

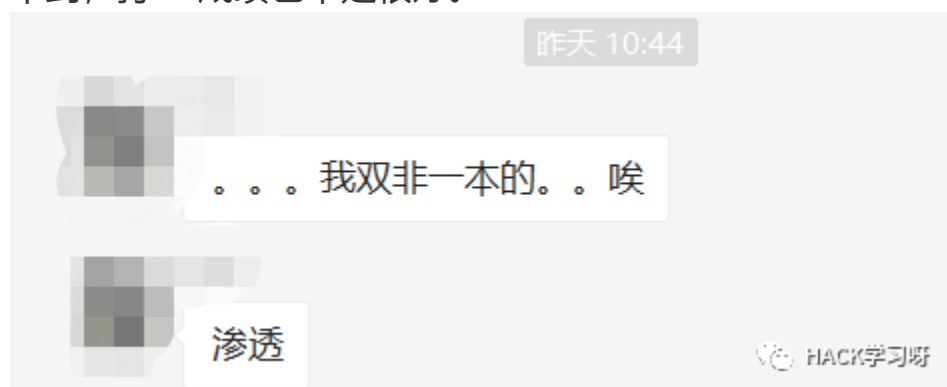
# 记一次代码审计到申请CVE到过程

原创 学员-素念 HACK学习呀

2020-11-05原文

## 0x00起因

这篇文章主要是分享一下我随手提交的一个PHP低质量cve，之所以想写一下的原因是因为这段日子在我渗透技术库群中有一个朋友说到自己双非一本，但是正式工在等保横行的苏州只有4.5k，想挖cve挖不到，打ctf成绩也不是很好。



我想表达的并不是他能力不够，我相信一本的学生整体学习能力是很强的。他可能只是入门晚，缺人指点。缺一个像咱HACK学习呀公众号这种学习指南。

说了这么多，大家可能觉得是废话，但是半路出家没人指点的痛苦我相信各位自学的朋友肯定都深有体会，包括我也是。这里就简单写一下我为申请cve做的一些努力和总结的方法，方便大家学习。

## 0x01 学习思路

凡是自学，百度肯定是少不了的，参考他人的学习方法吸收经验亦是进步。这些是我常用的“搜索引擎”：[www.baidu.com](http://www.baidu.com)、[google](http://google.com)、[www.freebuf.com](http://www.freebuf.com)、[xz.aliyun.com](http://xz.aliyun.com)、[4hou.com](http://4hou.com)、[www.t00ls.net](http://www.t00ls.net)、[www.secquan.org](http://www.secquan.org)、[forum.90sec.com](http://forum.90sec.com)等等

通过这些搜索PHP代码审计，你会发现五花八门的教程。有质量优的，质量差的。质量优的不懂代码审计的你看不来，质量差的看完没进步。

T00LS 圈子社区 90Sec 先知社区 FreeBuf Google 翻译 百度 国家信息安全漏洞共... 新标签页 百度英语

Baidu 百度 php代码审计 百度一下

www.freebuf.com/articles/web/1... 百度快照

**PHP代码审计神器——RIPS个人汉化版 J0o1ey的博客-CSDN博客**

 2020年3月26日 RIPS是一款PHP开发的开源的**PHP代码审计**工具,由国外的安全研究者Johannes Dahse开发,目前开源的最新版本是0.55。程序小巧玲珑,仅有不到500kb,其中的PHP语法分析非常...

CSDN技术社区 百度快照

**PHP代码审计之基础篇 Mi1k7ea-CSDN博客**

 2017年4月18日 最近在学**PHP代码审计**,那便将学习的笔记都整理一遍吧  
~ 前期准备: 当然,最基本的前提是至少大数学过PHP的语法。1、安装相关软件,如Sublime text、Notepad++、ed...

CSDN技术社区 百度快照

其他人还在搜

[php代码审计工具](#) [java代码审计](#) [php代码审计书籍](#) [查看ssh登录日志](#)  
[网站渗透入侵全部教程](#) [fortify](#) [httpcanary高级版破解](#) [数组循环右移](#) [代码审计报告](#)

**PHP代码审计入门篇bluecms - 先知社区**

2019年12月19日 都说bluecms是**代码审计**入门的香饽饽,我这个web菜鸟也来凑个热闹 sql注入 安装上打开目录,随便点了一个ad\_js.php 里面有个sql语句 ...

