**前言**

1、漏洞挖掘一定要细，不要放过每一个角落。

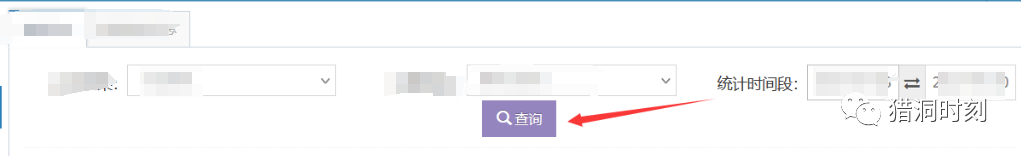
2、还有的就是要学会重复挖掘老站，因为每隔一段时间，你的实力和你的思想都是不一样的，回到老站说不定就能挖到新的漏洞。

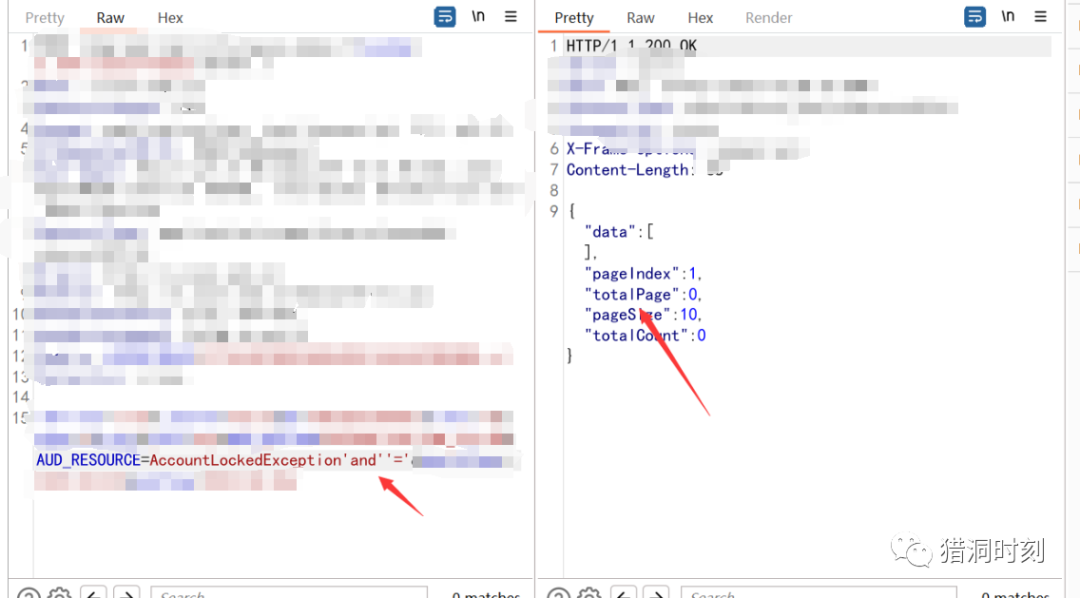


**SQL注入一**

这次的注入在一个"日志查询"的功能，在进行查询的时候，出现了注入。并且我可以告诉大家，在这种功能点很容易出现注入。

在AUD\_RESOURCE这个参数后面，添加一个**单引号**，直接返回的是一个空白页面，什么数据也没有，但是在后面添加 ' and ''=' 后就成功闭合，将数据进行完整的返回，因此证明存在注入。

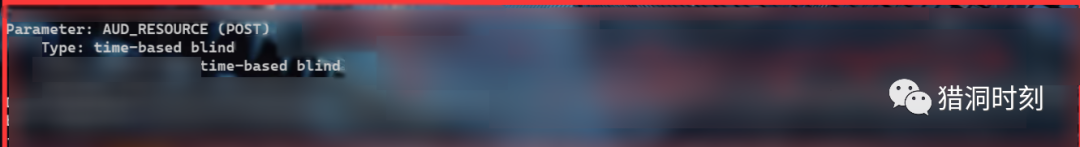




证明存在注入之后，还需要SQLMAP进行验证一下，否则edusrc不收取，或者使用sleep函数进行延迟也可以。

python .\sqlmap.py -r post.txt --batch --dbms 数据库类型 --tamper "space2comment.py" -p AUD\_RESOURCE

