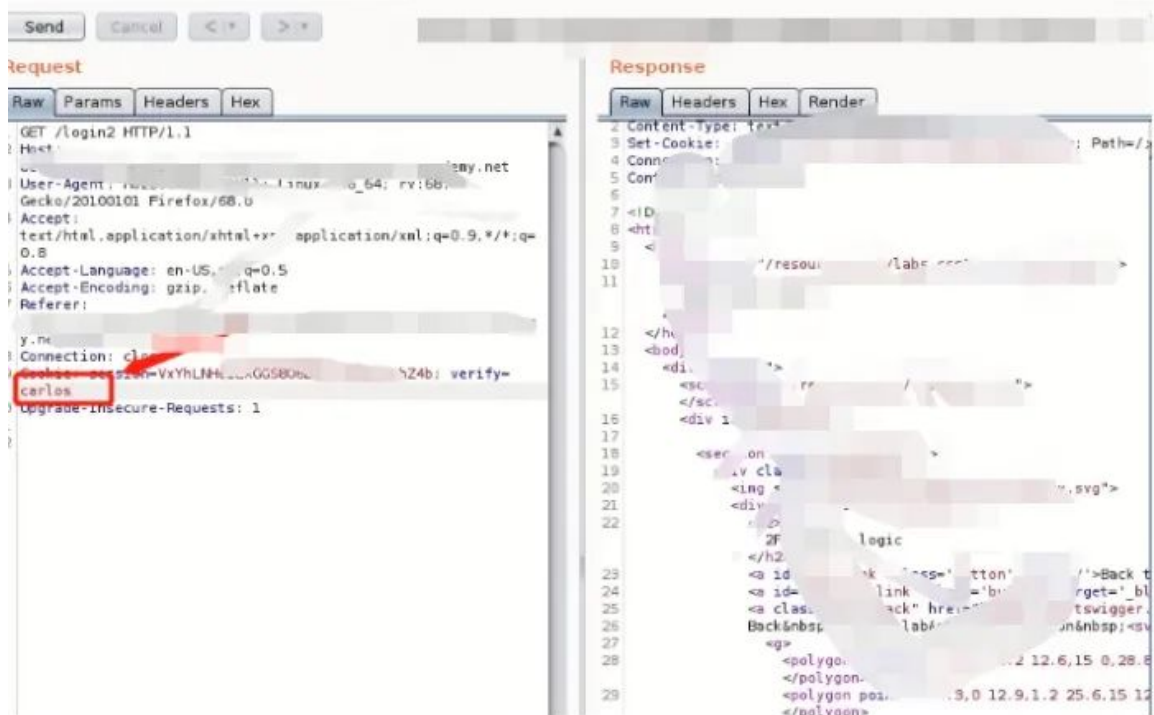
当carlos用户完成初步的登陆时，但是服务器在校验输入安全口令时存在设计上的逻辑错误。

出现安全口令的地方：

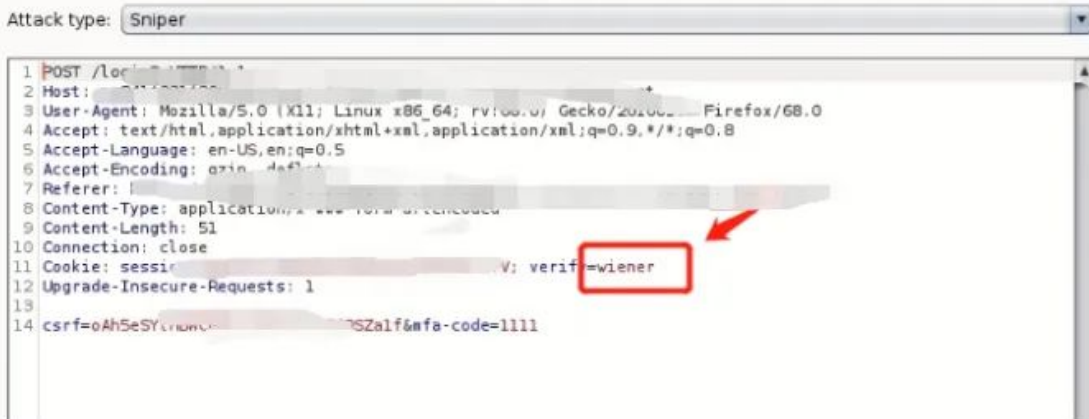


The screenshot shows a light blue login form. At the top, it displays the error message "Incorrect security code" in red. Below this, it says "Please enter your 4-digit security code" in a lighter blue font. There is a white input field for the security code. At the bottom of the form, there is a green rounded button labeled "Login".