xz.aliyun.com/t/6... 百度快照

HACK学习呀

搜索PHP代码审计入门，有些是各种漏洞的介绍，但我相信近几年大家学某些漏洞肯定都是从代码层来学的。那这些介绍漏洞的代码审计入门文章对你的帮助其实不是很大了，因为你已经懂了漏洞的形成原理。且如果给你针对的注入上传的函数代码，那这真的太简单了。

## PHP代码审计入门：常见的危险函数和审计点



赞同 7



分享

7 人赞同了该文章

### 01什么是危险函数

函数设计出来就是让人使用的,之所以危险,是因为其功能过于强大.开发人员特别是刚从业的人员很少会完整阅读完整文档,再或者是没有意识到当给这些函数传递一些**非常规的,外部可控**的参数会带来什么影响,所以踩坑的几率非常大.

所以在进行代码审计的时候,比较多的部分都是在审计调用这些危险函数的时候,参数是不是**外部可控**的,有没有进行**正确的过滤**.

PHP获取外界传入的参数都是通过下面几个全局函数的形式,所以审计参数传入经常要和这几个变量打交道

	A	B
1	变量	说明
2	\$_GET	数组,存放着所有通过 URL 参数传递的数据
3	\$_POST	数组,当 HTTP POST 请求的 Content-Type 是 application/x-www-form-urlencoded 或 multipart/form-data 时,把 HTTP Body 的部分解析成关联数组
4	\$_FILES	数组,存放着通过 HTTP POST 上传的文件信息
5	\$_COOKIE	数组,存放着 HTTP 头里 cookie 段内容
6	\$_REQUEST	数组,默认情况下包含了 \$_GET、\$_POST 和 \$_COOKIE 的数据。
7	\$_SERVER	数组,包含了 HTTP 头,服务器环境等信息
8	\$_SESSION	数组,存放当前会话可用的 SESSION 变量
9		

HACK学习呀

而有些文章他们大体都说了代码审计的总体流程,其实你通过这些流程大致已经懂了代码审计中要做什么。可以全局看,可以针对流量去看什么的。

Python爬虫之爬取动态页面数据 23457

通过DVWA学习SQL注入漏洞 19646

分类专栏

Web安全42篇

Kali22篇

安全开发0篇

提权3篇

Metasploit3篇

爬虫4篇

最新评论

使用Kali无线渗透获取宿舍WIFI密码 (WP...

Misaka...: 为什么会爆硬盘啊, 我开任务管理器看对硬盘基本上没什么读写啊?

使用Kali无线渗透获取宿舍WIFI密码 (WP...

豌豆~: 是先连上WIFI吗

使用Kali无线渗透获取宿舍WIFI密码 (WP...

寻址00000001: 硬盘没爆了算你运气好

使用Kali无线渗透获取宿舍WIFI密码 (WP...

clearend: 同道中人兄弟👊👊👊

使用Kali无线渗透获取宿舍WIFI密码 (WP...

寻址00000001: 这个不配合其他方式, 搞不出来。为了破解隔壁的wifi我跑了10天的...

最新文章

博客迁移: )

通过DVWA学习DOM型XSS

phar反序列化漏洞

php代码审计

创作中心

收藏

通读全文法: 麻烦但全面

敏感函数参数回溯法: 高效常用, Seay源代码审计系统

定向功能分析法: 主要根据程序的业务逻辑来审计, 首先是用浏览器逐个访问, 看看程序有哪些功能, 根据相关功能推测可能存在的漏洞

审计的基本流程:

1、整体了解

2、根据定向功能去针对每一项功能进行审计

3、敏感函数参数回溯法

整体了解:

1、网站结构:

浏览源代码文件夹, 了解程序的大致目录。

admin: 后台管理目录

install: 网站的安装目录, 其中的install.sql为数据库的结构信息

sys: 这个目录里面一般存放着配置信息文件和公共函数库, 分别为config.php和lib.php

user: 这里面记录着用户的一些操作, 如用户注册等

index.php: 一般为网页的首页文件, 也是审计的突破口

2、入口文件:

index.php、admin.php文件一般都是整个程序的入口, 通过index.php可以访问到整个系统的架构、运行流程、包含哪些配置文件、哪些过滤文件以及安全过滤文件, 了解程序的基本结构。点赞Mark关注该博主, 随时了解TA的最新博文

