使用burp-cph插件对对动态参数爆破

某天我像往常一样渗透着网站，但是遇到下面这样的一个场景……

作者：天启 @涂鸦智能安全实验室

0x01 背景

某天我像往常一样渗透着网站，但是遇到下面这样的一个场景，有一个登陆框如图 1 所示，没有验证码，我使用 burpsuite 抓取一下登陆时的所有的请求数据包，发现最主要的请求是两个，如图 2 所示

图 1 登录框

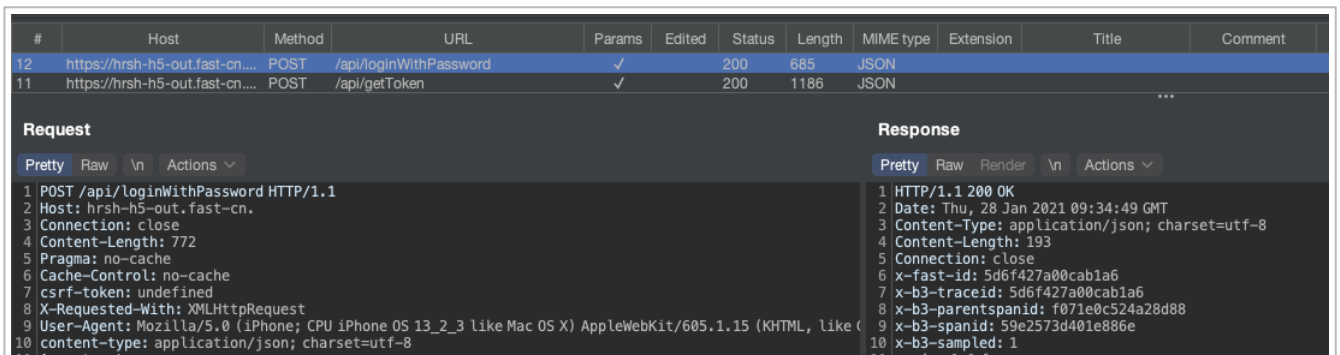


图 2 登录时最主要的两个请求

通过多次试错登陆并且抓取数据包发现，每一个 loginWithPassword 请求中的 token 和 publicKey 都是动态变化的，并且这两个参数都是来自于上一个 getToken 请求之中。我继续试错了几次密码，发现系统并没有锁定策略，这时第一个出现在我脑海的想法是这个登录点其实可以爆破的，但是有一个很现实的问题是，我怎么获取上一个请求的响应包的值作为下一个请求的请求体呢？这时候经过一番搜索终于找到一个看似能解决我这个问题场景的办法：burpsuite 宏。

0x02 引入一个 可能很多人都没听过的概念 burp 的宏

安全从业人员几乎天天跟 burpsuite 打交道，但是真正知道 burpsuite 宏的应该是少数。我们看一下 burpsuite 的官方定义：A macro is a sequence of one or more requests。 You can use macrosto the application，obtaining anti–csrf tokens，etc(宏是一个或多个请求的序列，你可以将宏应用到会话中来获取绕过 csrf 的 token)。翻译成白话文就是：你可以动态获取 token，然后绕过 csrf 的限制，那我们下面就用一个实际的例子来讲解一下这个 Macros 是如何工作。

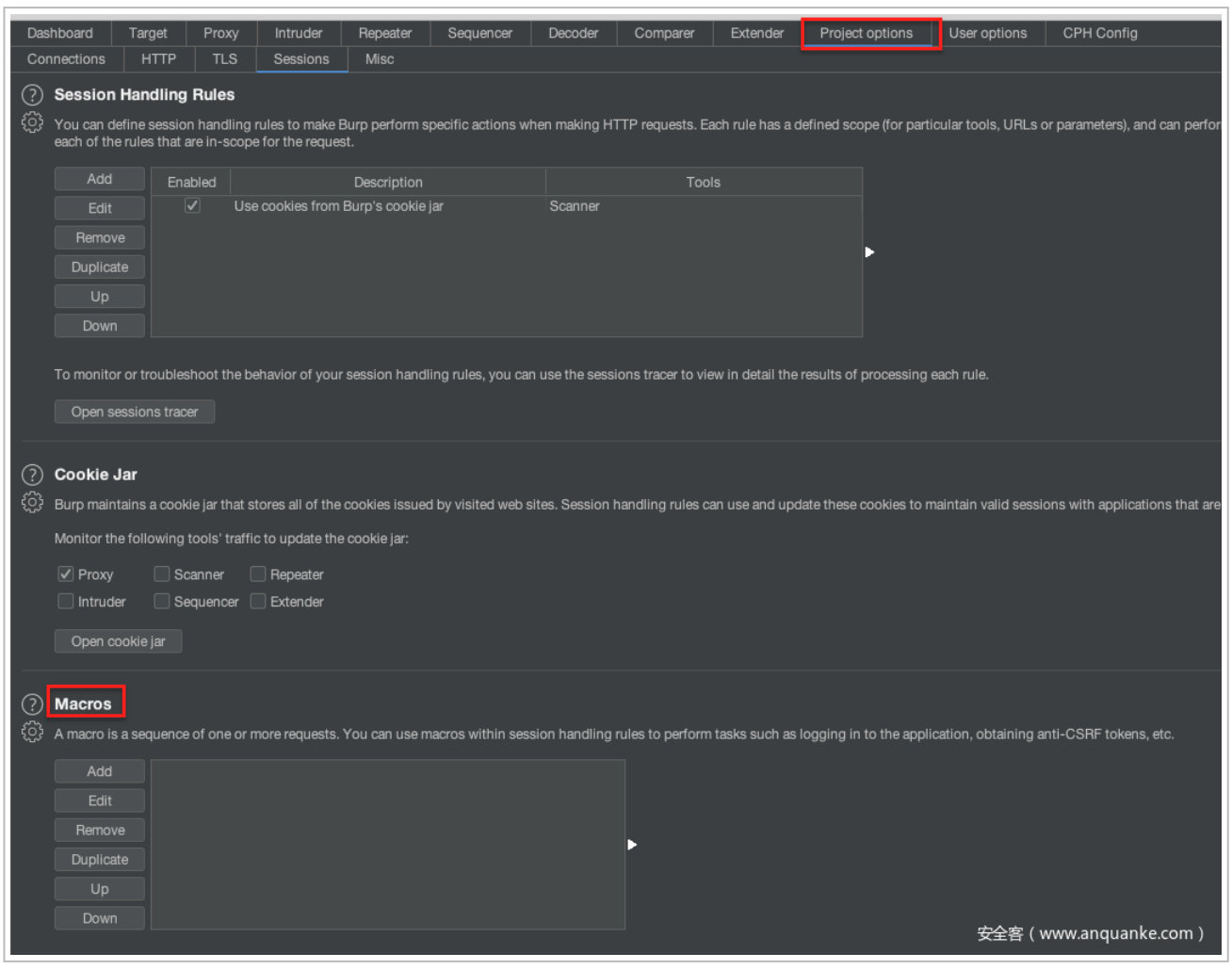


图 3 宏在 burpsuite 中的位置

我直接使用网上已经搭建好的实验环境进行测试，https://vuln–demo.com/burp_macro/macro.php。当点击 submit 时候，发现页面直接显示 token 不匹配，那说明此时请求的时候，一定向目标服务器发送了 token，只不过这个 token 是错误的 token，那正确的 token 在哪呢？这个小游戏的具体规则是什么呢？我们接着往下看！

图 4 实验测试环境

此时我们抓取一下点击 submit 时的数据包，并且全局搜索一下 token，看一下这个 token 出现在哪些地方？我们清晰地看到响应中存在着 token，那是不是意味着我们只要拿着本次响应中的 token 作为下一次的请求的参数就可以请求成功了？试一下看看吧！

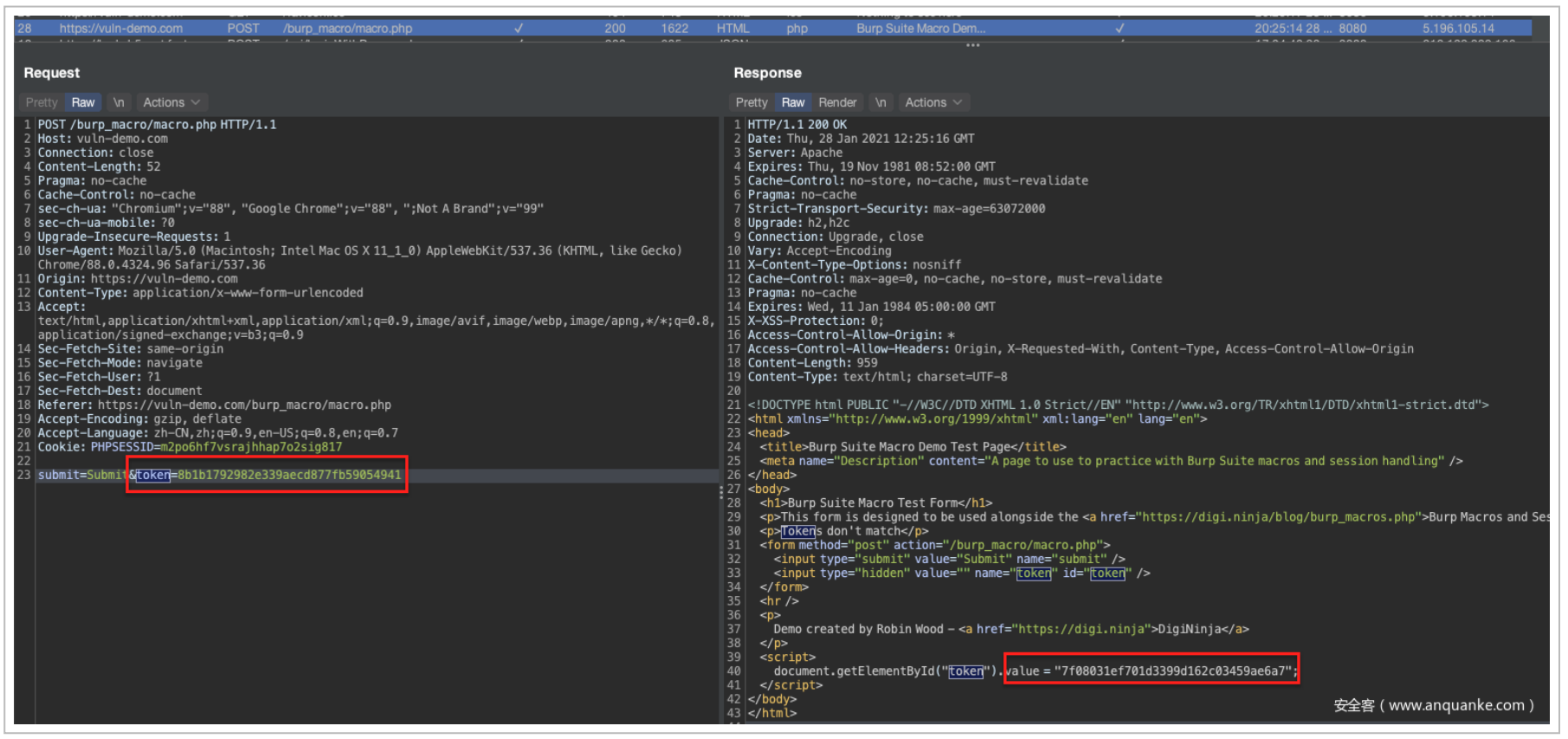


图 5 一个正常的请求

我们发现只要使用上一个请求响应中的 token 作为下一个请求的参数就能请求成功！至此规则清楚了，那如何使用 burpsuite 的宏来帮助我们自动完成这个工作呢？