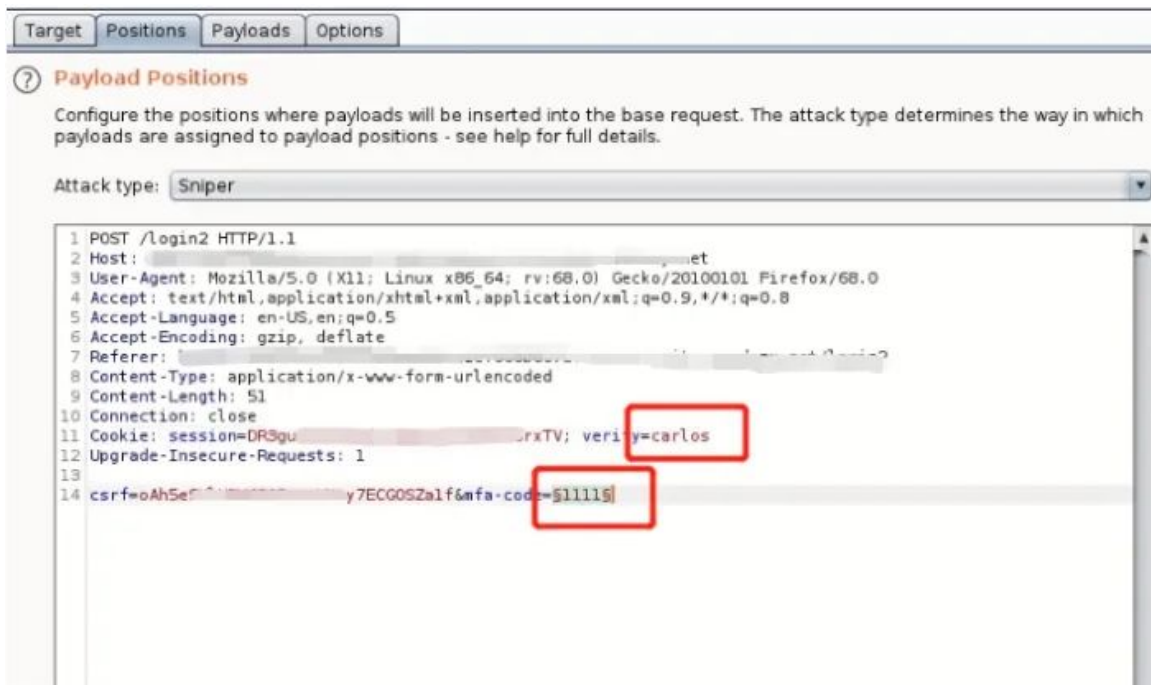
此时我们抓包发现，他在cookie中直接暴露了用户名，此时尝试将其修改为wiener并发送到intruter：



payloads are assigned to payload positions - see help for full details.
payloads are assigned to payload positions - see help for full details.



