渗透测试之通过代码审计打点 - FreeBuf 网络安全行业门户

> 在渗透测试时，经常会遇到公开漏洞已经修复、只有一个登录框等情况……

前言

在渗透测试时，经常会遇到公开漏洞已经修复、只有一个登录框等情况，这时候如果能拿到代码进行审计一波，往往能打开一扇新的大门。

代码审计 (Code Audit) 顾名思义就是通过阅读源代码，从中找出程序源代码中存在的缺陷或安全隐患，提前发现并解决风险，这在甲方的 SDL 建设中是很重要的一环。而在渗透测试中，可以通过代码审计挖掘程序漏洞，快速利用漏洞进行攻击，达成目标。

审计思路

常见的审计思路有：

1、寻找敏感功能点，通读功能点代码；

优点：精准定向挖掘，利用程度高；

缺点：命名不规范的代码容易被忽略，导致失去先机；

2、根据敏感关键字回溯参数传递过程；

优点：通过搜索敏感关键字可快速定位可能存在的漏洞，可定向挖掘，高效、高质量；

缺点：对程序整体架构了解不够深入，在漏洞定位时比较耗时，逻辑漏洞覆盖不到；

3、直接通读全文代码；

优点：对整体架构熟悉，了解程序的数据流处理、配置文件、过滤函数等；

缺点：耗时长，需要足够的时间熟悉整体架构；

审计方法

按照是否使用 (半) 自动化工具划分，有工具扫描、人工审计和两者相结合的几种审计方式，笔者比较喜欢硬刚，毕竟正常挖业务层面漏洞没有那么复杂，不太需要对底层的一些特性了解的很清楚。

按照数据流向可分为正向审计和逆向审计，正向审计即从功能入口点进行跟踪，一直到数据流处理结束；逆向审计即先根据一些关键词搜索，定位到可能存在风险的关键词 / 函数，再反推到功能入口，看整个处理过程是否存在漏洞。

成功案例

案例一：一个任意文件下载漏洞引起的代码审计

在一次授权测试中，找到一处任意文件下载漏洞，正常思路先下载网站的配置文件，看看数据库是否可以外联

不幸的是不可以外联（其实现在能外联的也很少了），幸运的是在配置文件中收集到了一个域用户，这是意外之喜。

接着下载 Download.aspx 文件看看内容

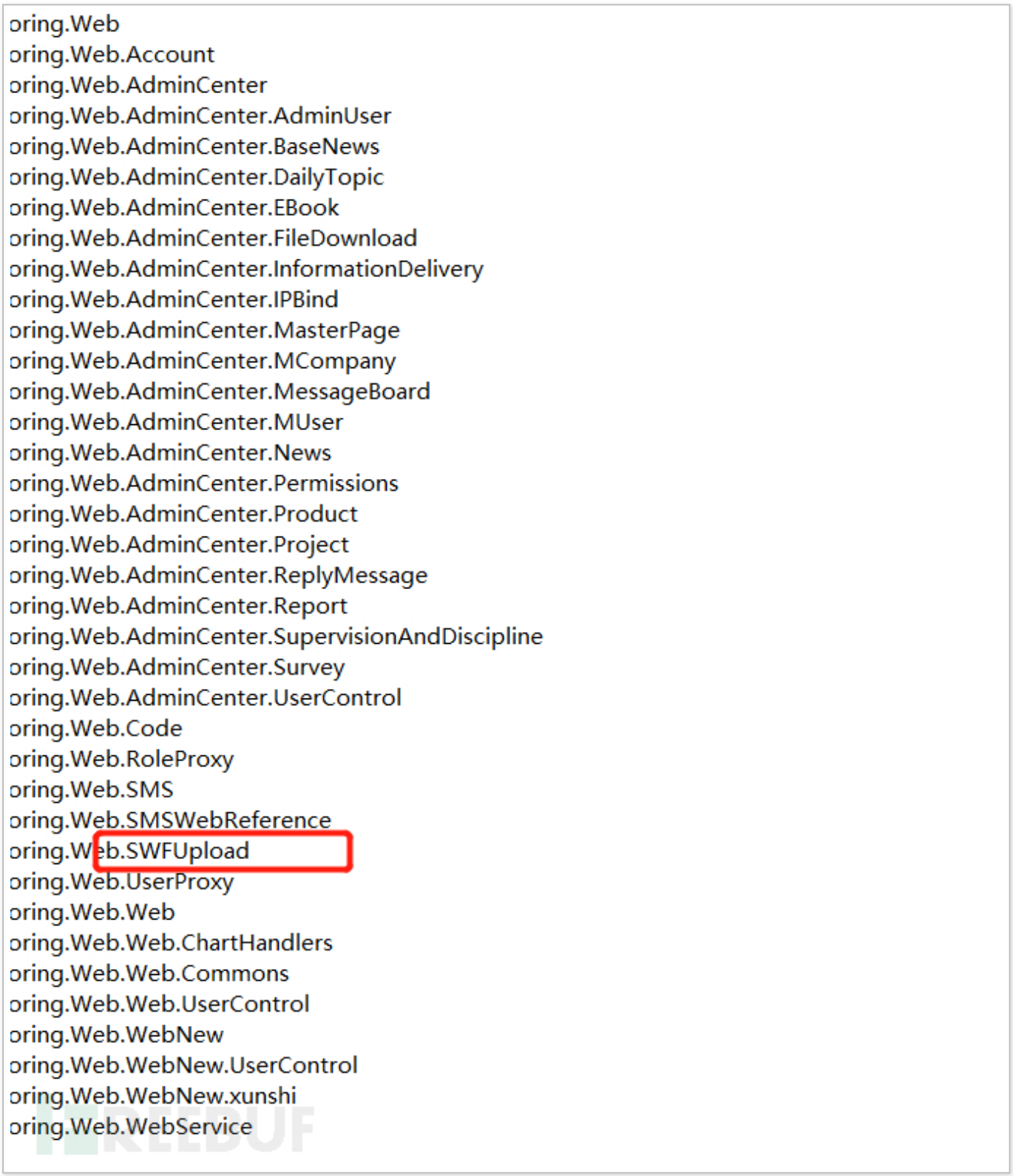
我们知道，.net 是编译型语言，在 aspx 中一般是没有服务代码的，这里看到使用了 Inherits 来继承后端代码。