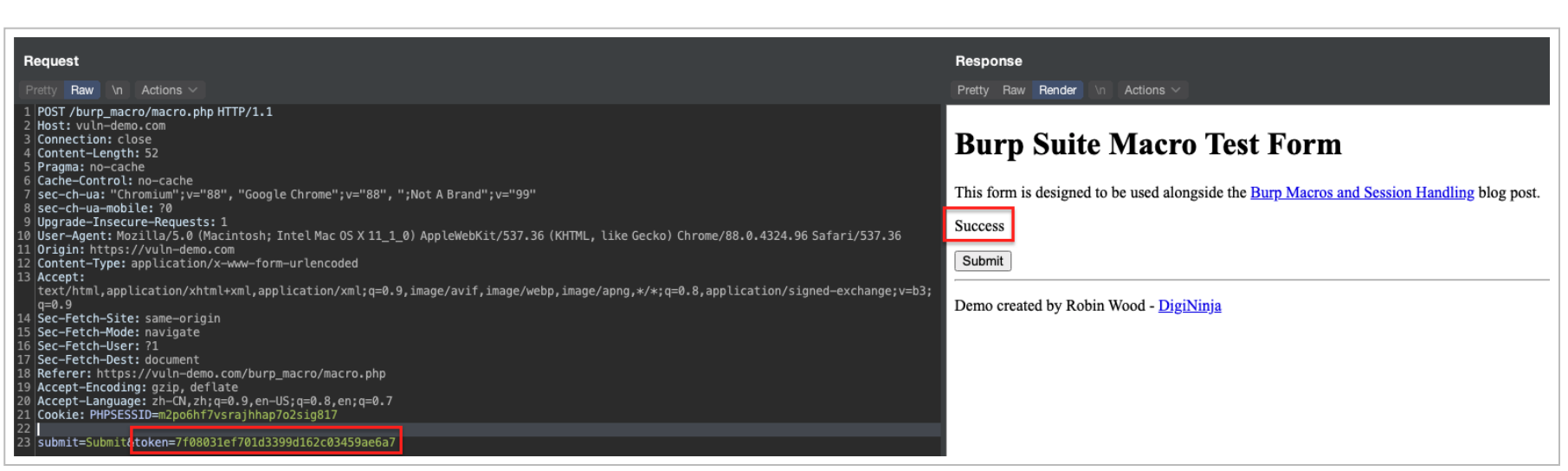


图 6

使用宏需要我们两步操作 1 创建宏 macros 2 创建一个使用宏的规则，接下来我们就用实例来讲解一下如何创建一个 macros 并且使其生效。

第一步创建宏

找到 token 所在请求的记录

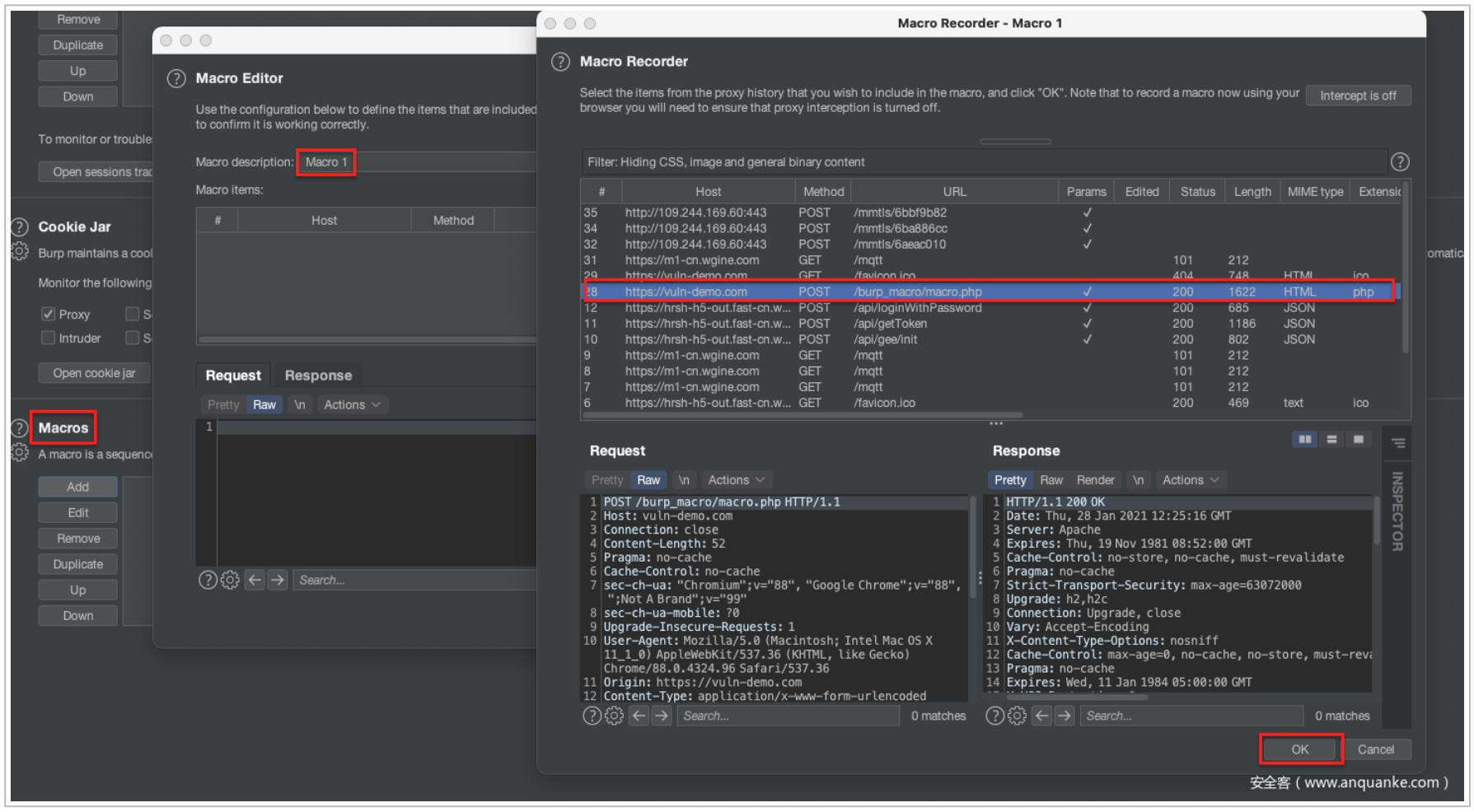


图 7.1

点击 Configure item 进行配置，“submit”和“token”不需要修改，默认即可

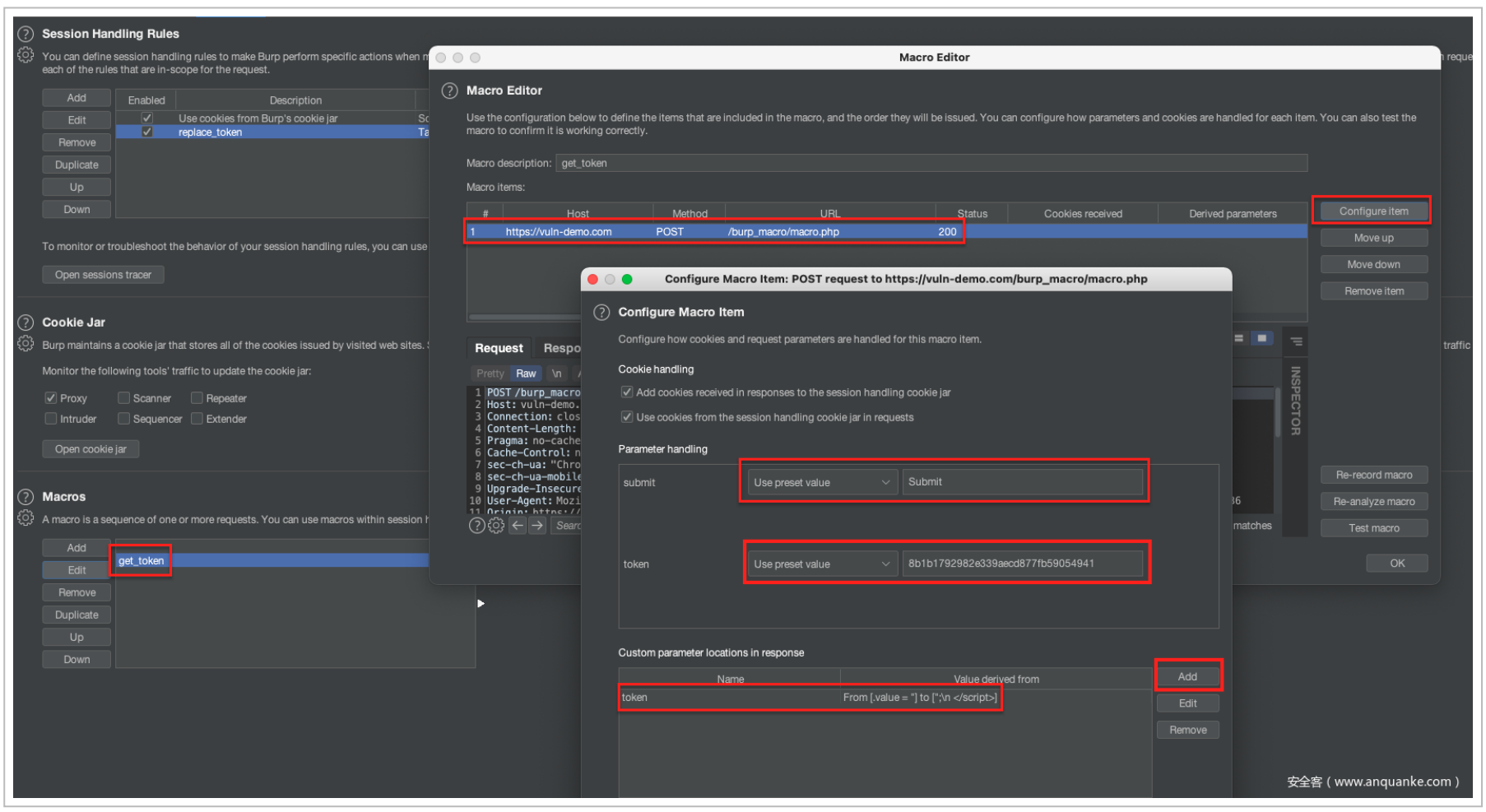


图 7.2

此时给需要动态替换的参数 token 设置规则，参数名字 token 要和请求中参数名字一字不差，因为 burpsuite 的宏就是根据这个名字进行动态替换的。