对四位数安全口令进行爆破可尝试登录wiener。



如果登录成功，可越权其他用户。

3. 暴力破解2FA验证码或者利用已公开的漏洞（如：CVE-2019-15617）

这是我在freebuf中看到的一篇文章（<https://www.freebuf.com/vuls/219813.html>），目标系统的动态口令OTP（One Time Password）出现验证机制不当，任何人都可以通过简单的暴力枚举来绕过它。

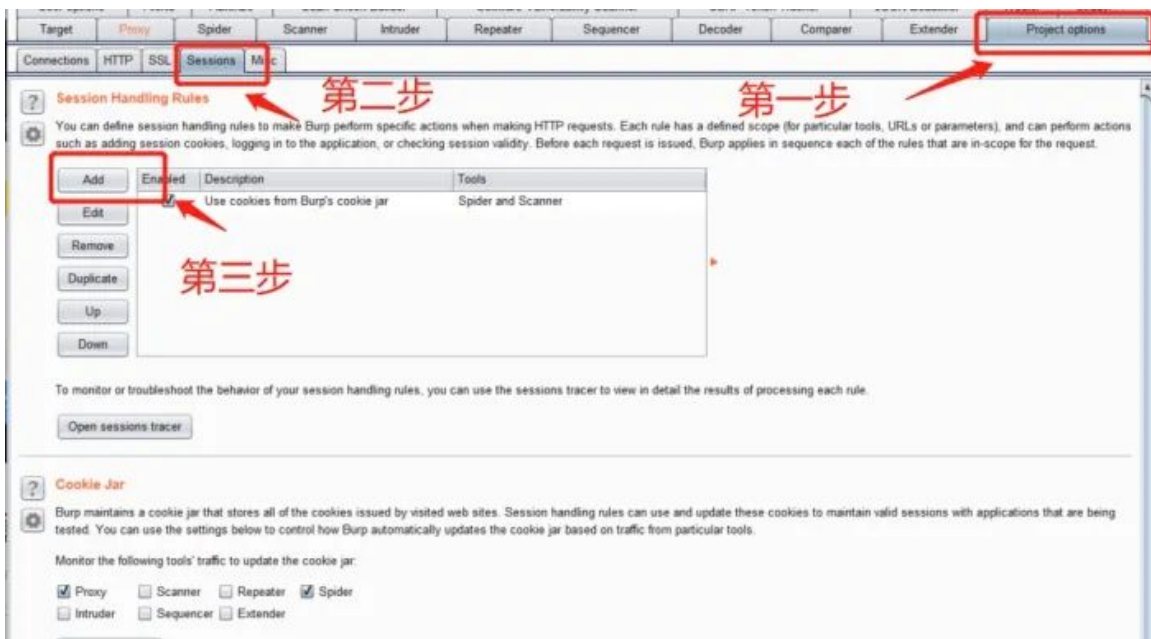
4. 绕过cookie+次数验证的2FA验证码

这是在一个漏洞环境中发现的一个新的思路，觉得有意思就将其写下来，当我们两次输入错误的安全口令后，我们所登录的帐号将会进行注销，因此我们需要使用burp的宏功能绕过这种机制。

例如账号的登录逻辑为：



当两次安全口令登录失败后又会跳转到 GET /login 登录界面。我们针对这种情况可以开启宏功能对其自动登录。开启宏操作如下：



Session handling rule editor

Details Scope

? **Tools Scope**
Select the tools that this rule will be applied to.

- Target
- Scanner
- Repeater
- Spider
- Intruder
- Sequencer
- Extender
- Proxy (use with caution)

? **URL Scope**
Use the configuration below to control which URLs this rule applies to.

- Include all URLs
- Use suite scope [defined in Target tab]
- Use custom scope
- Use advanced scope control