点赞7

评论23

分享

收藏33

手机看

打赏

关注

一键三连

HACK学习呀

但你是否觉得自己这样零散的学缺了一个中心点呢? 我称它为树干。这个树干是什么呢? 在我眼中他是代码开发能力, 它决定了你的总体高度。为什么这么说呢, 你想审计它你总得看得懂吧。当然我这里的看得懂, 不是指读懂某些直观漏洞的代码, 实际cms当中会出现各种各样得问题, 一个人这么写, 一个人那么写。你通过学漏洞的代码真的看得懂开发者得整体流程吗? 简单得那些不用说了, 稍微难一点点, 用点对象, 用点框架, 你真的看得懂吗? 看不懂就可以洗洗睡了, 学习人。

## ThinkPHP5学习总结3—— ThinkPHP5框架总览

### 架构总览

ThinkPHP5 基于MVC（模型 — 视图 — 控制器）来组织：



模型与视图，用控制器强制分离，数据请求与展示由控制器统一调配！

#### 1.URL默认采用PATH\_INFO方式

`http://域名 / 入口文件 / 模块 / 控制器 / 操作`

入口文件：应用的入口，如index.php

模块：应用单元，如 user 对应一个目录

控制器：控制单元，如 Index 一般为类文件

操作：执行单元，如 add() 控制器类文件中的执行方法

举例：`http://tp5.com/index.php/index/user/list/ id/10/name/peter`

`id/10/name/peter` 是参数列表

#### 2.常用术语

应用，入口  
模块，控制器，方法，参数  
模型，视图

#### 2.1入口文件（index.php）

（1）入口文件是整个MVC应用的起点

### 公告

昵称： 风意不止

园龄： 2年

粉丝： 0

关注： 6

+加关注

2020年11月						
日	一	二	三	四	五	六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5
6	7	8	9	10	11	12

### 搜索

### 常用链接

我的随笔

我的评论

我的参与

最新评论

HACK学习呀

除了树干之外，一棵树少不了树枝。那树枝是什么呢？树枝就是我上面所说的百度PHP代码审计的东西。

（1）有每种漏洞的利用套路。如我们最基础的漏洞，不同的注入方法，不同的上传绕过方法，不同的xss利用方法，不同的代码执行方式。

## PHP代码审计五（代码执行漏洞）

原创 Mr. Anonymous 2020-05-16 19:55:52 142 收藏 1 原力计划

版权

分类专栏: PHP代码审计 文章标签: php 安全 代码审计

PHP代码审计阶段，基本要理解常见Web漏洞，如Sql注入、XSS、CSRF、文件上传等，近十种安全漏洞的产生原因和初步的防御方法。

### 文章目录

代码执行漏洞

漏洞简介

代码执行漏洞相关函数：

eval()函数

assert() 函数

CTF实例

preg\_replace() 函数

call\_user\_func() 函数

动态函数

PHP中双引号引起的代码执行漏洞

HACK学习呀

(2) 有人家复现的整体流程。通过人家的文章去学习如何从头到尾审计一门cms，先看什么后看什么，哪些需要着重的看。

## 代码审计：审计思路之实例解说全文通读

转载 ru\_li 2016-05-03 14:58:58 3571 收藏 1

分类专栏: 代码审计

在我的新书《代码审计：企业级web代码安全架构》发布之际，借用这篇文章跟大家分享下代码审计的一些思路，目前该书已经可以在淘宝和京东等网站购买。本文章首发在freebuf。

根据敏感关键字来回溯传入的参数，是一种逆向追踪的思路，我们也提到了这种方式的优缺点，实际上在需要快速寻找漏洞的情况下用回溯参数的方式是非常有效的，但这种方式并不适合运用在企业中做安全运营时的场景，在企业中做自身产品的代码审计时，我们需要了解整个应用的业务逻辑，才能挖掘到更多更有价值的漏洞。

全文通读代码也有一定的技巧，并不是随便找文件一个个读完就可以了，这样你是很难真正读懂这套Web程序的，也很难理解代码的业务逻辑，首先我们要看程序的大体代码结构，如主目录有哪些文件，模块目录有哪些文件，插件目录有哪些文件，除了关注有哪些文件，还要注意文件的大小、创建时间。我们根据这些文件的命名就可以大致知道这个程序实现了哪些功能，核心文件是哪些，如下是discuz

(3) 有针对每种漏洞的研究性文章。在这一模块我喜欢直接去做ctf，看各种不同漏洞的高难度研究文章。这一块可以锻炼你的极端环境解决能力，虽然这一块很难很复杂。但是你想走到人家达不到的高处，就得学人家学不会且使用的技能。

## CTF之代码审计汇总

原创

D-R0s1

2018-10-08 22:02:57

👁 4494

★ 收藏 14

版权

分类专栏: CTF WriteUp

web

文章标签:

代码审计

web

最近在做bugku中代码审计的题目，发现了web题目中存在着大量的php漏洞以及弱类型函数的绕过问题，现在借着bugku中的题目来统一的整理一下。希望通过整理可以温故而知新。

**第一题：**extract变量覆盖 焦点：extract(\$\_GET)

```
1 | <?php $flag='xxx'; extract($_GET); if(isset($shiyang)) { $content=trim(file_get_contents($flag)); if($shiyang==$c  
< >
```

该题目中extract函数并没有使用第二个参数，即函数得到两个value相同的key时，后一个产生的变量的值将覆盖前一个变量的值。参考payload：?shiyang&flag

**第二题：**strcmp比较字符串 焦点：if (strcmp(\$\_GET['a'], \$flag) == 0)

```
1 | <?php $flag = "flag{xxxxx}"; if (isset($_GET['a'])) { if (strcmp($_GET['a'], $flag) == 0) //如果 str1 小于 str2 :  
< >
```

这个函数是用于比较字符串的函数 int strcmp ( string \$str1 , string \$str2 ) 参数 str1第一个字符串。str2第二个字符串。

如果 str1 小于 str2 返回 < 0;

如果 str1 大于 str2 返回 > 0;

如果两者相等，返回 0。

重点：任意数组通过strcmp和任意字符串相比较的结果都是NULL

👤 HACK学习呀

这些在你代码足够OK的前提下吃透不是很难，只有当你树干足够强大了，树枝才能长得好。当然你说自己代码一般般，也能审计出漏洞这也是很正常的。

树干在你接触某一门语言时已经开始成长了，你也许在学习漏洞时提升了代码阅读能力，在打ctf时提升了代码阅读能力。让你感觉其实代码审计也挺简单的，这些都是路子。树枝吸收的营养最后还是会给到树干，但对于我这种打ctf脑子转不过来，看人家复现cms漏洞流程越看越懵逼的人，在学习审计时先把代码基础完善比较好。

## 漏洞分析：

我们重点来看报错的堆栈信息：

```
1 in Connection.php line 456
2 at Connection->execute('INSERT INTO `user` (...', []) in Query.php line 241
3 at Query->execute('INSERT INTO `user` (...', []) in Query.php line 2095
4 at Query->insert(['username' => ['inc', 'updatexml(1,concat(0...', '233'])] in Index.php line 17
5 at Index->sql()
6 at ReflectionMethod->invokeArgs(object(Index), []) in App.php line 343
7 at App::invokeMethod(object(Index), 'sql', []) in App.php line 595
8 at App::module(['index', 'index', 'sql'], ['app_host' => '', 'app_debug' => true, 'app_trace' => false]) in App.php line 595
9 at App::exec(['type' => 'module', 'module' => ['index', 'index', 'sql']], ['app_host' => '', 'app_debug' => true, 'app_trace' => false]) in App.php line 595
10 at App::run() in start.php line 19
11 at require('D:\phpstudy\WWW\thin...') in index.php line 17
```

很明显到第五行以后的部分都是框架初始化的部分，我们可以略过。感兴趣可以自行研究。我们重点关心后续SQL执行的操作。

我们看到在第五行调用Index类中的sql方法的时候调用了Query类的insert方法，这个类在 thinkphp\library\think\db\Query.php，2079行。然后我打印这里传入的第一个参数，也就是参数表中的\$data参数，结果如下：

```
array(1) { ["username"]=> array(3) { [0]=> string(3) "inc" [1]=> string(39) "updatexml(1,concat(
```

条条大路通罗马，每条路都能走通的。做ctf的代码题，可以提升自己  
对某些知识点的研究。看人家总结审计漏洞的文章，可以让自己懂得  
开发者经常会犯的BUG！

如果这是LOL，那我会主点开发，副点CTF，再点思路。在这里就是  
我对自己审计的态度，学好开发，打好CTF。

## 0x02 简单分析流程

在这里简单把我之前某个二货十足的洞说一下，主要为了练习自己的  
代码阅读能力。从工具匹配到的漏洞我都去读了一遍。



在ucms1.5.0的/install/index.php中的\$\_POST['mysql\_dbname']存在XSS漏洞。漏洞点在当前文件的226行

```
223 }
224 if ($ifin==0) {
225     echo("<meta http-equiv='Content-Type' content='text/html; charset=UTF-8'>");
226     die('创建数据库失败, 请确认您的mysql账号' . $_POST['mysql_dbname'] . '是否有创建数据库权限, 或者手动建立数据库');
227 }
```