然后鼠标双击 value 值 burpsuite 会自动生成正则进行匹配，然后点击 OK 即可

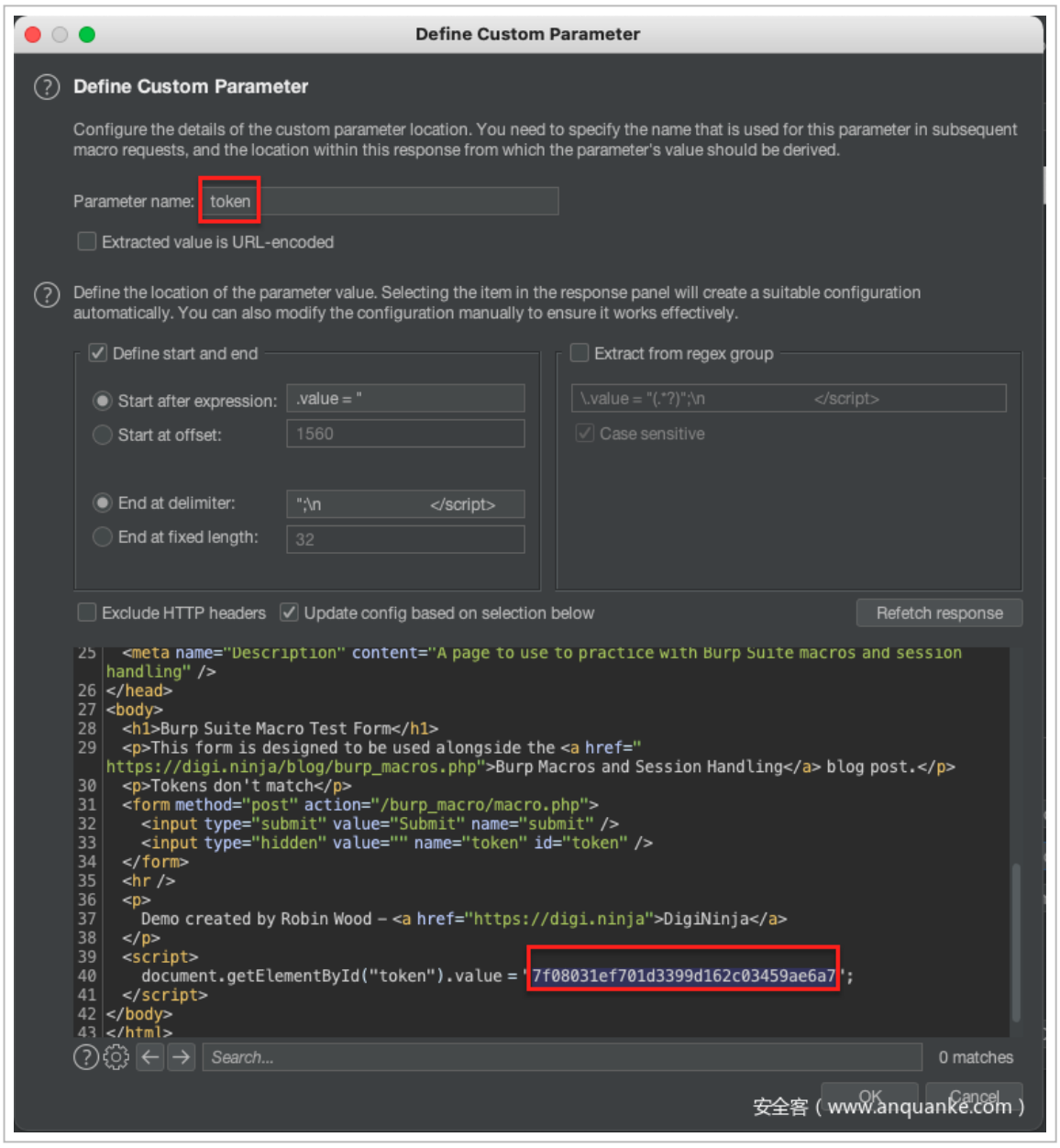


图 8

第二步创建引用宏的规则

点击 Add 创建一个规则 replace_token(可以随便取)，点击 Rule Action 下面的 add，然后选中“run macros”之后再选中 get_token 点击 OK 即可

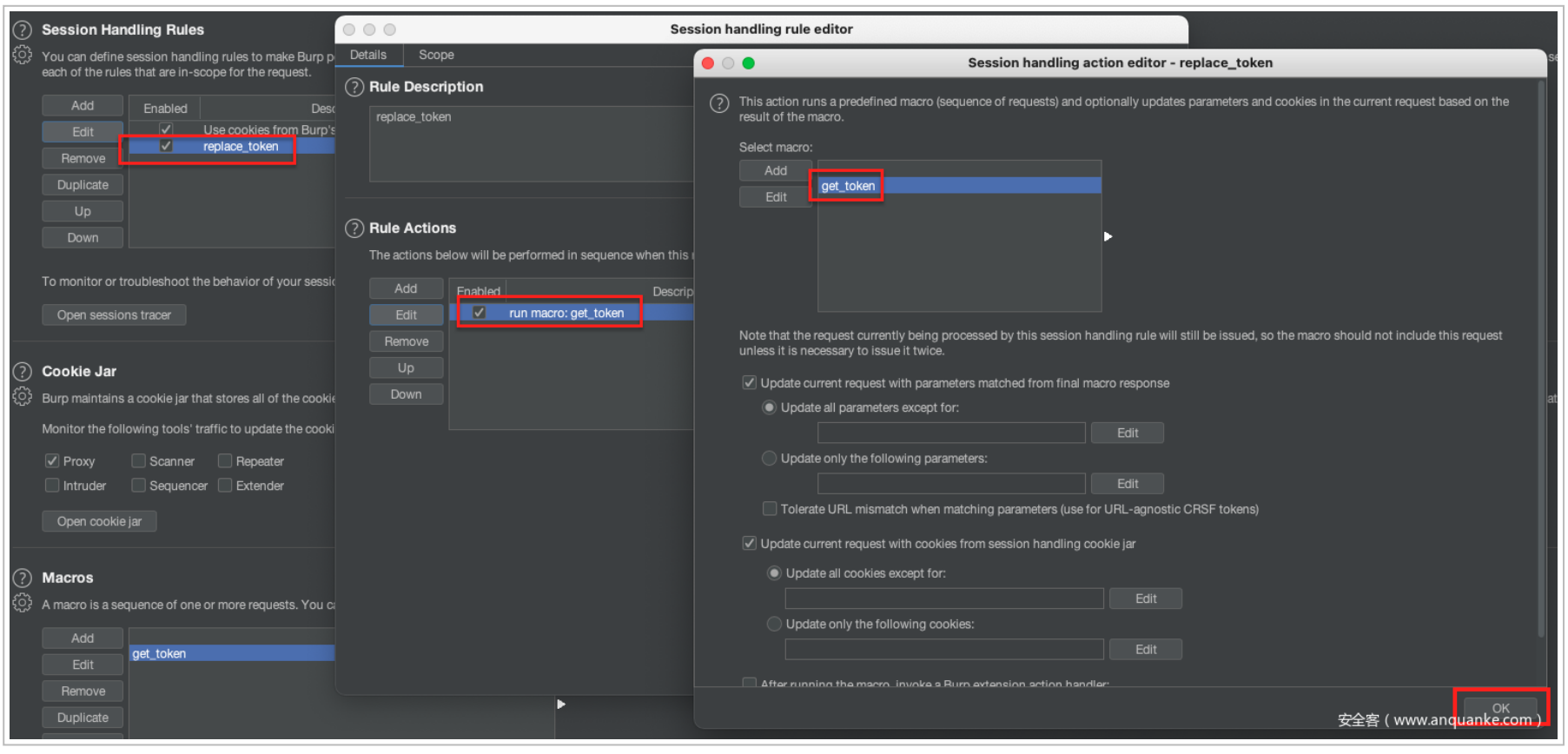


图 9

最后一步的设置是关于这个宏的生效范围，让该宏在 target，scanner，repeater，intruder… 生效，这里为方便我让其对所有的 URL 都生效，一切都设置完之后，点击 OK 即可