可以看到这是一个时间盲注。



提交后给了四个积分。

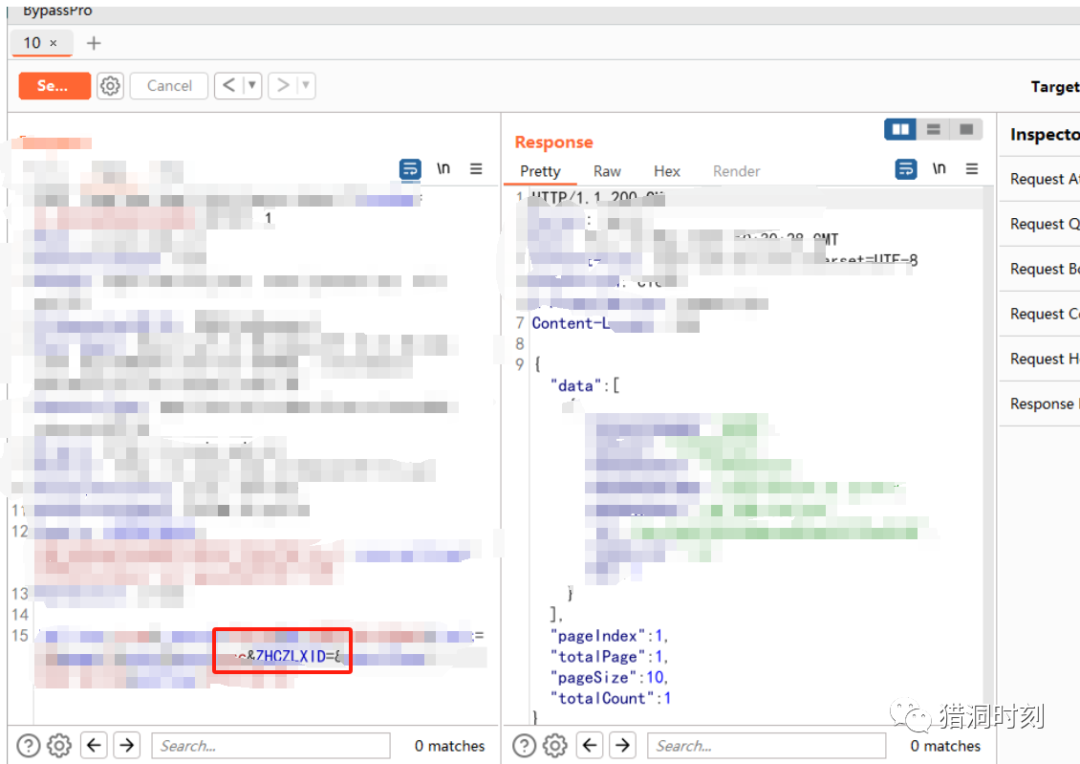


原本提交这个漏洞后，再摸索一遍，什么也没有然后就走了。但是过了一星期又重新摸回这个站点，于是就有了下面的SQL注入八连杀。

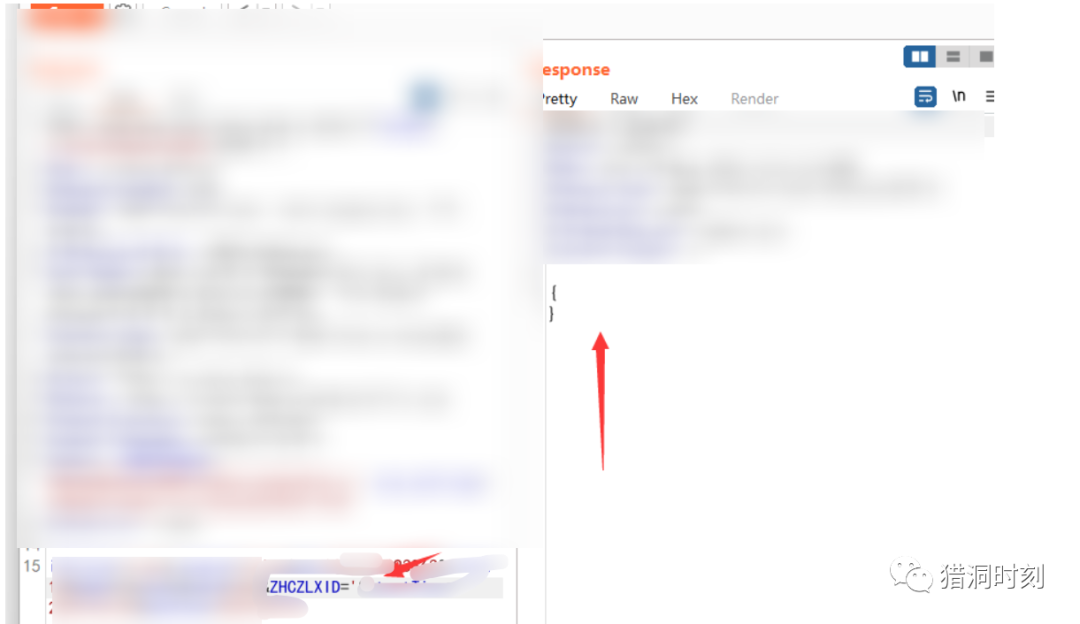
**SQL注入二**

这次的注入点在"账号管理"，点击账号管理的时候进行抓包。

正常情况下，ZHCZLXID参数为空，但是能够成功返回出数据。

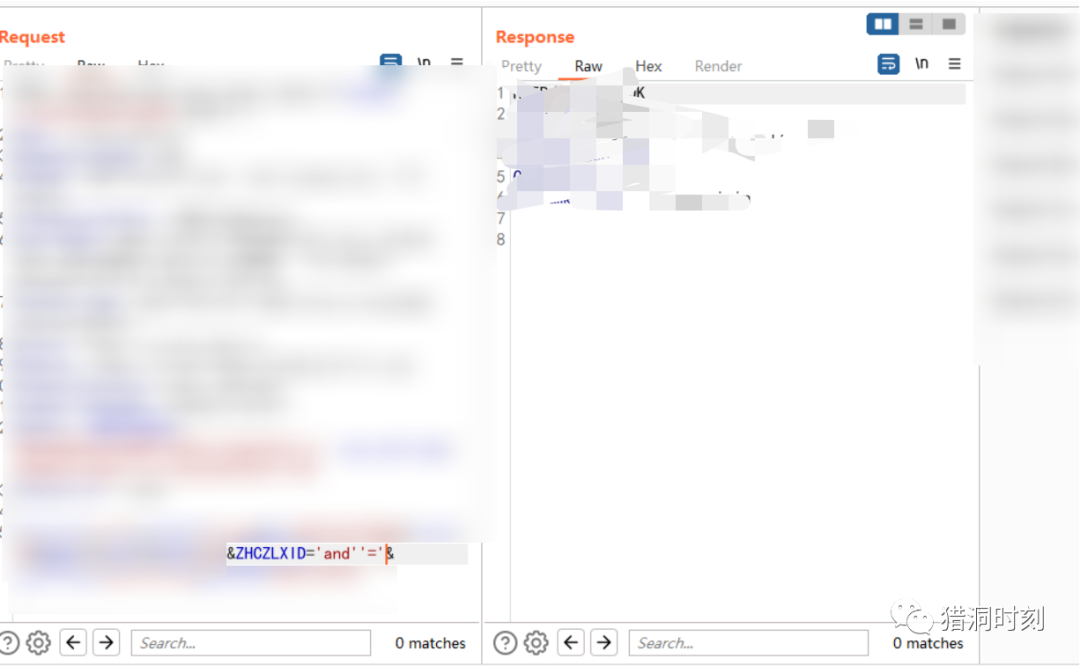


然后还是习惯性的添加一个单引号，发现只返回了一个{}，情况有变！



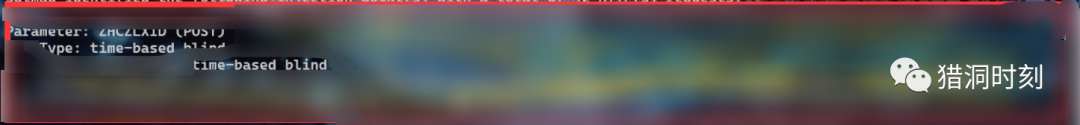
然后继续尝试进行闭合，使用and ''='进行闭合。

发现返回值直接没有了，但是返回值没有，可不代表不存在注入！

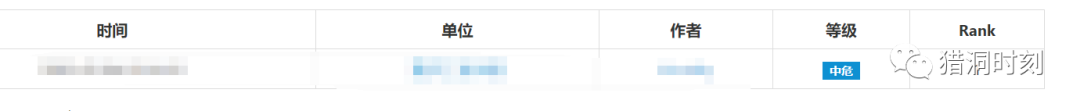


python .\sqlmap.py -r post1.txt -p ZHCZLXID --tamper=space2comment --random-agent --dbms 数据库类型