Include in scope

| Enabled | Prefix |
|---------|--------|
|---------|--------|

Add Edit Remove

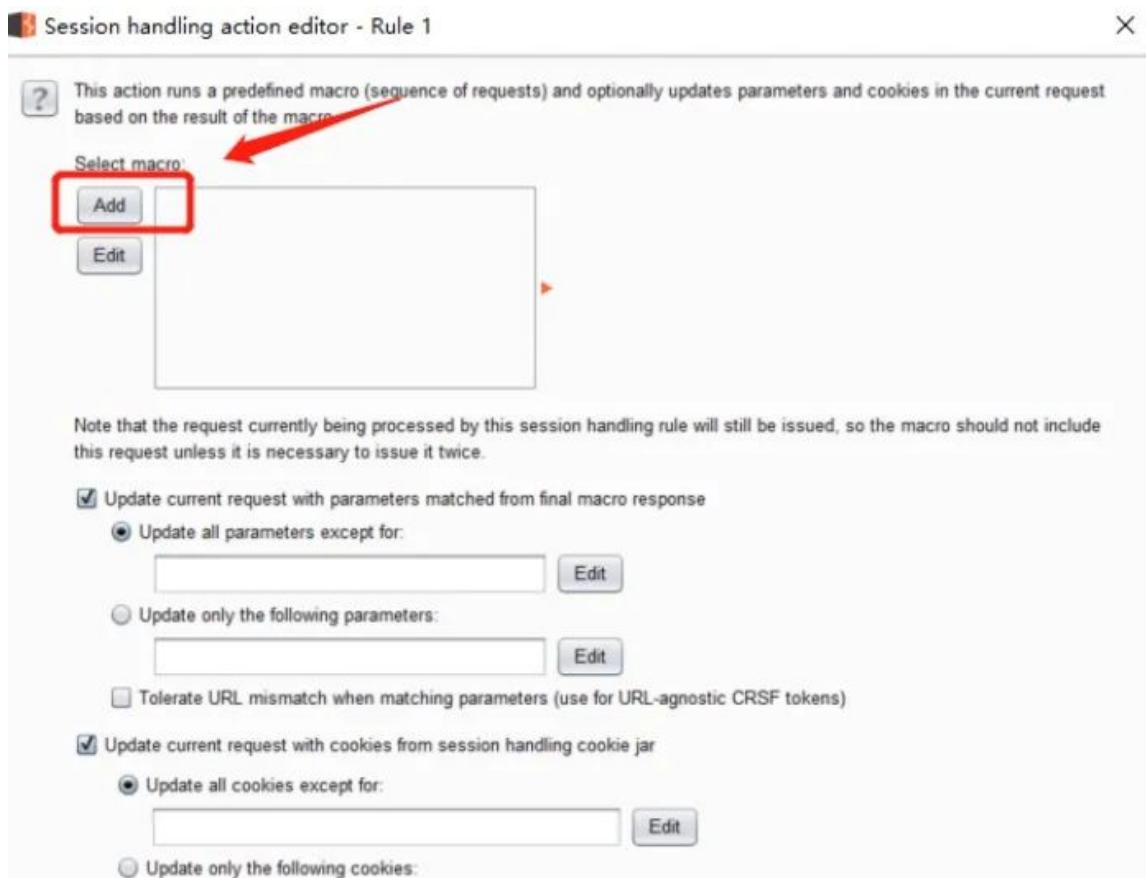
Session handling rule editor

Details Scope

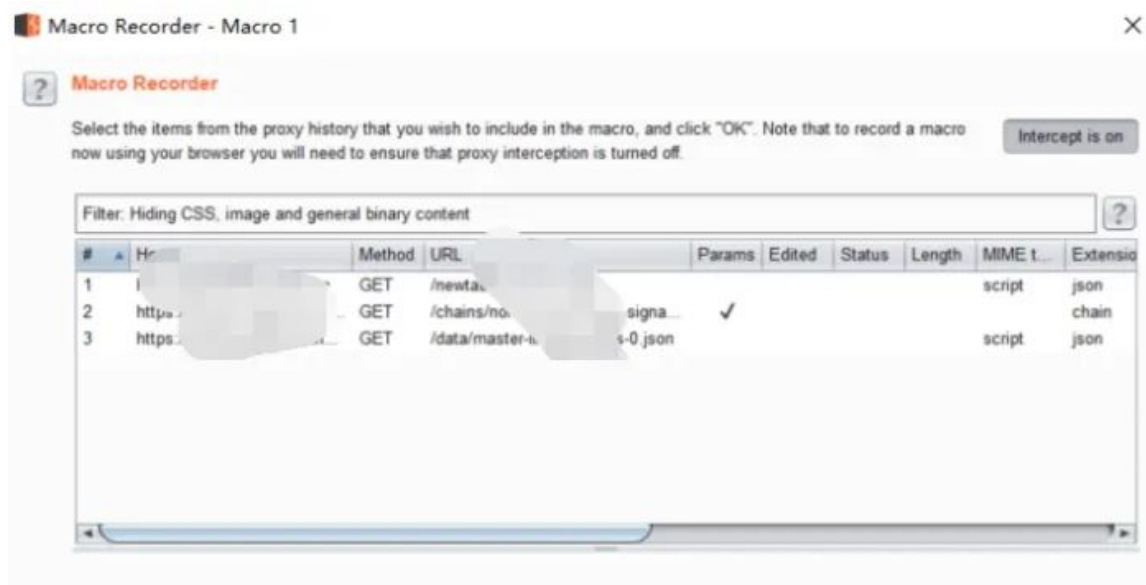
? **Rule Description**
Rule 1

? **Rule Actions**
The actions below will be performed in sequence when this rule is applied to a request.

| Add | Enabled | Description |
|-------------------------------------|--------------------------|--|
| <input type="checkbox"/> | <input type="checkbox"/> | Use cookies from the session handling cookie jar |
| <input type="checkbox"/> | <input type="checkbox"/> | Set a specific cookie or parameter value |
| <input type="checkbox"/> | <input type="checkbox"/> | Check session is valid |