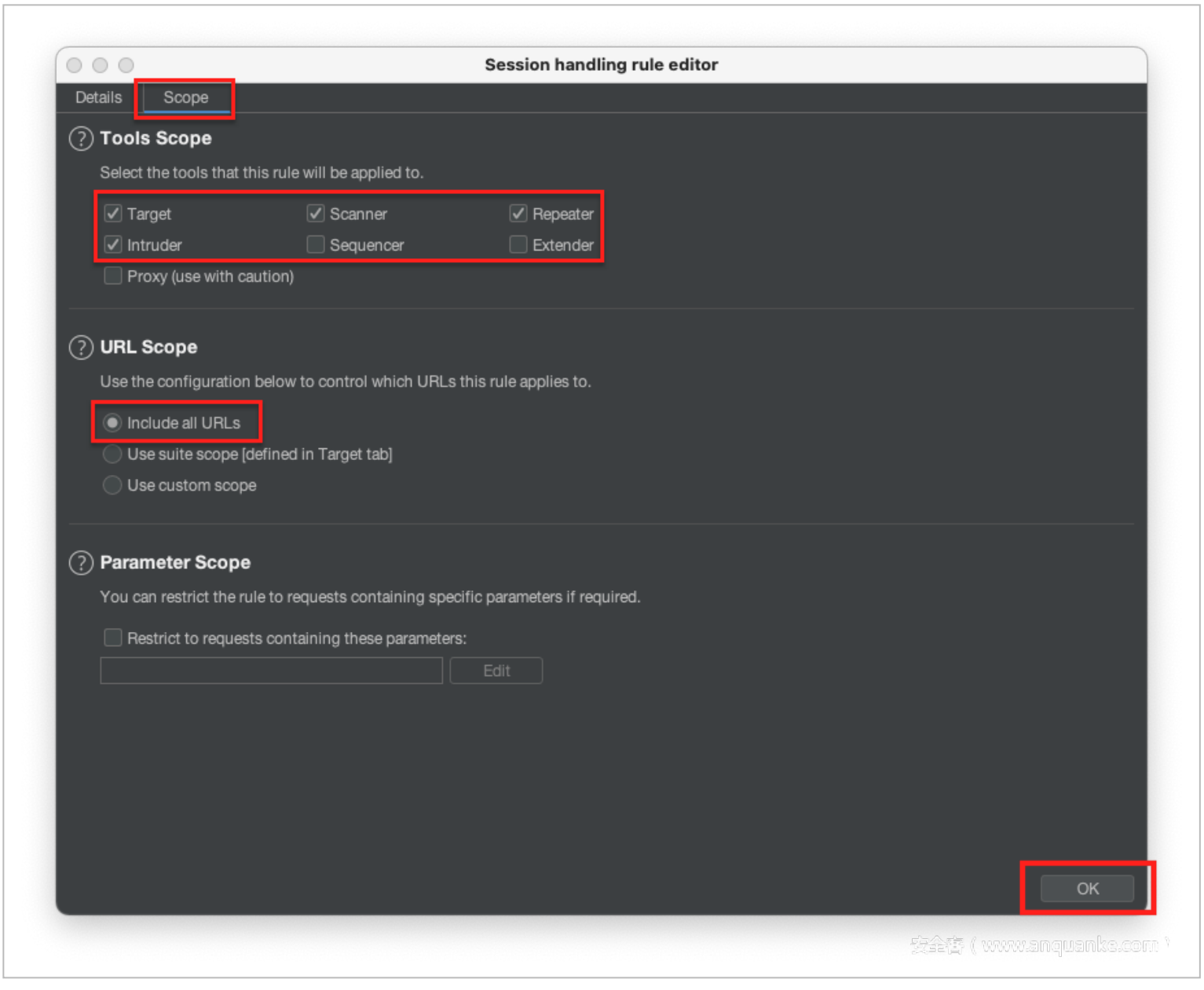


图 10

最后就来检验一下前面设置的宏是否生效，前面宏的生效范围包括了 repeater，那就拿 repeater 来做一个简单实验吧！下图中 token 我并没有手工进行替换，然而当我重放时 token 自动被宏给替换掉了，并且系统校验成功了！那说明我们的宏生效了！

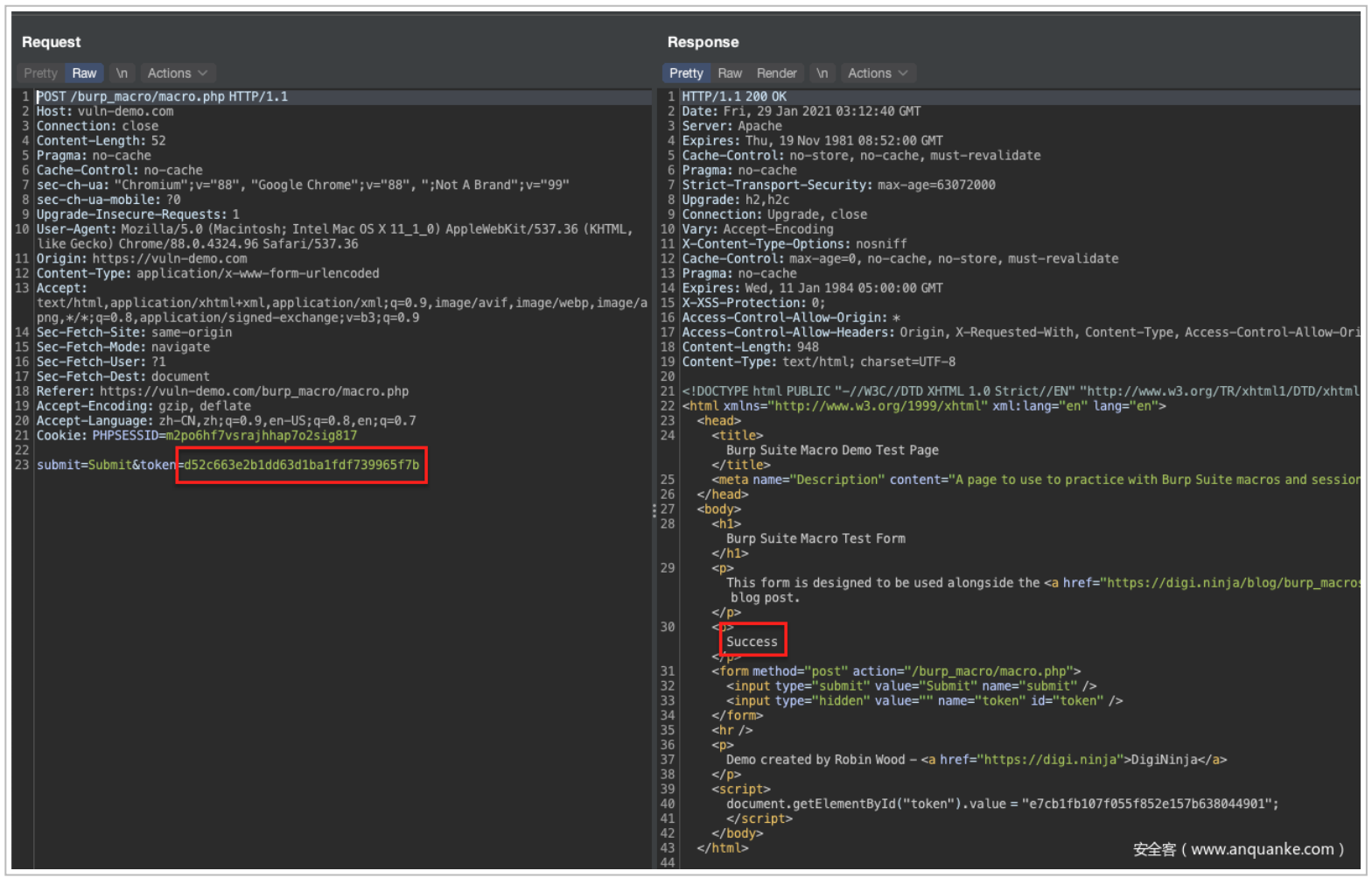


图 11

0x03 burpsuite 宏功能的局限性

经过上面对宏的描述，读者心里是不是有个感觉，哇，burpsuite 竟然还有这个功能，这个功能还挺强大的呀!

Burpsuite 的这个宏虽然强大，但是却解决不了我的问题，这是为什么呢? 还记得上面我们设置宏时设置的参数 token 吗? 为什么前面要强调设置参数的时候要一字不差? 因为 burpsuite 只能根据 token 这个字符串来进行动态替换，而且也只能动态替换两种格式的数据包，第一种：GET /burp_macro/test.php?token=122345677 HTTP/1.1。第二种 POST /burp_macro/macro.php HTTP/1.1 submit=Submit&token=d52c663e2b1dd63d1ba1fdf739965f7b。读者看出猫腻了吗? 也就是说 burpsuite 的宏只能处理 GET 和 POST 请求中参数是 token=”” 这种格式的参数。对于其他的场景他是无能为力的。而我遇到的场景是 json 格式的数据包。对于这种格式的数据包 burpsuite 是无能为力的!

0x04 引入一个新的神器 burp-cph

既然我会遇到这种问题，别人肯定也会遇到同样的问题，我在 burpsuite 的官方论坛上找了一下，果然还是有不少人遇到同样的问题，而且官方推荐了 burp-cph 这个插件 https://portswigger.net/bappstore/a0c0cd68ab7c4928b3bf0a9ad48ec8c7，那接下来我就演示一下他是否能解决我的问题?