在这里存在着一个触发条件\$ifin等于0，继续往上看。从210行开始对\$ifin进行赋值处理

```
210 //创建数据库
211 if ($_POST['database'] != 'mysqlpdo' || $_POST['database'] != 'mysql') { //mysql数据库
212     $site_db=array('database'=>$_POST['database'],'host'=>$_POST['mysql_host'],'dbname'='', 'user'=>$_POST['mysql_user'],'password'=>$_POST['mysql_password']);
213     $db = new db();
214     $db->connect($site_db);
215     $db->query('sql: CREATE DATABASE IF NOT EXISTS ``' . $_POST['mysql_dbname'] . '``');
216     $alldb = $db->all('sql: SHOW DATABASES');
217     $ifin=0;
218     foreach($alldb as $val) {
219         if ($val['Database']==$_POST['mysql_dbname']) {
220             $ifin=1;
221             break;
222         }
223     }
```

在217行第一次给\$ifin赋值0，然后进入218行的循环判断，判断\$val['Database']是否和传入的'mysql\_dbname'是否相等。如果相等那么\$ifin为1，这样就不会执行我们的die()操作。在这里寻找'mysql\_dbname'不等于\$val['Database']的逻辑条件。

\$val是\$alldb数组中的一个键，在这里看\$alldb数组是如何产生的。回到216行，可以看到他是\$db->all('SHOW DATABASES')后返回的数据。在这里可以看一下\$db类的all()方法。

它将传入的\$sql给如\$this->query()函数

```
1523 public function all($sql)
1524 {
1525     $this->query($sql);
1526     return $this->fetchAll();
1527 }
```

在\$this->query()中使用了原生的查询函数mysql\_query()对传入的\$sql进行查询

```

1442 public function query($sql)
1443 {
1444     $this->querynum++;
1445     if($this->database=='mysql') {
1446         $res = mysql_query($sql,$this->link);
1447     } else {
1448         $res = $this->link->query($sql);
1449     }
1450     if ($res) {
1451         $this->$stmt = $res;
1452         $this->$sql = $sql;
1453         return $this;
1454     }
1455     $this->errorMessage();
1456     Return false;
1457 }

```

HACK学习呀

在这里回到216行，可以明确这里的作用是将后面的代码传入数据库执行。SQL语句的意思是查看所有数据库，那么配合219行之后的内容来说就是当数据库内不存在\$\_POST['mysql\_dbname']的数据库名时\$ifin=0。

```

216     $alldb = $db -> all( sql: 'SHOW DATABASES');
219     if ($val['Database'] == $_POST['mysql_dbname']) {
220         $ifin=1;
221         break;
222     }

```

HACK学习呀

但是这里还有一个点在215行，这里会将\$\_POST['mysql\_dbname']传入的值写入数据库。写入数据库后再 show databases，那么正常逻辑来说这里是不会导致\$ifin=0的。

```

215     $db ->query( sql: 'CREATE DATABASE IF NOT EXISTS `$_POST['mysql_dbname']`.``');

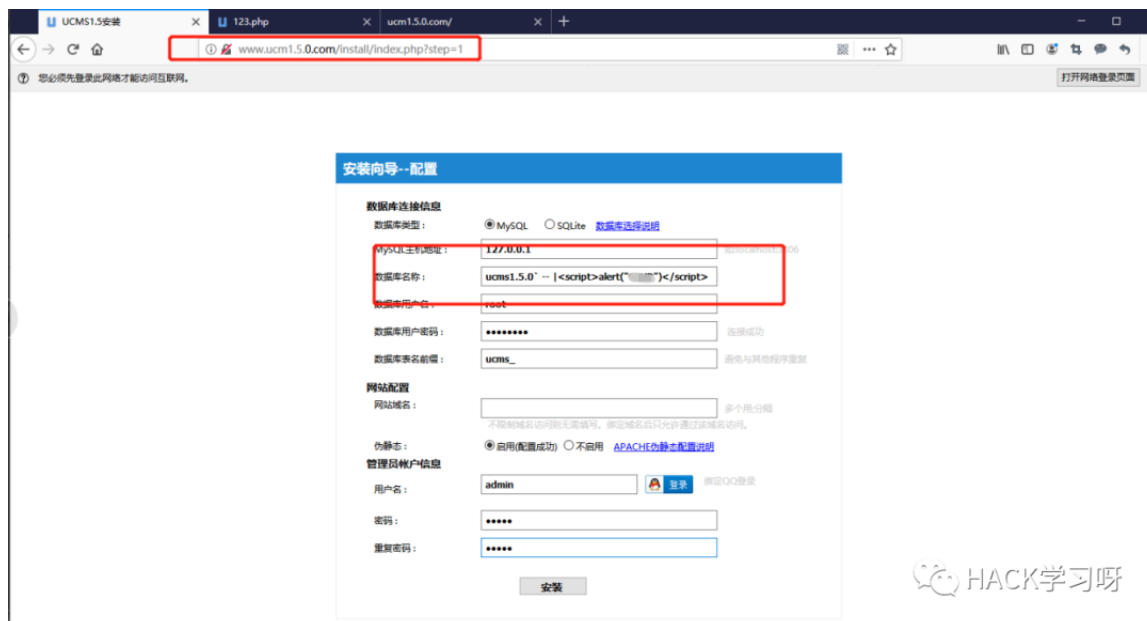
```