可以看到成功跑出来注入漏洞，是个时间盲注。

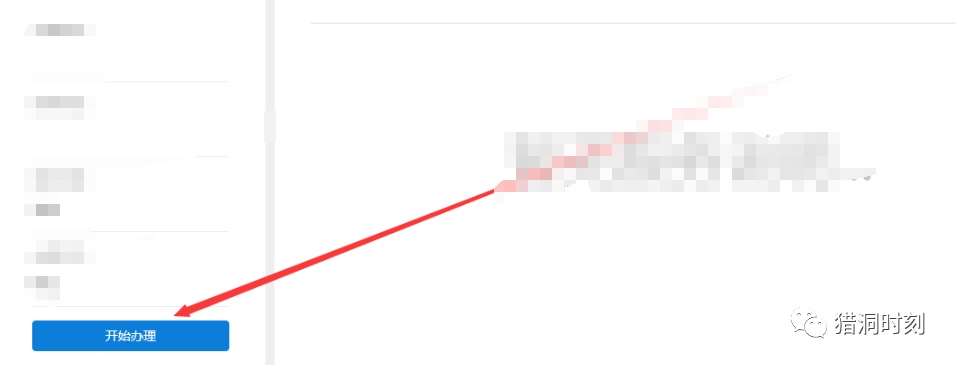


因为是同个系统，并且多次提交同类型漏洞，因此这次只给一个积分。

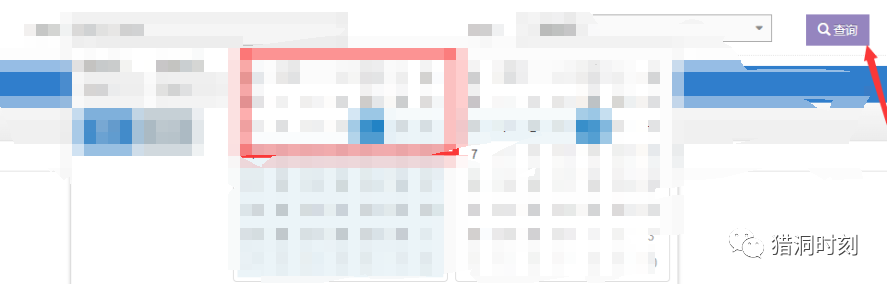


**SQL注入三**

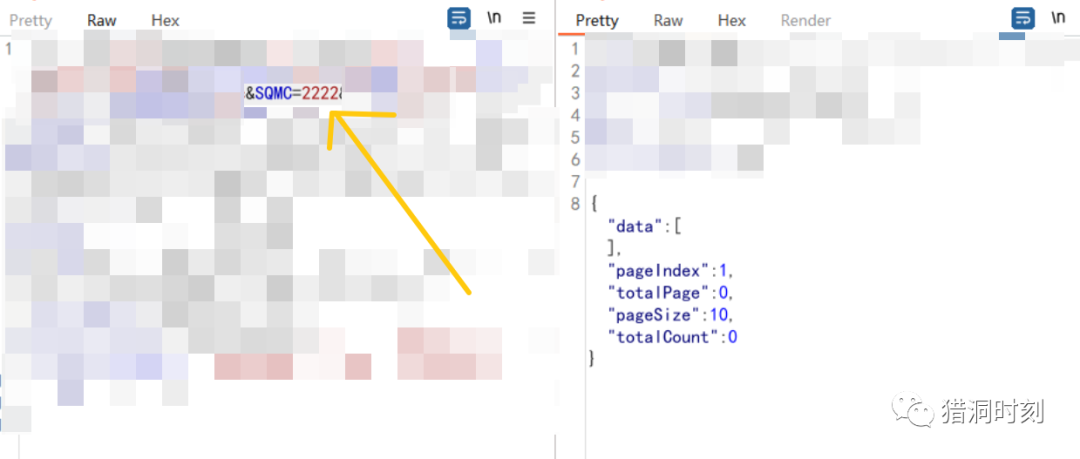
这次的漏洞出现在某服务的办理。



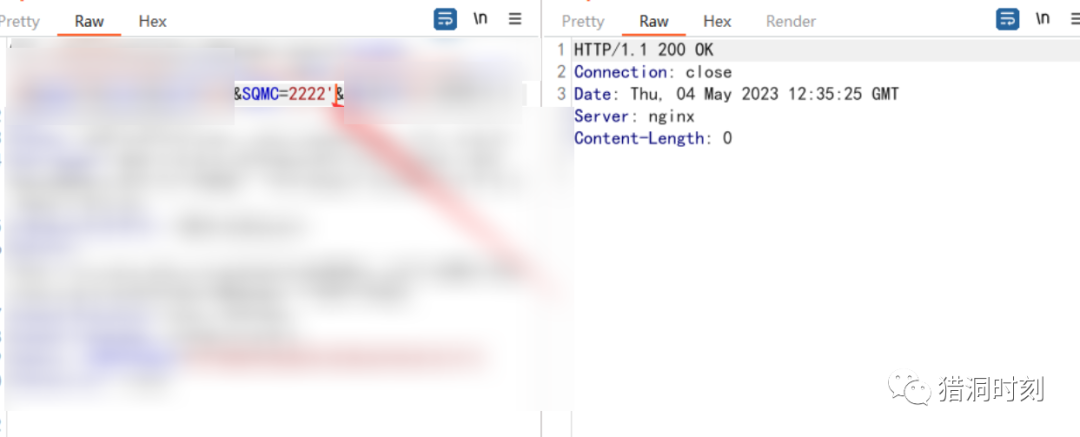
进入办理后，发现这里有一个根据时间进行查询的功能，是不是跟上面的第一个注入漏洞一模一样，没错，这里也存在注入！



点击查询的时候进行抓包，这是正常情况下的返回包



但是在添加一个单引号之后，直接返回为空，发现存在了返回的变化！

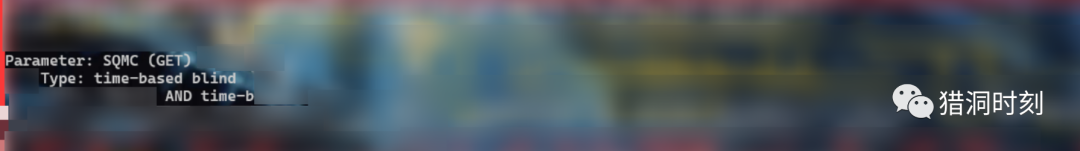


随后使用and ''='进行闭合，发现又成功返回了数据，因此存在注入！



然后就丢入SQLmap中进行漏洞验证。

python .\sqlmap.py -r .\post.txt --tamper=space2comment --dbms 数据库名 -p SQMC



最后因为多次提交同类型漏洞，也是只给了一个积分。。。