第一步安装插件

插件安装成功之后的界面如下所示

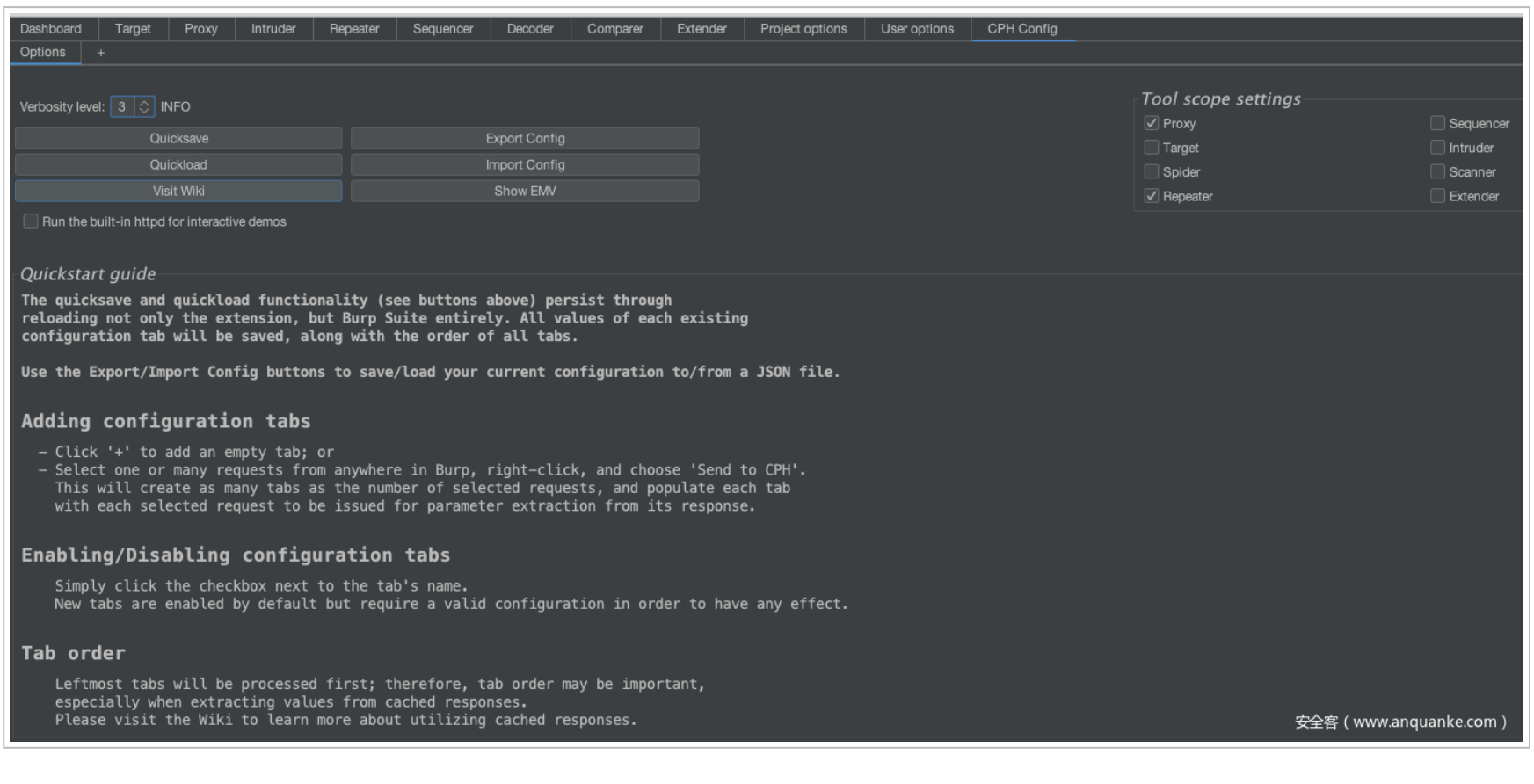
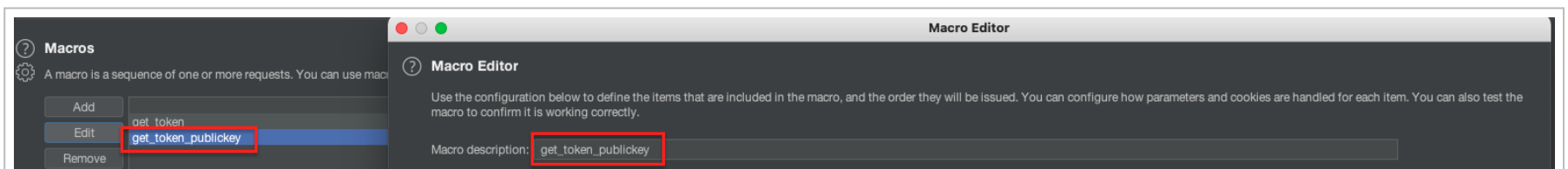


图 12

burpsuite 的宏本身设置有供第三方插件调用的接口，所以我们可以让 burp-cph 和 burpsuite 进行联动。大概的玩法就是我们在发起 loginWithPassword 请求之前让 burpsuite 宏动态请求 getToken 获取动态变化的参数，然后使用 burp-cph 将提取的参数自动替换到 loginWithPassword 请求中，之后再请求 loginWithPassword，这样就保证了爆破时动态变化的参数每次随着请求的不同而不同。

第二步，设置 burpsuite 宏并与插件进行联动

新增一个 get_token_publickey 宏，选择 / api/getToken 请求，之后不再需要配置任何参数，直接点击 OK 即可



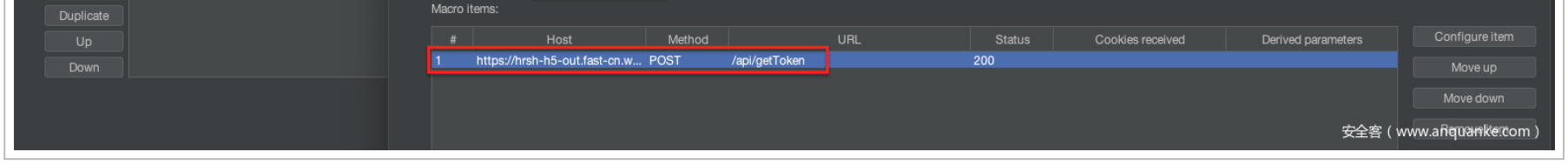


图 13

新增一个调用 get_token_publickey 的规则 replace_token_publicKey，并且在运行完该宏之后将结果直接抛给插件 CPH 让其进行处理

