Burp Suite四种模式详解：

四种模式分别为sniper（狙击手）、Battering ram（破城槌）、Pitchfork（木叉、杈）、Clusterbomb（集束炸弹）

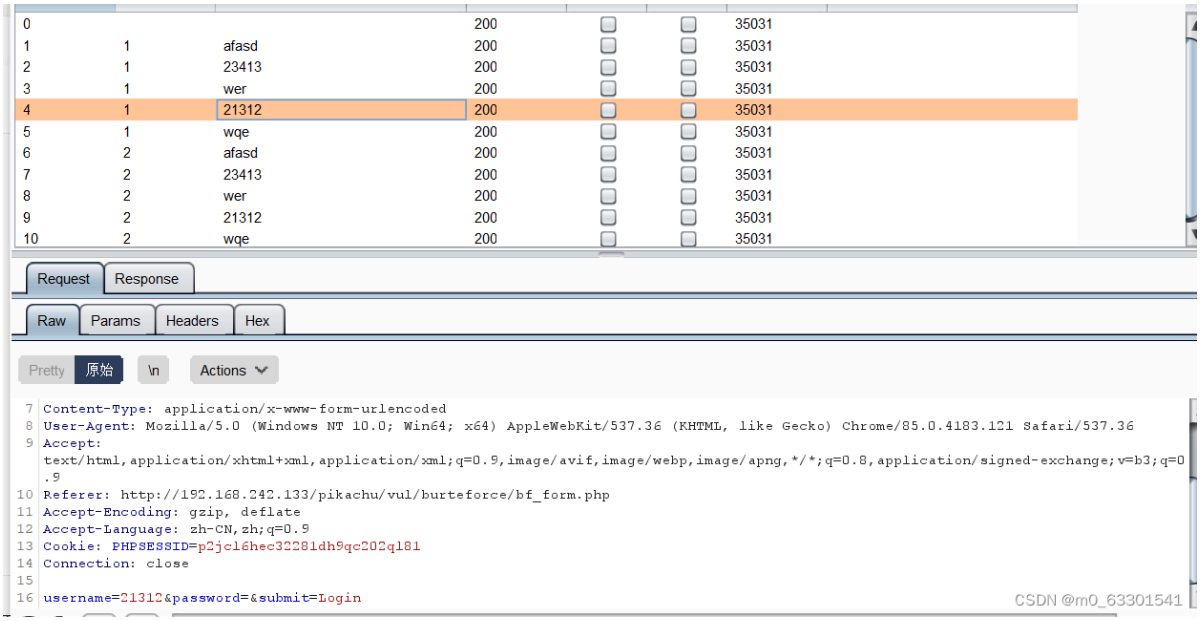
其中可以按照字典数量大致分为多字典类型攻击和单字典类型攻击

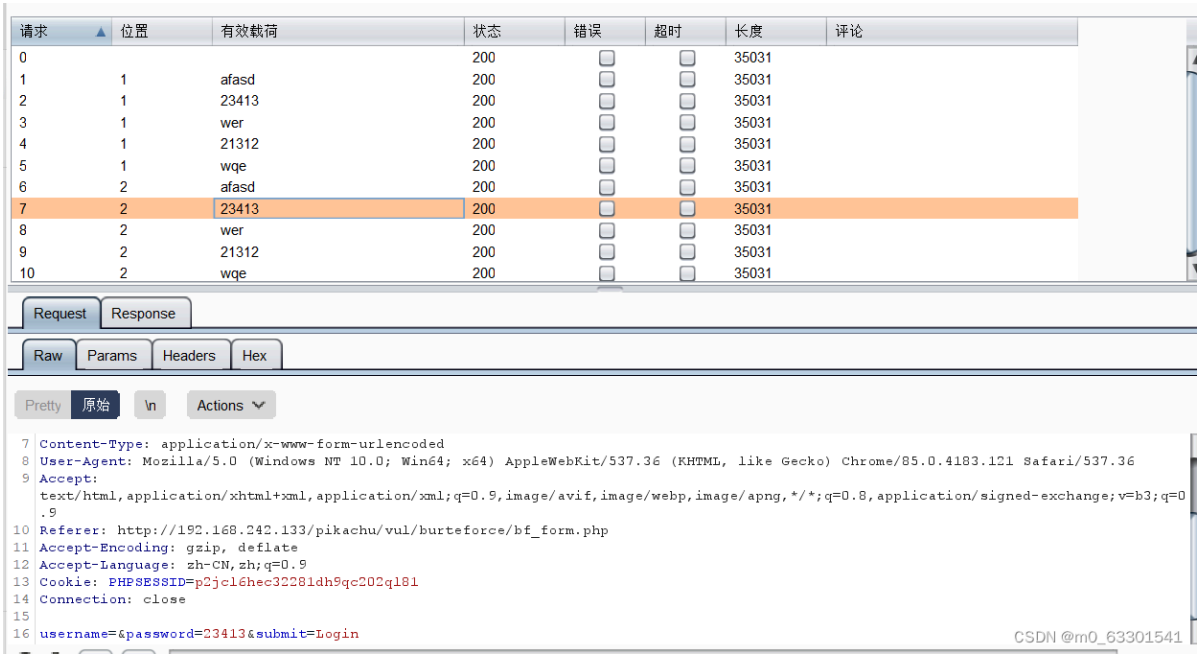
> 单字典(只有一个字典)
> 1.Sniper：按顺序一个一个参数依次遍历，一个参数遍历完，然后恢复成原数据，再遍历下一个参数。
> 2.Battering ram：每个参数同时遍历同一个字典。
> 一个相当于单挑，一个相当于群殴。