Inherits 是啥?

MSDN 官方解释：定义供页继承的代码隐藏类。 它可以是从 Page 类派生的任何类。 此特性与 CodeFile 特性一起使用，后者包含指向代码隐藏类的源文件的路径。

通俗一点就是代码都在这个 Inherits 指向的 dll 中了，那么我们下载到 dll 文件，就可以对后端代码进行一波窥探了

于是构造下载 XXXXXX.Web.dll，文件下载到本地后，IL Spy 打开就是一顿肉眼观察



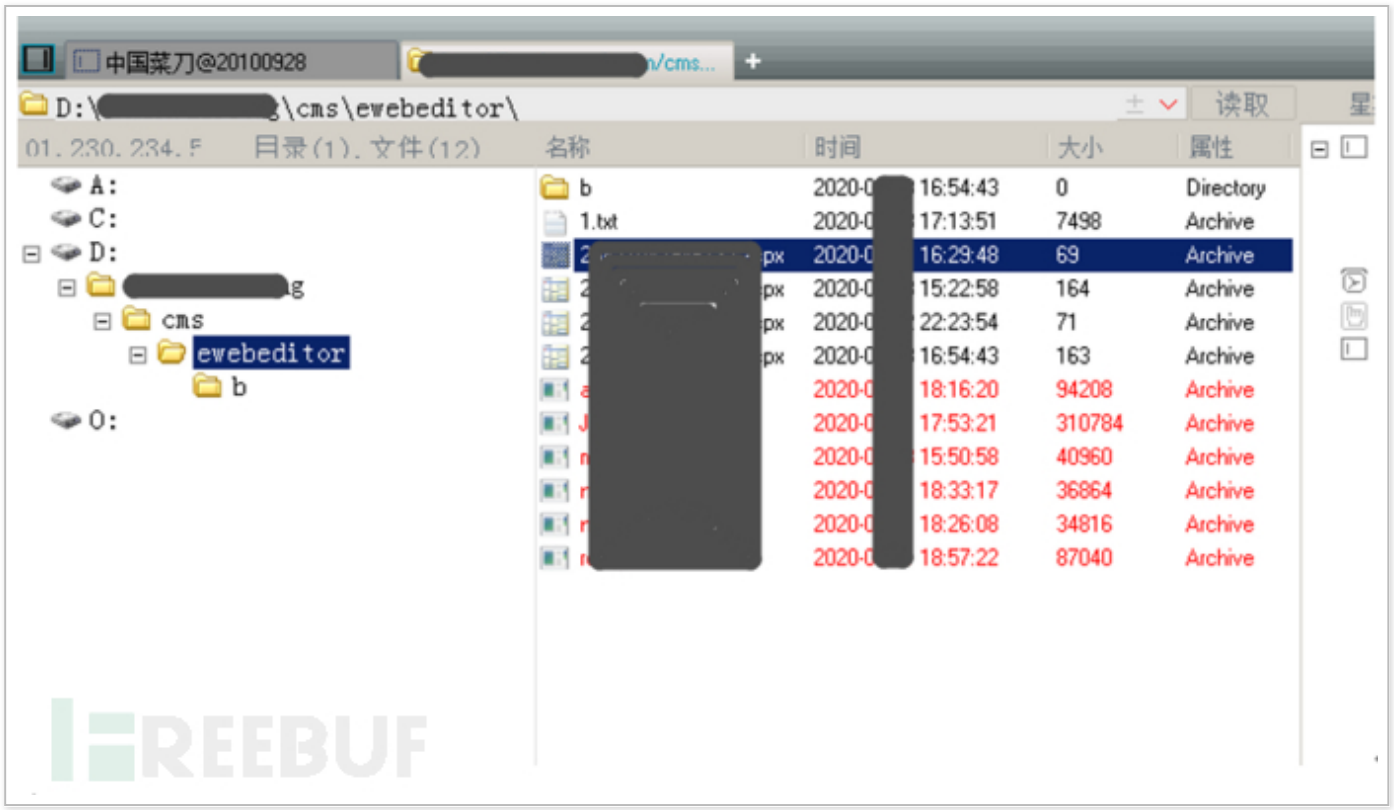
大致看了一下目录，有 Upload 字样，优先级直线上升，先进去看看

```
public void uploadFile()

        {

            string formStringParamValue = SWFUrlOper.GetFormStringParamValue("path");

            string formStringParamValue2 = SWFUrlOper.GetFormStringParamValue("fn");

            bool flag = SWFUrlOper.GetFormStringParamValue("small").ToLower() == "true";
```

```
        ……
        string[] array = new string[]
        {
            "jpg",
            "gif",
            "png",
            "bmp"
        };
        string formStringParamValue3 = SWFUrlOper.GetFormStringParamValue("data");
        try
        {
            System.Web.HttpPostedFile httpPostedFile = base.Request.Files["Filedata"];
            string b = string.Empty;
            string text = string.Empty;
            if (httpPostedFile.ContentLength > 0)
            {
                text = httpPostedFile.FileName;
                if (text.IndexOf(".") != -1)
                {
                    b = text.Substring(text.LastIndexOf(".") + 1, text.Length - text.LastIndexOf(".")
                }
                SWFUploadFile sWFUploadFile = new SWFUploadFile();
                if (flag)
                {
                    sWFUploadFile.set_SmallPic(true);
                    sWFUploadFile.set_MaxWith((formIntParamValue == 0) ? sWFUploadFile.get_MaxWith() :
                    sWFUploadFile.set_MaxHeight((formIntParamValue2 == 0) ? sWFUploadFile.get_MaxHeigh
                }
                sWFUploadFile.set_IsWaterMark(isWaterMark);
                int num = 0;
                string text2 = sWFUploadFile.SaveFile(httpPostedFile, formStringParamValue, formString
                ……
    }
```

38 行进行了文件保存，之前没有对文件的内容、后缀等有任何过滤，开开心心挖到任意文件上传。二话不说本地构造上传个 shell

```
<form name="form" method="post" action="http://xxxx.com/cms/SWFUpload.aspx" enctype="multipart/form-data"

<input type="file" name="Filedata">

<input type="submit" name="Submit" value="upload" ></form>
```