| <input type="checkbox"/> | <input type="checkbox"/> | Prompt for in-browser session recovery |
| <input checked="" type="checkbox"/> | <input type="checkbox"/> | Run a macro |
| <input type="checkbox"/> | <input type="checkbox"/> | Run a post-request macro |
| <input type="checkbox"/> | <input type="checkbox"/> | Invoke a Burp extension |



点击add之后会自动弹出一个框：



由于我没有这个环境就只将其简单说一下，依次选中我们带有 GET /login 、 POST /login 、 GET /login2 的URL，之后一路点击OK即可。

验证漏洞

(1) 验证码不失效：找回密码判断时，仅判断验证码是否正确，没有对验证码的过期时间进行限制。

(2) 验证码未校验：输入手机号和验证码进行重置密码的时候，仅对验证码是否正确进行了判断，未对该验证码是否与手机号匹配做验证。

(3) 验证码可直接返回：发送验证码后，可以通过抓包等方式得到正确的验证码，直接填写验证码跳转更改密码页面。

(4) 可直接修改接收的手机或邮箱。

(5) 本地验证可绕过：客户端在本地进行验证码是否正确的判断，主要是根据接收到验证成功或验证失败的包判断是否验证成功，而该判断结果也可以在本本地修改，最终导致欺骗客户端，进入密码修改界面。