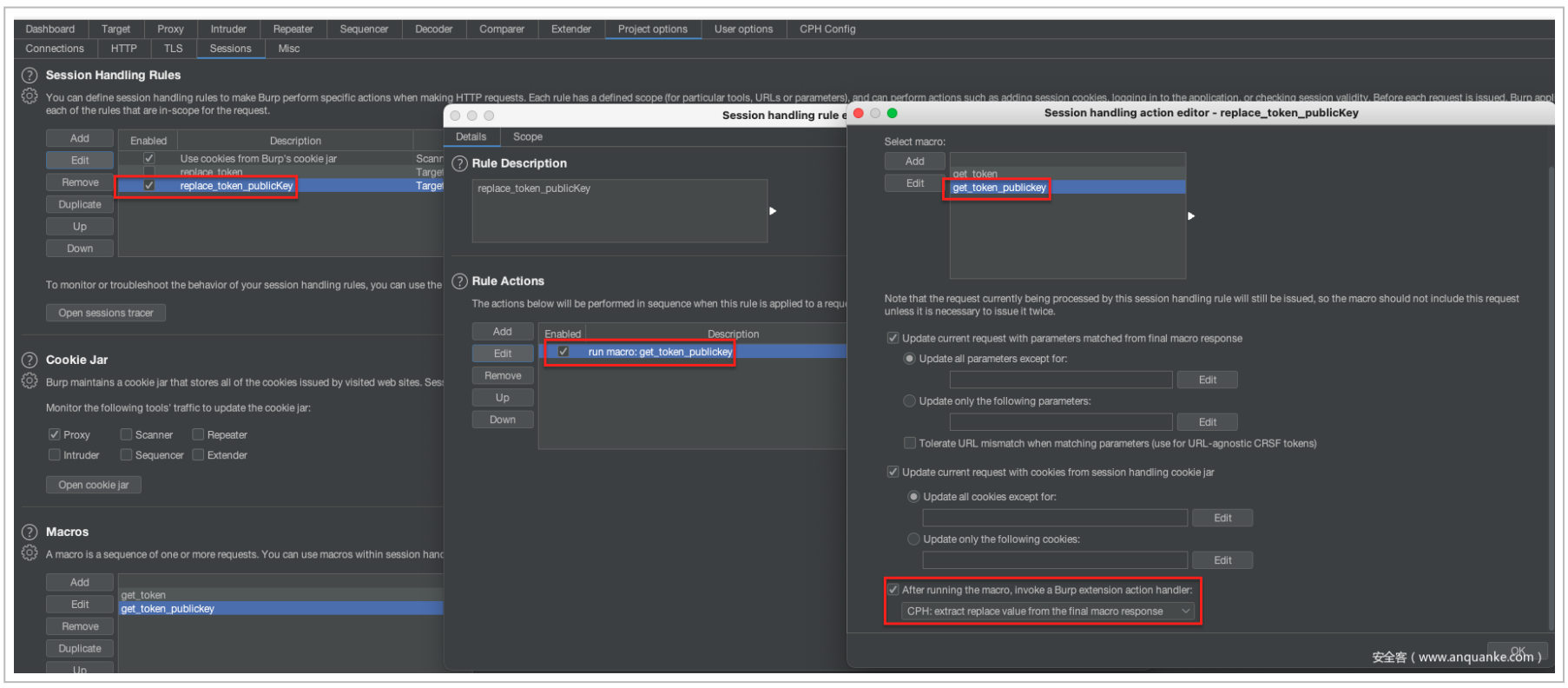


图 14

第三步，配置插件 CPH

在配置 CPH 的第一步是将需要目标地址写进 scope 里面，这样 CPH 才能对我们测试的地址生效。

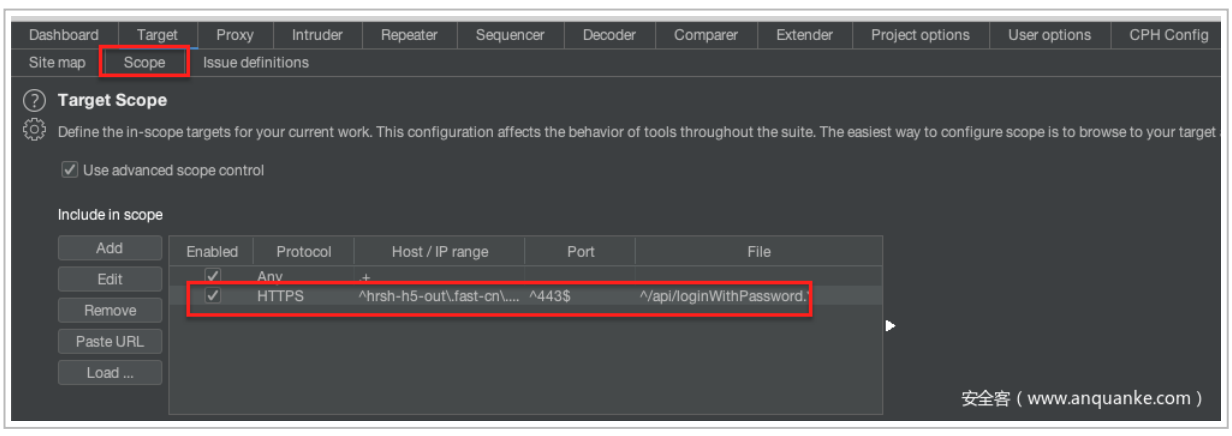


图 15

接下来我们的思路是让插件 CPH 从 burp 的宏的结果中动态提取 token 和 publicKey 变量，然后重新利用此参数进行请求。

设置 burp 的宏

设置 burp 的宏，并抓取特定的 getToken 的请求记录即可，其他不需做任何操作

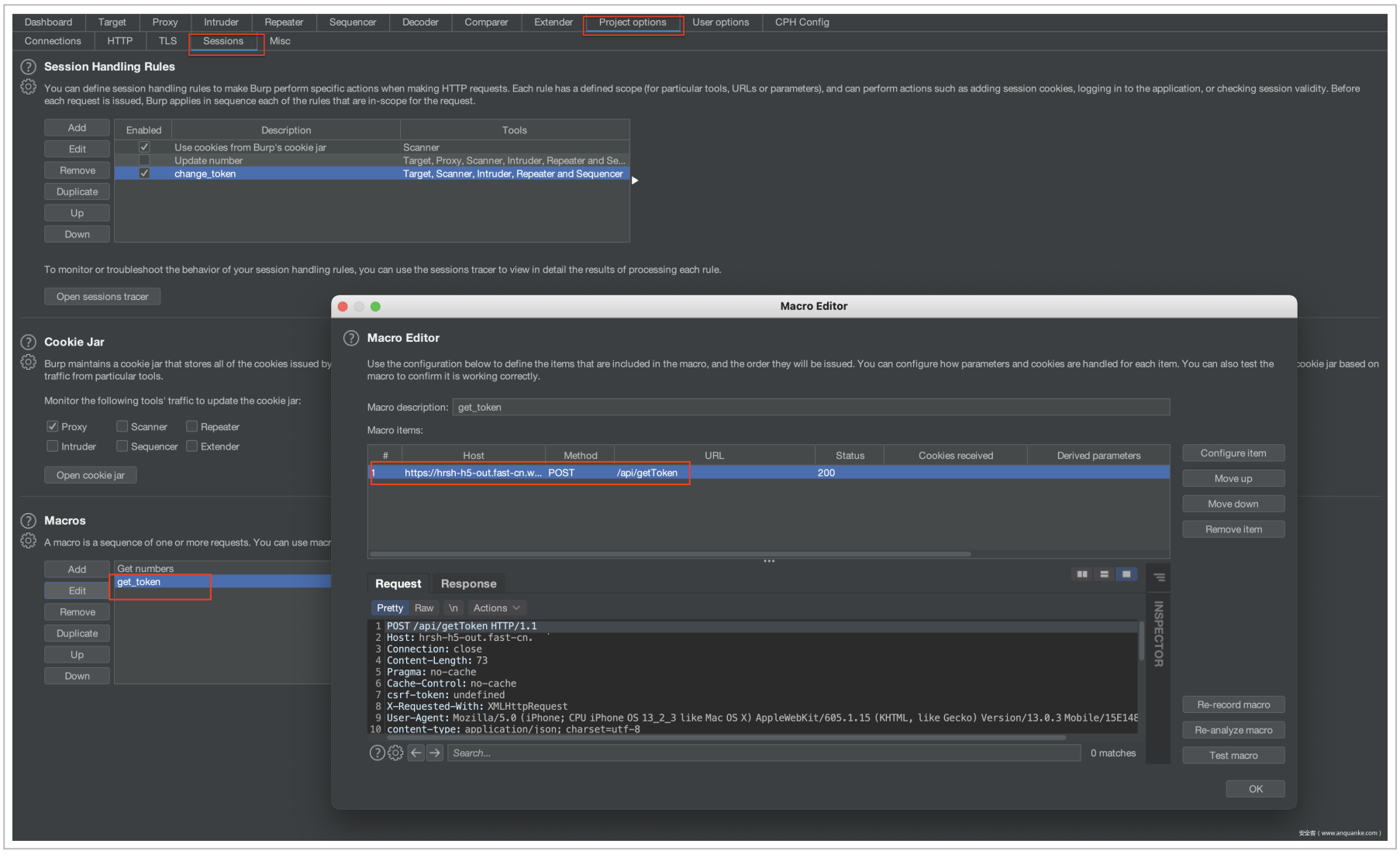


图 16

设置一条调用我们刚才设置宏的规则 change_token，只需要将结果转到 CPH 插件即可，然后就不需要做任何操作了

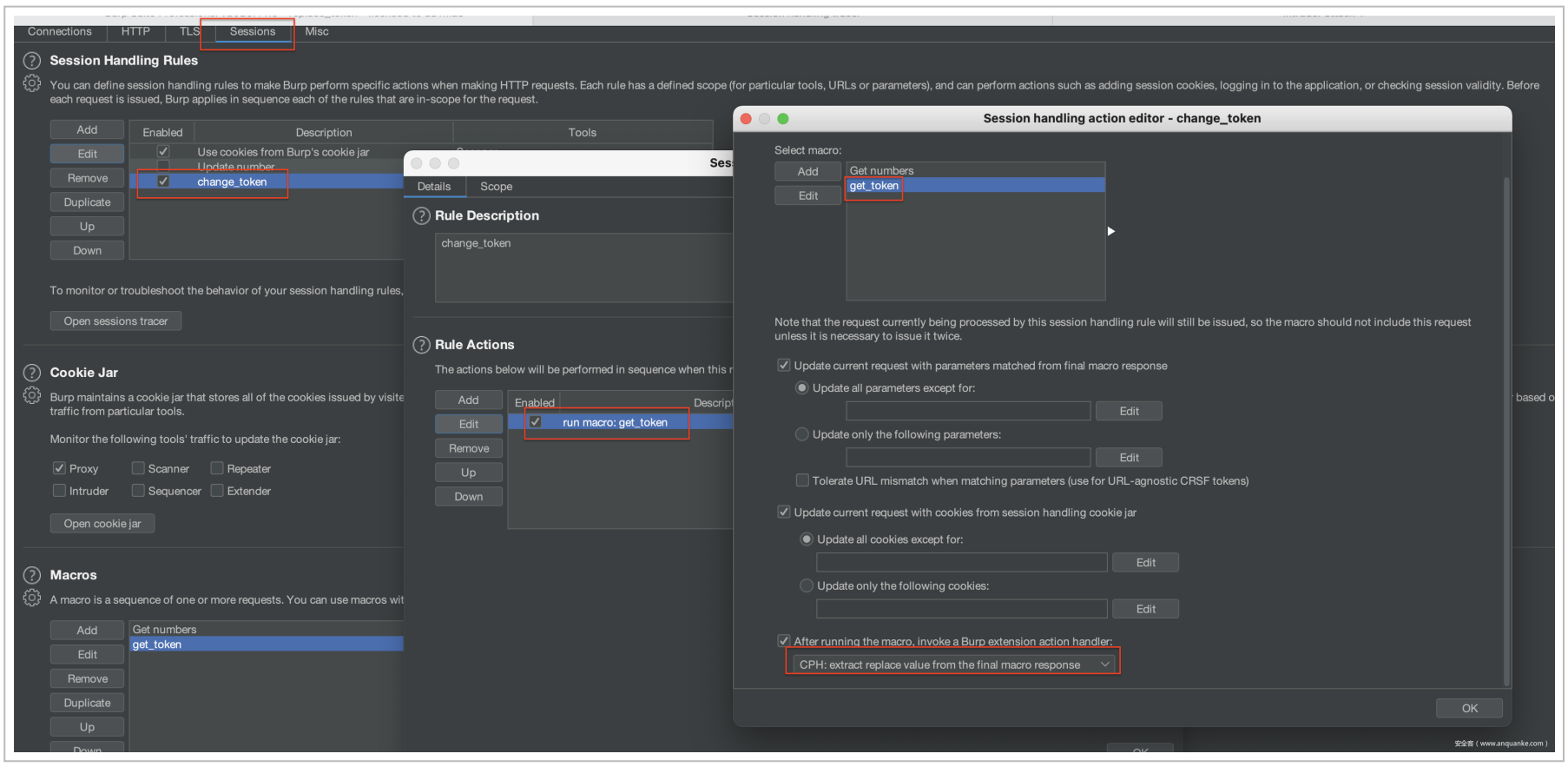


图 17

设置 CPH 与 burp 的宏进行联动

图 18 是我已经设置完成一首是能够动态替换 token 的配置界面，刚开始大家看的时候可能有一点懵，下面我给读者简单解读

图 18 是我已经设置完成，并且能够动态替换 token 的配置界面，刚开始大家看的时候可能有一点懵，下面我给读者简单解读一下。

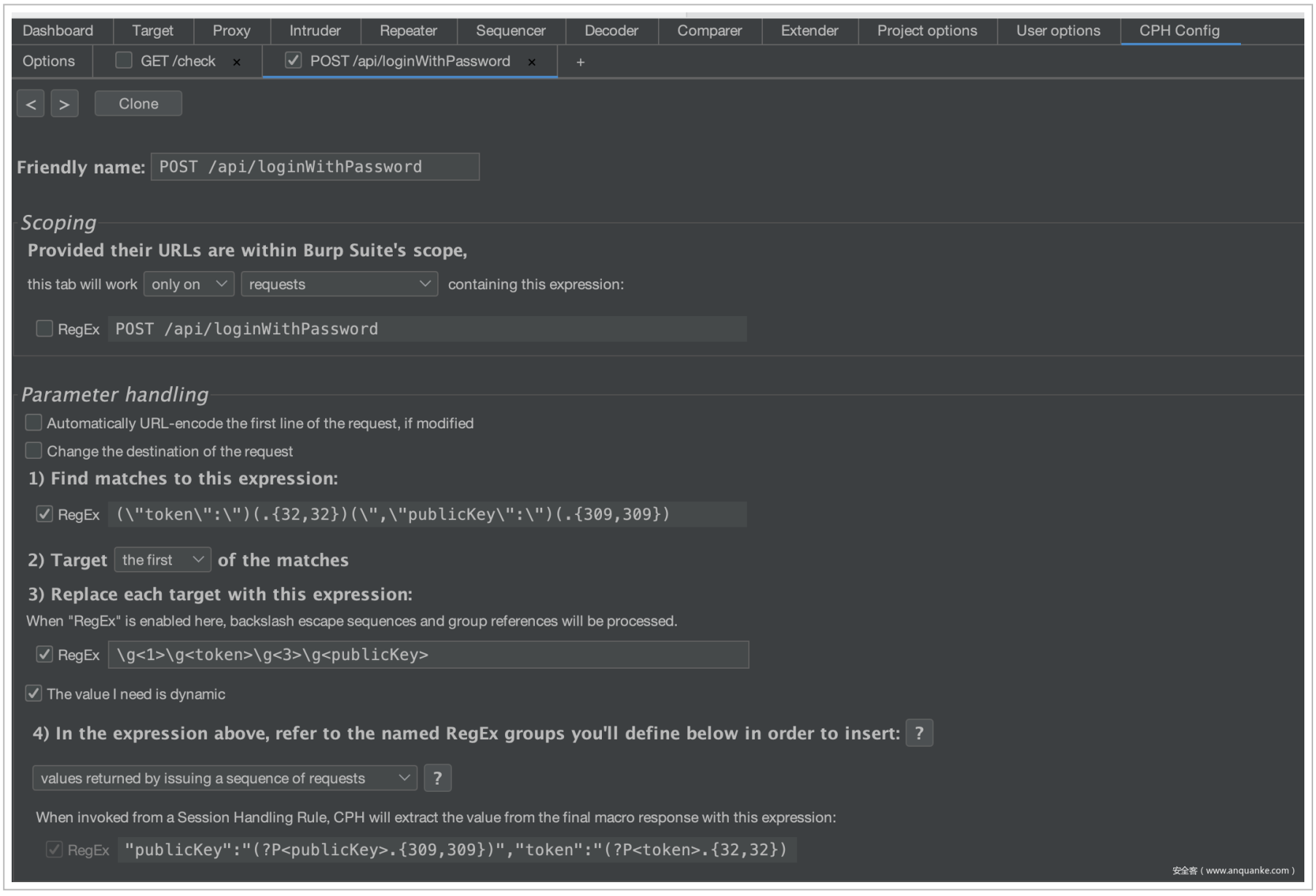


图 18

Scoping 这里的配置作用是对特定的请求生效，比如图 18 中只要数据包中含有 POST /api/loginWithPassword，则规则就对该条数据包起作用。

“1)”的作用则是匹配动态变化的参数，比如实际的效果就是匹配从”token”到 publicKey 的全部内容。