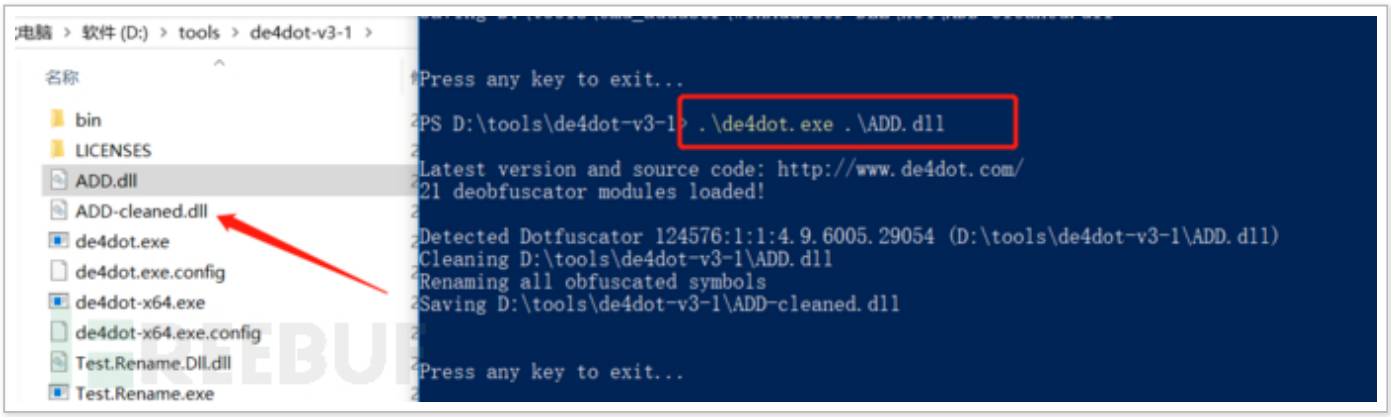
假如这里代码被混淆的话, 可以使用 de4dot 进行反混淆, de4dot 支持 10 几种混淆方式的反混淆：

```
Dotfuscator

.NET Reactor

Xenocode

CryptoObfuscator

SmartAssembly

......
```