将传入的字符写入数据库再查询出来和传入的字符进行比较，如果不等则触发XSS。这里就存在着问题，如何才能让他们两不相等呢？真相就是sql注入的闭合注释！

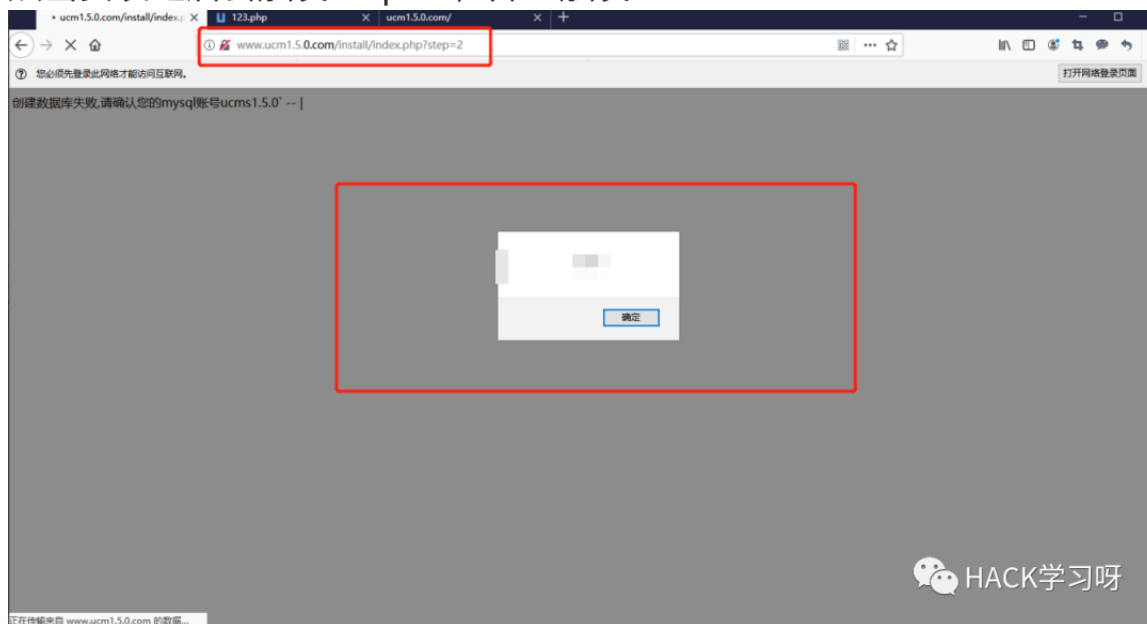
但是这里存在着一个问题，就是闭合符号并不是单引号'，而是`。在这里我们可以传入 ucms1.5.0` --|<script>alert("xss")</script>。这样数据库内写入的就是ucms1.

5.0数据库名，再次show databases查询出来和传入的ucms1.5.0` -

|<script>alert("xss")</script> 进行对比时就会不一样，从而触发XSS!