(6) 验证码易识别。

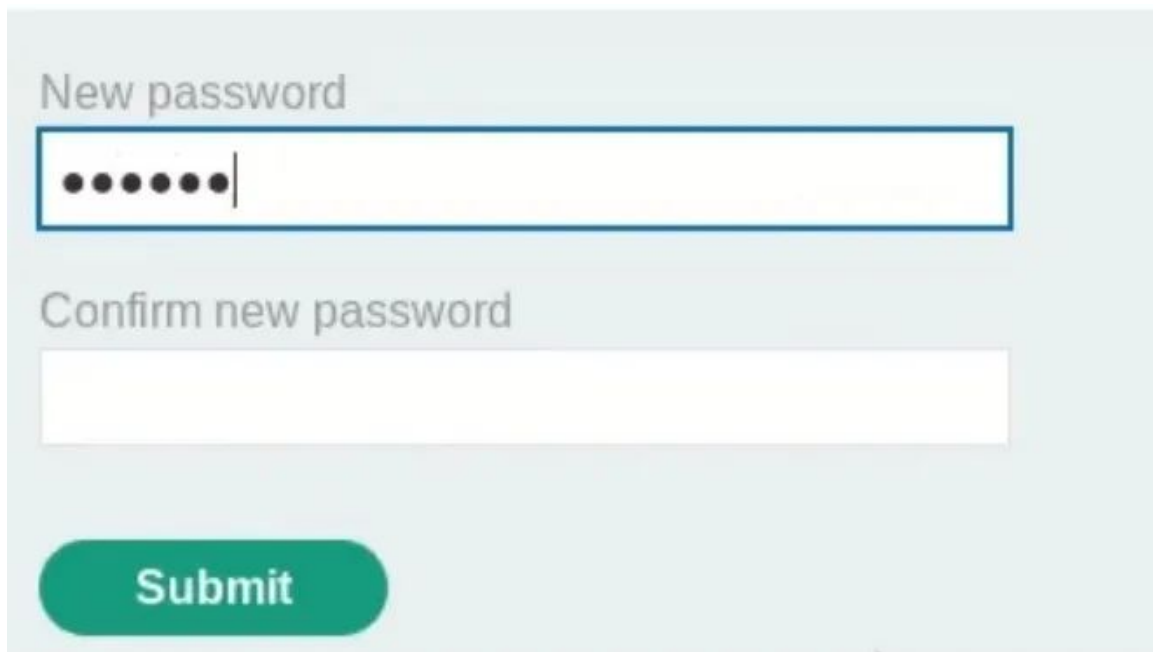
其他漏洞

1. 重置密码：

(1) URL重置密码

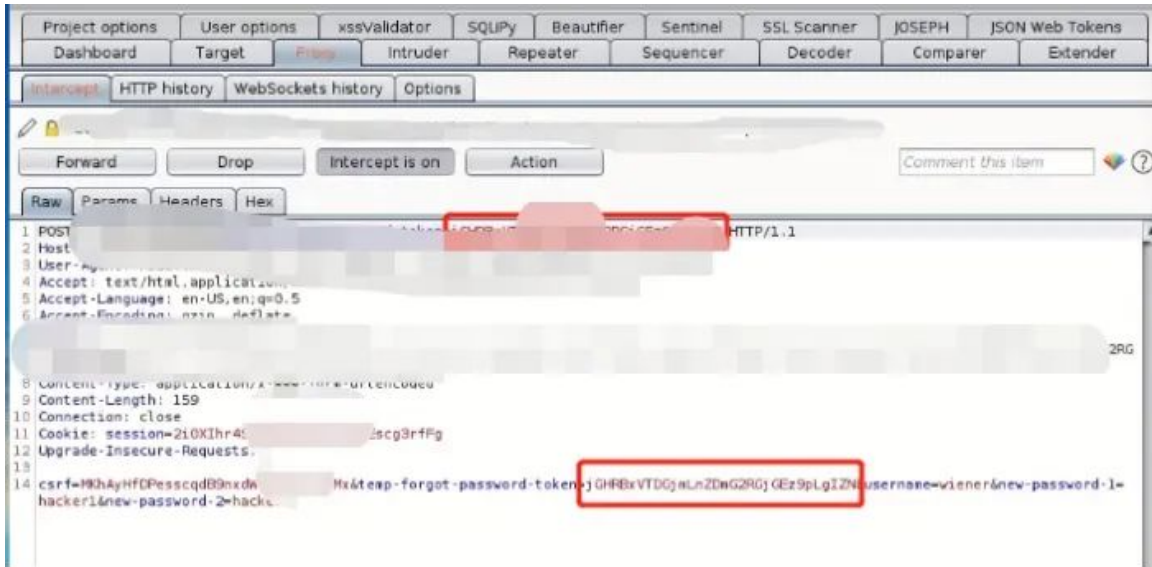
一般是在找回密码的时候，会给指定邮箱发送一个用于校验的url链接，链接中一般肯定会存在2个比较重要的参数，一个是用户名（或

者uid、qq邮箱之类的能代表用户身份的参数），另一个就是一个加密的字符串（通过服务器端的某种算法生成的用来验证用户身份的参数）。然后用户在重置密码的时候，点击邮箱中的重置密码链接，就可以重置帐号密码了。这种设计产生重置密码漏洞的情况，一般是由于重置密码链接中的表示用户名的参数和用于校验的加密字符串参数没有进行一一对应，导致可以被黑客偷梁换柱，从而重置密码。也就是说，那个验证身份的加密字符串是万能的。