> 多字典(有多少参数就有多少字典）
> 1.Pitchfork：多个参数同时进行遍历，只是一个选字典1，一个选字典2（相当于50m赛跑同时出发，只是赛道不同）
> 2.Cluster bomb：有点像两个嵌套的for循环，参数i和参数j，i=0，然后j要从0-10全部跑完，然后i=1，然后j再从0-10跑完，一对多，多次遍历

1、sniper（狙击手）

sniper（狙击手）攻击方式只能同时对一个参数进行字典遍历，如图所示：

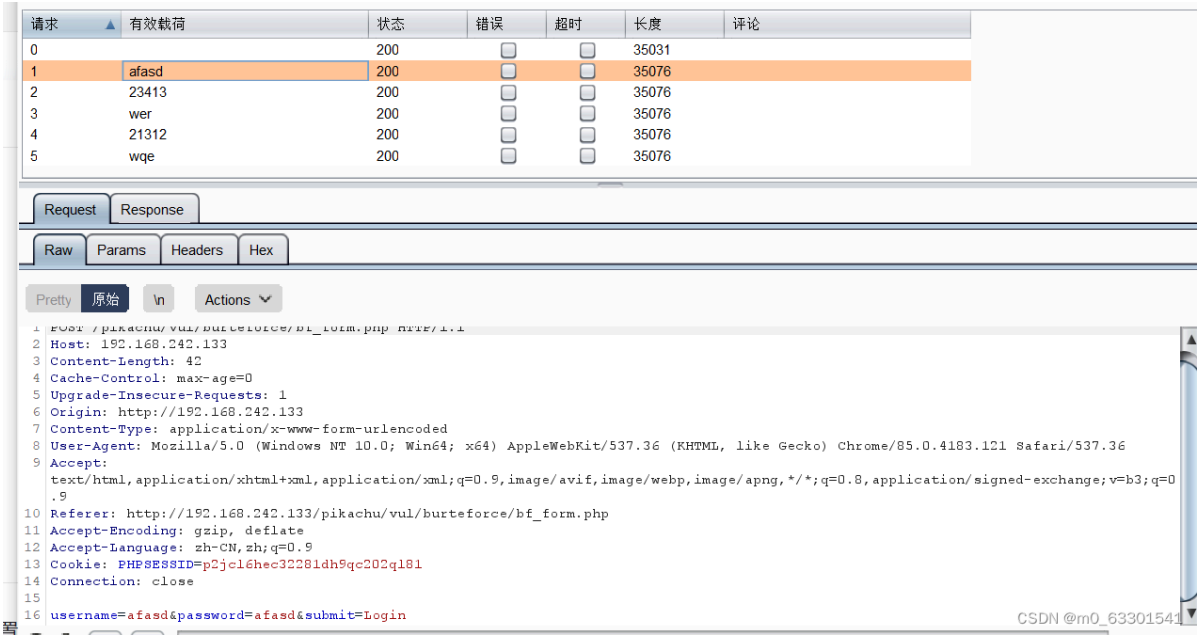




爆破后得到的结果同一时间内只有一个参数参与遍历，也就是说，这种爆破方式是没有办法进行账号和密码的同时爆破的，只能进行对单一参数的爆破。

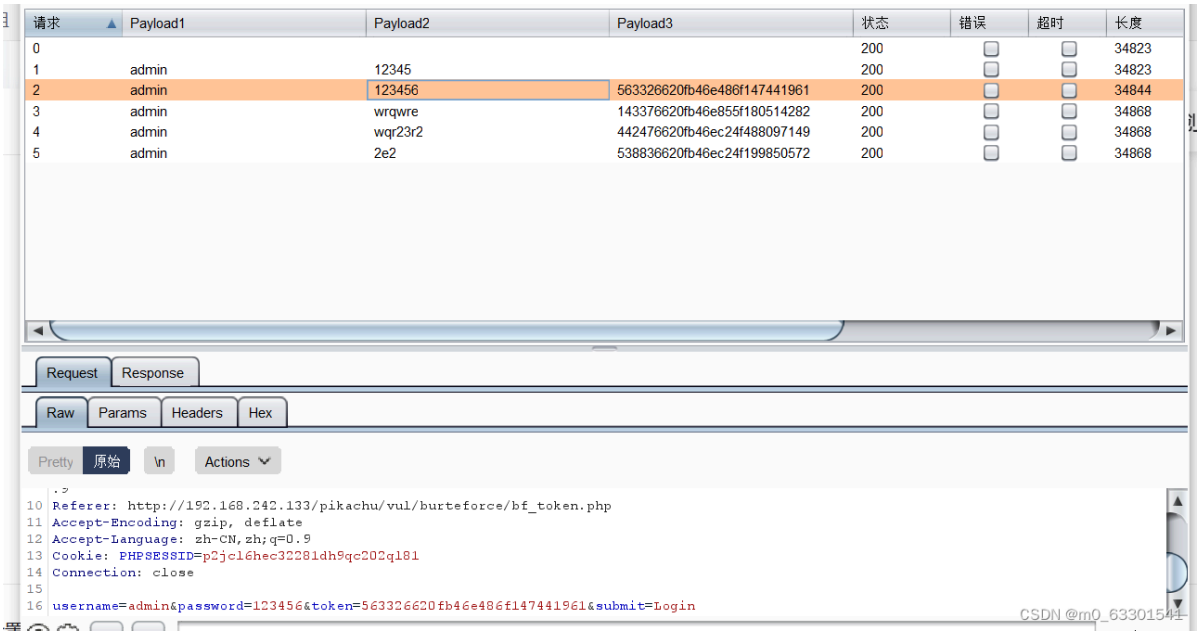
2、Battering ram（破城槌）

Battering ram(破城槌爆破)攻击方式是能够同时对多个参数基于一个字典进行遍历，如图所示：



我们可以看到，这种爆破方式适用于知道账号爆破密码的情况，如果在账号密码都不知道的情况下，一对一爆破，则意义不大，因为大部分爆破爆破都只是会回显username or password error。

除此之外，该攻击模式还有一个优点，在面对类似pikachu爆破板块的“token防爆破”部分的时候，因为该攻击模式一对一的特点，三个参数同时进行遍历，那么迭代的token数据也能实时更新。