点击安装之后会触发step=2，并且触发XSS

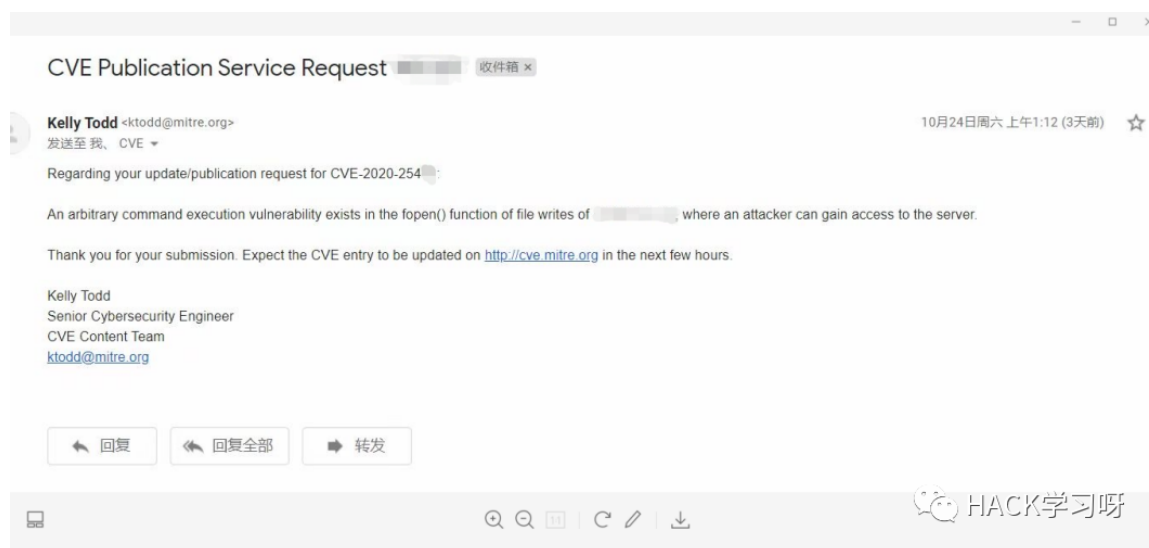


看完是不是觉得这二货写的啥，这洞有什么意思。我自己都有这种感觉，我在干嘛，我是谁。之所以想读还是因为想练习读代码的能力，

就像其他师傅说的。读越多的代码，审越多的洞，以后的大洞都是这些小玩意儿积累起来的。

## 0X03申请过程

我那个cve也就一个上传命令执行，感觉就是基础漏洞。没有搬到文章里的必要，在这里就主要写一下我申请的流程。







在这里我参考了套哥的文章<https://www.freebuf.com/168362.html>，有兴趣的同学可以认证看一下。但是我比较懒，还是直接把必要流程写好了。

直接访问<https://cveform.mitre.org/>申请，这个杂七杂八都要！

← → ↻ https://cveform.mitre.org

**CVE** Common Vulnerabilities and Exposures  
The de facto international standard for vulnerability identification and naming

Follow CVE    

### Submit a CVE Request

\* Required

\* Select a request type

\* Enter your e-mail address

**IMPORTANT:** Please add [cve-request@mitre.org](mailto:cve-request@mitre.org) and [cve@mitre.org](mailto:cve@mitre.org) as safe senders in your email client before completing this form.

Enter a PGP Key (to encrypt)

If you would like us to send an encrypted response, please provide a PGP key up to 20,000 characters. If your PGP key is longer than 20,000 characters, please provide a URL or contact us at [cve@mitre.org](mailto:cve@mitre.org) to identify an alternative solution.

HACK学习呀  
激活 Windows  
转到设置以激活 Windows。

这个看不懂就谷歌翻译，这里选一个申请cveid。

← → ↻ cveform.mitre.org

**CVE** 漏洞识别和命名的事实上的国际标准

关注 CVE    

### 提交CVE请求

\* 必填

\* 选择请求类型

\* 输入您的电子邮件地址

索取CVE ID

索取CVE ID

申请ID请求 (仅适用于CNA)

通知CVE有关发布的信息

要求更新现有的CVE条目

请求有关CVE编号颁发机构 (CNA) 计划的信息

其他

**重要说明:** 将CVE ID分配给您的漏洞后, 除非您提交指向该漏洞的公共信息的URL, 否则它将在CVE列表中发布。没有公共参考, CVE ID将在CVE列表中显示为“保留”。请尽快更新CVE, 并参考该漏洞的详细信息。有关更多信息, 请参见此常见问题解答。

输入PGP密钥 (进行加密)

如果您希望我们发送加密的回复, 请提供最多20,000个字符的PGP密钥。如果您的PGP密钥超过20,000个字符, 请提供URL或通过[cve@mitre.org](mailto:cve@mitre.org)与我们联系, 以找到替代解决方案。

HACK学习呀

其他就按照翻译来填写

### 需要

\* 漏洞类型

\* 产品的供应商

受影响的产品/代码库

\* 产品

\* 版本

[\[-\] 删除](#) [\[+\] 添加](#)