正则 (\”token\”:\”)(.{32,32})(\”,\”publicKey\”:\”)(.{309,309}) 将匹配到的内容分成了 4 部分: 第一部分 (\”token\”:\”) 匹配的内容是”token”:，第二部分 (．{32，32}) 代表是的 token 的内容，token 的内容固定是 32 位，第三和第四部和前面类似。

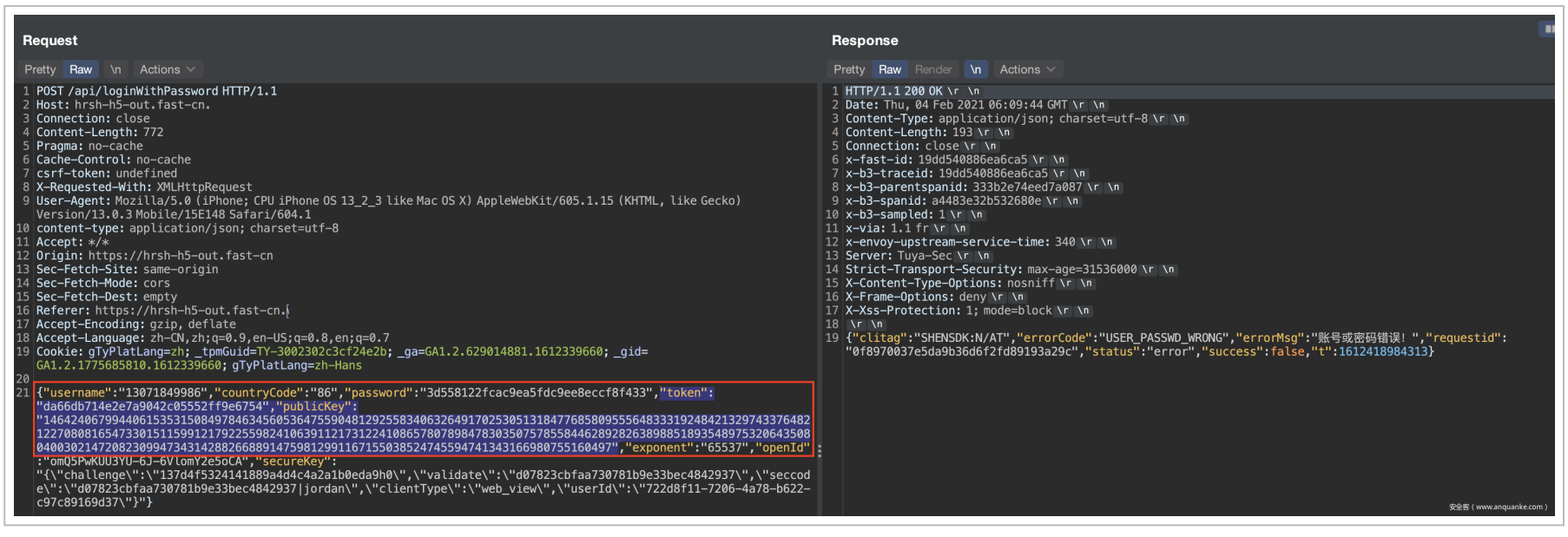


图 19

“4)”的作用是从前面设置的 burpsuite 宏的结果中匹配动态变化的 token 和 publicKey 的内容 (?P<test>.{1,1}) 是一种特定的正则格式，举个简单的例子我们可以用”token”:”(?P<token>.{32,32})从宏的响应结果中匹配到内容 da66db714e2e7a9042c05552ff9e6754 如下图 20 所示

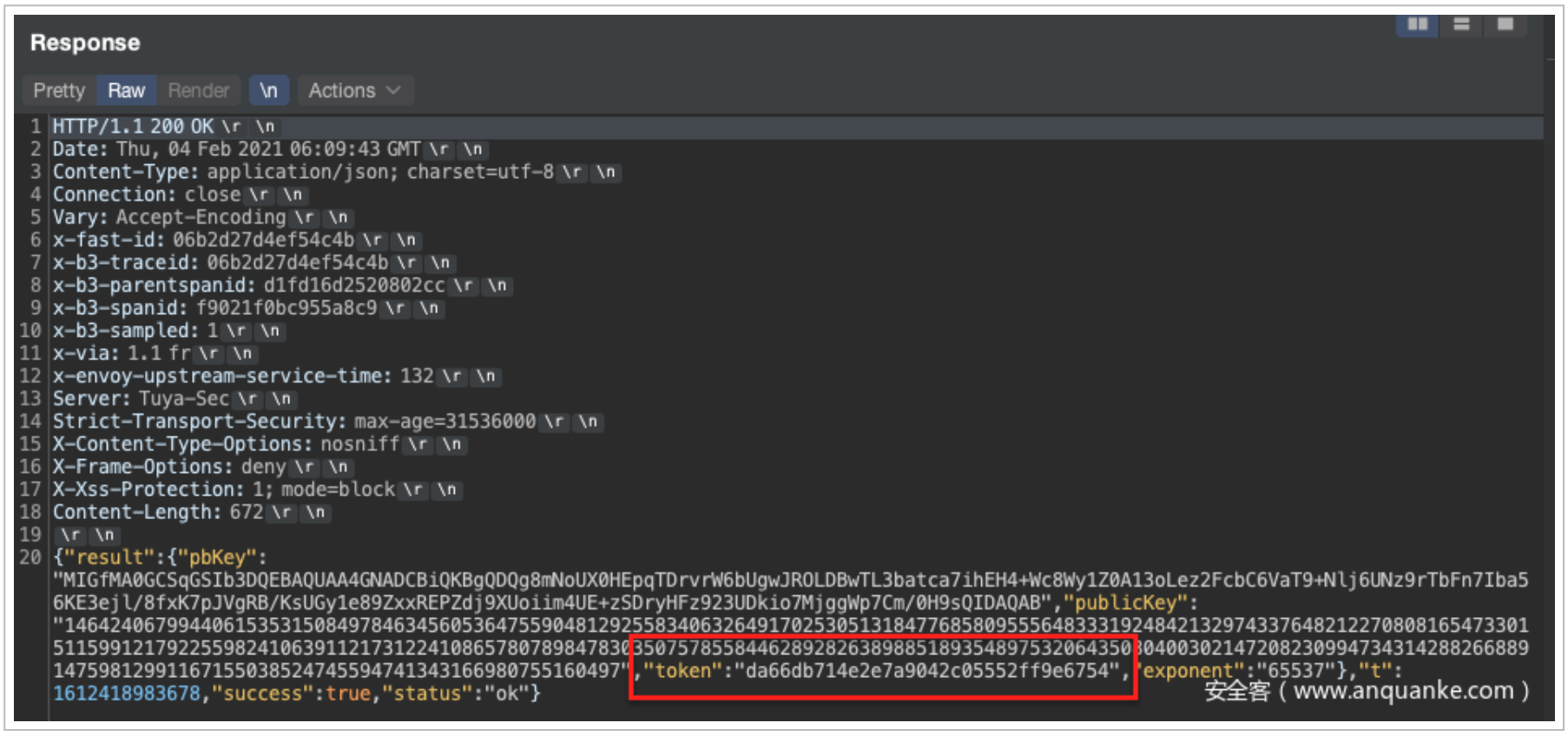


图 20

“3)”的作用则是将”4)”中提取的 token 和 publicKey 重新替换到请求中，然后再次发送请求。\g<1 > 是一种特定的格式，代表的内容是”1)”第一个 () 里面匹配的内容 \ g<token > 则代表的是”4)”中匹配到的 token 内容，至此我们再回头看一下图 18 此时是不是已经一目了然了。

0x05 最终的效果

CPH 不会将替换值的结果直接展示在我们的面前，但是实际请求时已经动态将值进行了替换，这句话可能有点绕，那我们就截图进行说明一下，如图 21 所示，我直接对 loginWithPassword 进行了爆破，虽然，我们看到 intruder 界面 token 值没有发生变化，但是实际在进行爆破时 CPH 已经将值进行了替换，我们使用变量追踪器看看一下 macros 和 cph 动态对参数的处理如图 22 所示，每一个 token 和 publicKey 都不一样，至此我想要的目的达到了~