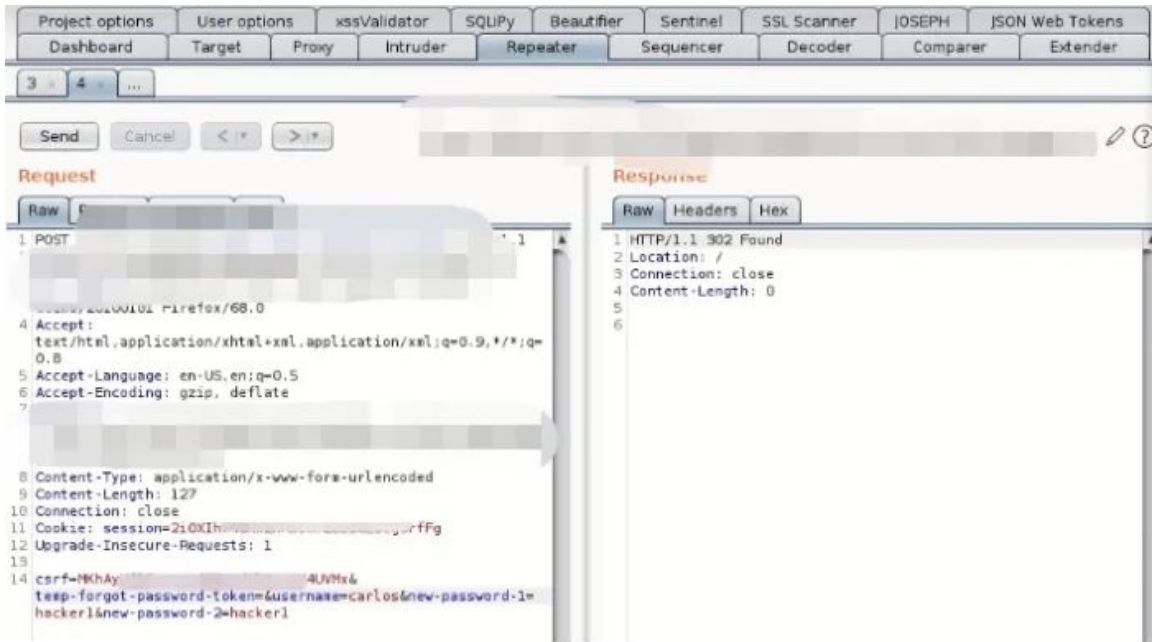


The image shows a web form for resetting a password. It has a light blue background. At the top, the text "New password" is displayed in a grey font. Below it is a text input field with a blue border, containing six black dots and a vertical cursor line. Underneath the first field is the text "Confirm new password" in a grey font, followed by an empty text input field. At the bottom of the form is a green rounded rectangular button with the word "Submit" written in white.