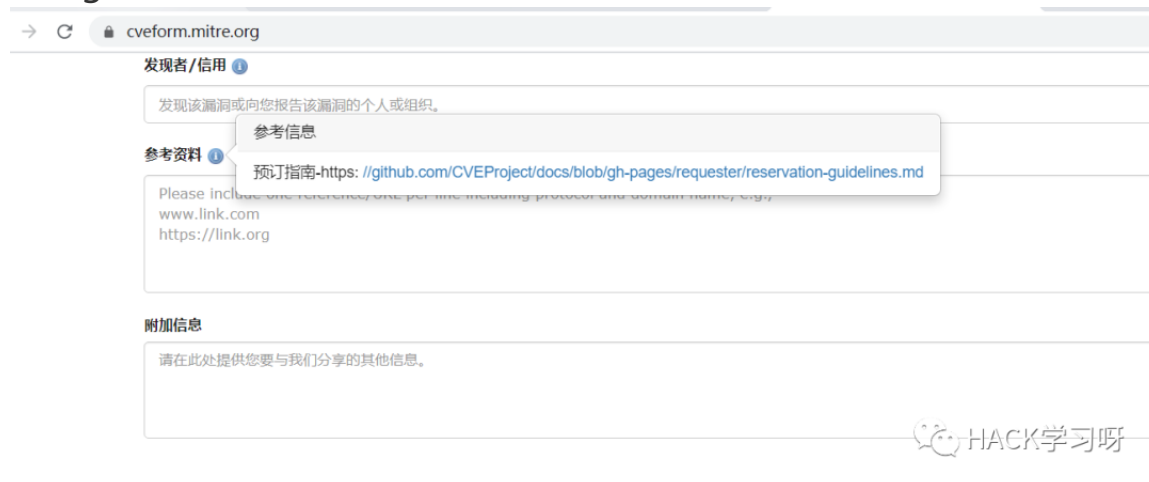
### 可选的

供应商是否已确认或确认该漏洞? ☐ 是 ☒ 没有

攻击类型

HACK学习呀  
激活 Windows

在这里主要就是参考资料模块，因为你可能写了整体的过程。这里框太小放不下，你就可以扔一个自己的github链接，把发现过程写在自己的github里。你放其他平台链接，人家可能不点。



The screenshot shows the CVE form on [cveform.mitre.org](https://cveform.mitre.org). The form is divided into several sections: '发现者/信用' (Discoverer/Credit), '参考资料' (References), and '附加信息' (Additional Information). The '参考资料' section is highlighted with a tooltip that says '参考信息' (Reference Information) and provides a link to the CVE request guidelines: <https://github.com/CVEProject/docs/blob/gh-pages/requester/reservation-guidelines.md>. The '发现者/信用' section has a tooltip that says '发现该漏洞或向您报告该漏洞的个人或组织。' (The person or organization that discovered the vulnerability or reported it to you). The '附加信息' section has a tooltip that says '请在此处提供您要与我们分享的其他信息。' (Please provide any other information you want to share with us here). The form also includes a 'Please include one reference, one per line including protocol and domain name, e.g., www.link.com' instruction and a '预订指南' (Reservation Guide) link.

文章的格式可以按照我这样的来写

```
# BLUECMS 1.6 Value parameter has SQL injection
```

```
## Vulnerability Type :
```

```
SQL Injection
```

```
## Vulnerability Version :
```

```
1.6
```

```
## Recurring environment:
```

- \* Windows 10
- \* PHP 5.4.5
- \* Apache 2.4.23

*## Vulnerability Description AND recurrence:*

这里就写自己的发现过程

申请的内容全部翻译成英文，还有就是人家的审核时间可能不是其他人所说的3-5天。可能是几周，看运气了。

### 0x03 结束寄语

不管是半途出家的还是正规军，学自己学热爱的是一件庆幸的事情。每个人有不同的学习方法，有人适合我说的方法，有人不适合我说的方法。在这里我也只是分享自己学习代码审计的一个思路，低质量cv e其实就是附带的。。当我们有了明确的目标之后，加上合理的学习方法。最后就只要堆时间莽就好了，事半功倍总那么让人感觉舒适。

在这里感谢贝塔安全实验室的Leafer师傅在我学习审计时给出的指导！  
最后祝各位师傅能找到合适的工作，拿着高额的薪水抱着心爱的妹子！



点赞 转发 在看



精选留言

---

用户设置不下载评论