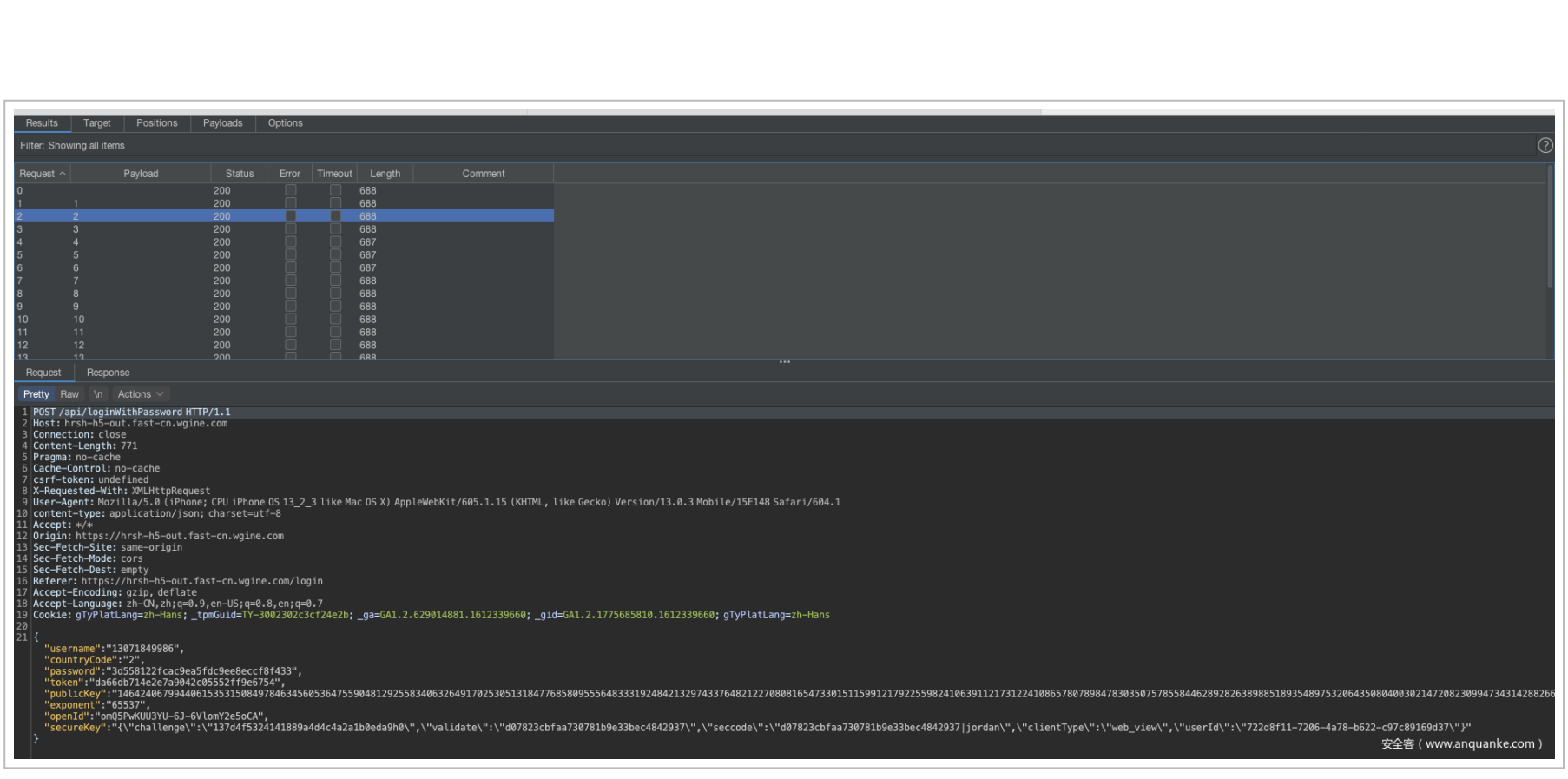
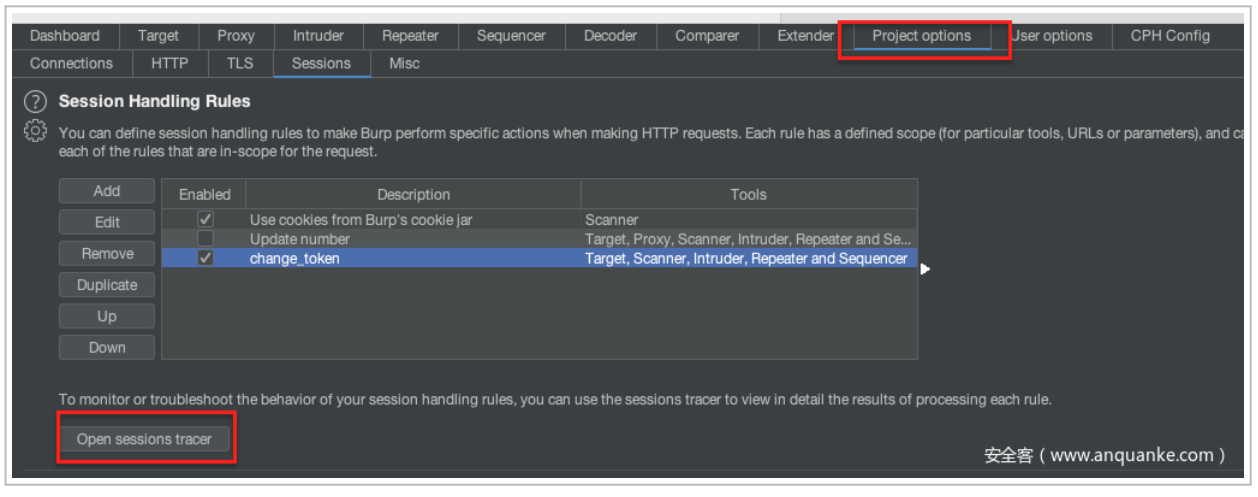


图 21



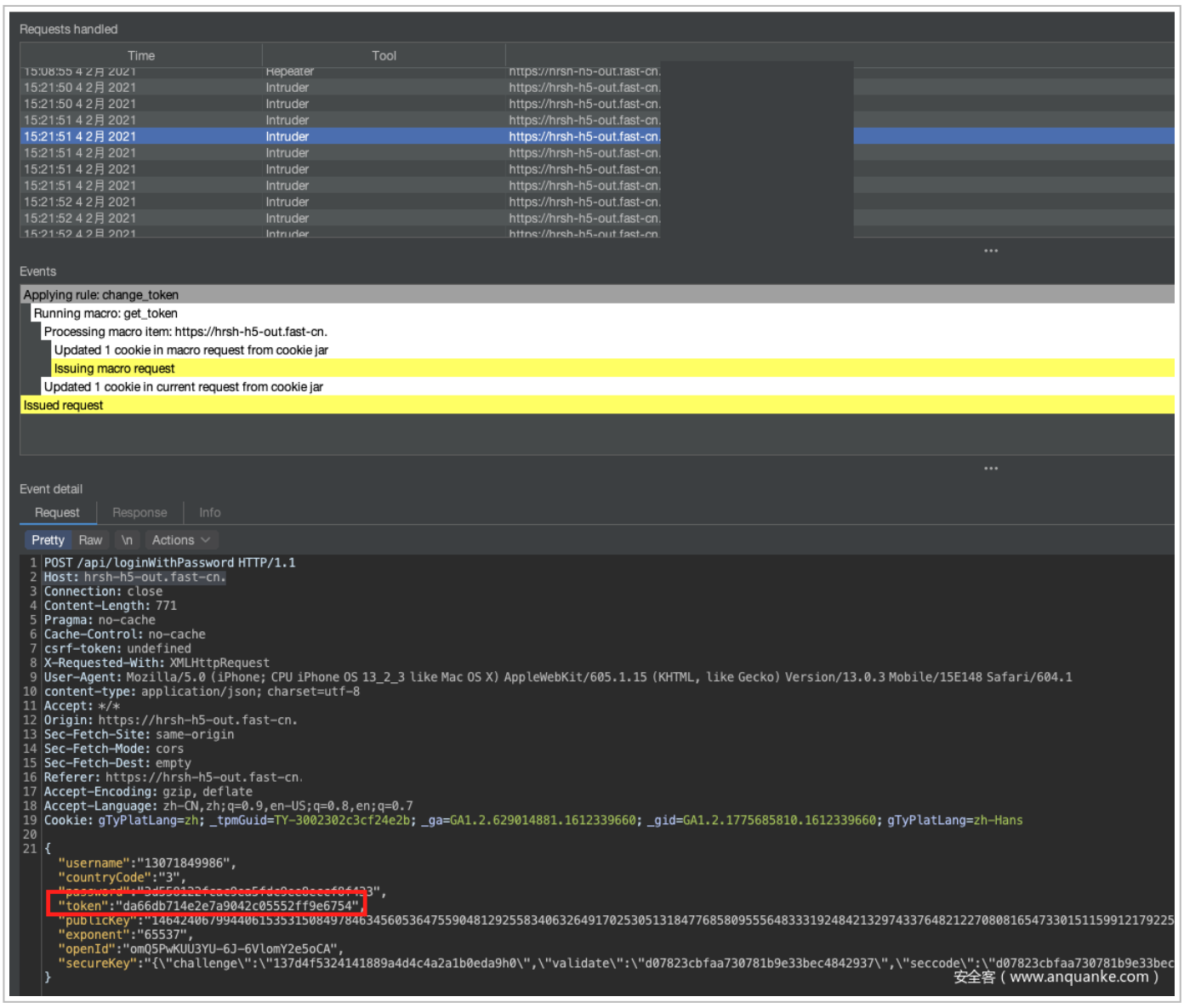


图 22

0x06 更深一步的思考

根据上面描述，应该可以解决 80% 以上的应用场景，但还有一种应用场景没有被覆盖掉，一个请求中的参数来自于多个请求，而且多个请求相互独立，这种场景又该怎么处理呢？又该怎么动态提取呢？聪明的读者可以想一想，其实 CPH 也是可以做的！感兴趣的可以自己动手试一下！当然可以加我微信: bmV0d29yay1zZWN1cml0eQ== 交流哈~

本文作者水平很有限，不足之处请多多包涵，大佬勿喷哈~