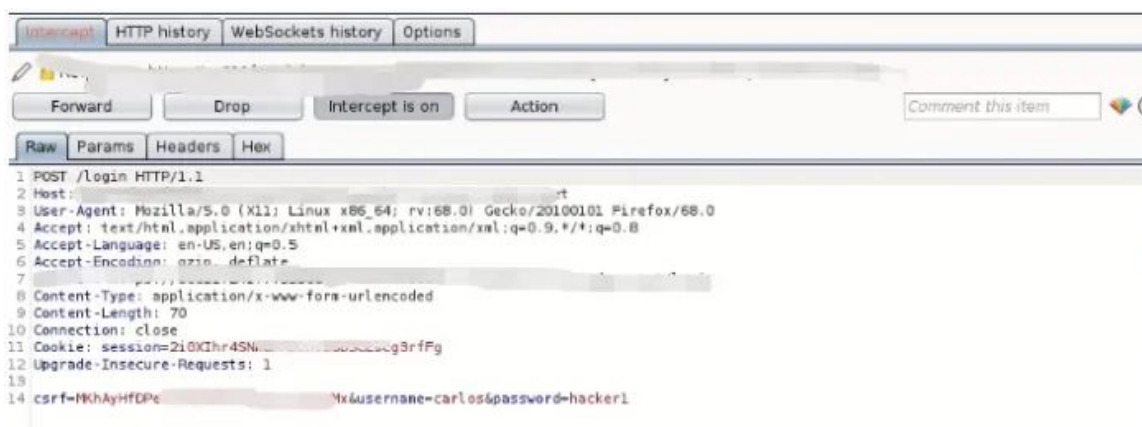
找到重置密码并抓包后发现token是万能的：



将用户名修改为其他已知的用户并删除token:



最后转到登录页面会发现重置密码成功!



除此之外加密字符串还可能是加密算法过于简单而被破解，导致密码重置。这种一般都是一些简单的加密算法，将一些关键参数比如用户名、邮箱、手机号、验证字符、时间戳等，进行一定的规则的组合然后进行md5、base64加密。

(2) 毒化重置密码

这个是在burp实验室中看到的。当要进行重置密码时，用户通常会输入其用户名或电子邮件地址，不久之后，他们会通过电子邮件收到重设URL。如果此URL是动态生成的，则有时可能容易受到密码重置中毒的攻击。简而言之就是我们输入用户名之后重置密码，通过拦截其请求并将host标头修改为自己的VPS，当用户点击重置密码的电子邮件地址后，自己的VPS日志中会出现带有动态token的重置密码链接，将URL重新修改为原URL访问即可：

初始的请求如下：

```
POST /password-reset HTTP/1.1
```

Host:原host地址

...

account=受害者邮箱

攻击者可以将Host改为自己的VPS地址：

```
POST /password-reset HTTP/1.1
```

Host:攻击者VPS

...

account=受害者邮箱

受害者可能会收到如

下重置密码URL:

<http://攻击者VPS/password-reset?token=xxxxxxx>

当受害者访问此URL之后便可在自己的VPS日志中看到后面的token值达到我们重置密码的目的。

(3) 修改响应状态，绕过修改密码。

当返回包出现true/false、0/1等信息可以尝试进行修改，可能会有出人意料的结果。曾经遇到过一个案例，当尝试修改返回包code值时，将其从0到9进行遍历，当code值为7的时候不用找回密码便直接登录了。。

(4) F12大法

有的重置密码可能会有以下步骤1. 输入账户名2. 验证身份3. 重置密码4. 完成。当你进行了1之后，直接去修改URL或者前端代码去进行3步骤从而也能成功的绕过了2的验证去重置密码。

(5) 有时候修改密码的时候会给邮箱或者手机发送一个新密码，那么抓包将手机号或者邮箱改成我们自己的可能会绕过其认证方式。

2. 用户注册

(1) 用户枚举

通过返回包中的信息判断用户是否注册。

(2) 任意用户注册

没有对用户信息进行校验。

(3) XSS、SQL注入

(4) 短信轰炸

这种在注册中比较常见，不断repeater即可，如果目标站增加了一些防护，可以加一些换行符等特殊字符绕过。

(5) 重复注册他人账号

当注册时显示用户已经注册了，我们可以在新用户注册的时候抓包，更改用户信息，可能会覆盖原用户并重新注册。

总结

身份认证类问题其实更多的是逻辑问题，只要我们足够细心，在关键地方进行尝试，说不定就可以得到意想不到的结果。现在在挖掘中更多的是思路的寻找，而不仅仅局限于以上总结。有些漏洞也是通用的，不仅仅出现在登录框等页面，比如在重置密码处说的修改响应状态，完全可以在登录或者注册时套用。同时也要注意组合漏洞的使用，漏洞之间的相互组合也能达到绕过身份认证的目的。

参考链接：

https://blog.csdn.net/chest_/article/details/103941792#%E7%94%A8%E6%88%B7%E6%B3%A8%E5%86%8C

<https://www.freebuf.com/articles/web/217052.html>

<https://www.secpulse.com/archives/136720.html>

.....



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

精选留言

用户设置不下载评论