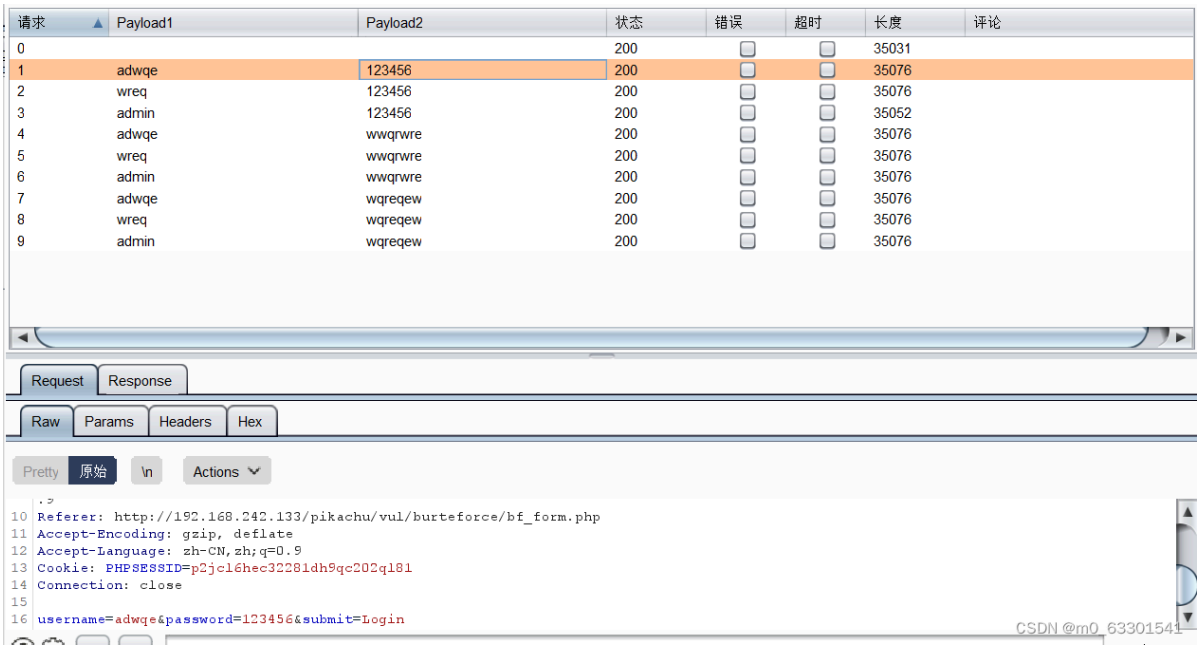


而集束炸弹在面对token这种每次刷新的放爆破手段会面对一些问题。

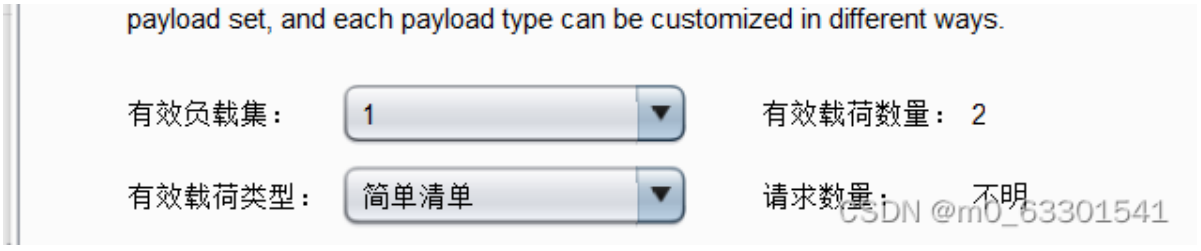
ps：写到这里我突然意识到一个问题，木叉（Pitchfork）攻击模式可能就适用于有类似token这种防爆破手段的场景，针对某一个用户爆破密码的情况。完全可以把某个用户设置为定量，而其密码和类token值为变量进行爆破。

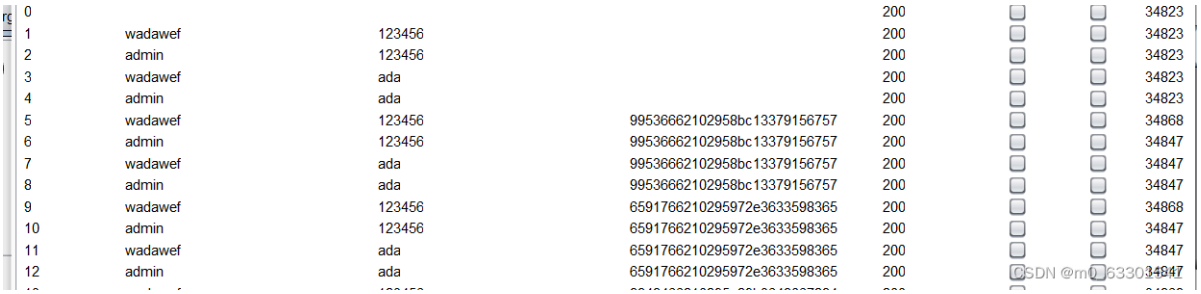
4、Clusterbomb（集束炸弹）

集束炸弹（Clusterbomb）这种爆破模式主打一个不放过，全部遍历，可以说是最精确的爆破方式，理论上来说，只要算力充足，字典集够庞大，就能爆破成功。如图所示



但是，这种爆破方式有一个弊端，那就是在面对类似“token防爆破”这种情况的时候，就会有很大的问题，因为这种情况有一个每次请求都更新的值，会造成永远不会停止的爆破。如图所示：





0				200			34823
1	wadawef	123456		200			34823
2	admin	123456		200			34823
3	wadawef	ada		200			34823
4	admin	ada		200			34823
5	wadawef	123456	99536662102958bc13379156757	200			34868
6	admin	123456	99536662102958bc13379156757	200			34847
7	wadawef	ada	99536662102958bc13379156757	200			34847
8	admin	ada	99536662102958bc13379156757	200			34847
9	wadawef	123456	6591766210295972e3633598365	200			34868
10	admin	123456	6591766210295972e3633598365	200			34847
11	wadawef	ada	6591766210295972e3633598365	200			34847
12	admin	ada	6591766210295972e3633598365	200			34847

在这种情况下，请求数量不明，且爆破结果不确定。

总结：
sniper（狙击手）:适用于只有一个参数的爆破。

Battering ram（破城槌）：属实鸡肋，实在想不到有啥应用场景，求大佬指点。

Pitchfork（木叉、杈）：适用于知道账号，不知道密码，且有一个在前端随时刷新的数据的爆破场景（如pikachu的“token防爆破”）。

Clusterbomb（集束炸弹）：适用于大部分的爆破场景，除了有一个在前端随时刷新的数据的场景。（也就是木叉应用场景）。