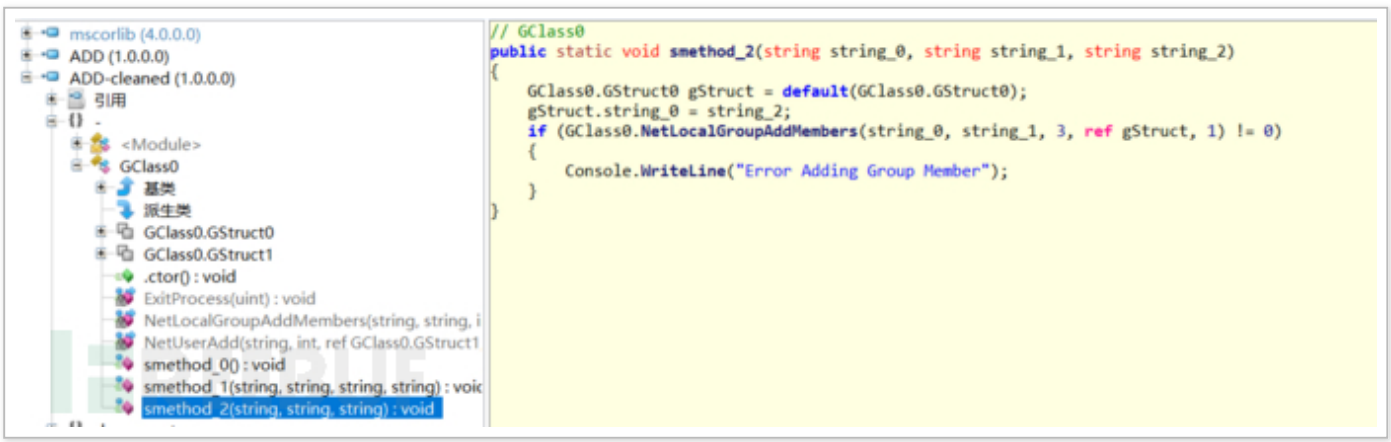
比如使用 Dotfuscator 混淆过的 DLL 是这样的：

使用 de4dot 反混淆：

CMD 命令行执行：de4dot.exe ADD.dll



看下效果：



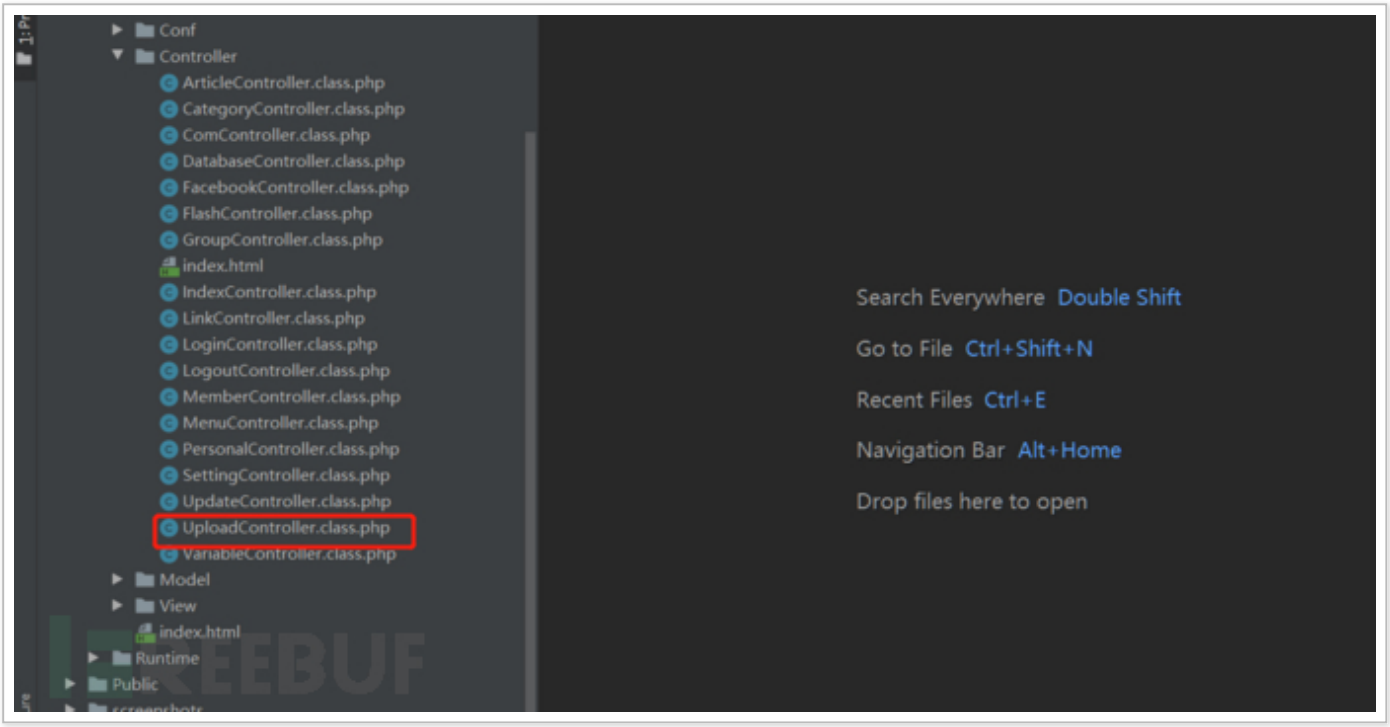
代码已经基本恢复到可读状态了，其他高级用法请参考 github 上的介绍。

案例二：某个开源系统的代码审计

授权渗透时发现只有一个登录框，遇到这种情况一般只能拼字典进行爆破了，还好客户使用的是一套开源系统二次开发，可以 down 到代码进行分析一波。

拿到代码看了下结构，是 thinkphp 的二开，遵循 MVC 模型代码那是一个层次分明。

因为目标只有登录框，所以这里我关注的重点除了文件上传、SQL 注入，又多了一个绕过登录的想法了，no 代码 no 哔哔～～



Upload 太刺眼了，忍不住进去分析一下

```
class UploadController extends ComController{

 ......

private function saveimg($file){

        $uptypes=array(

            'image/jpeg',

            'image/jpg',

            'image/jpeg',

            'image/png',

            'image/pjpeg',

            'image/gif',

            'image/bmp',

            'image/x-png'
```

```
        );
        $max_file_size=2000000;     //上传文件大小限制, 单位BYTE
        $destination_folder='Public/attached/'.date('Ym').'/'; //上传文件路径
        if($max_file_size < $file["size"]){
            echo "文件太大!";
            return null;
        }
        if(!in_array($file["type"], $uptypes)){
            echo "文件类型不符!".$file["type"];
            return null;
        }
        if(!file_exists($destination_folder)){
            mkdir($destination_folder);
        }
        $filename=$file["tmp_name"];
        $image_size = getimagesize($filename);
        $pinfo=pathinfo($file["name"]);
        $ftype=$pinfo['extension'];
        $destination = $destination_folder.time().".".$ftype;
        if (file_exists($destination) && $overwrite != true){
            echo "同名文件已经存在了";
            return null;
        }
        if(!move_uploaded_file ($filename, $destination)) {
            return null;
        }
        return "/".$destination;
    }
    ......
```

只对文件 MIME 类型进行了检测，做安全的都爱这样的开发工程师，比心～

不过这里继承了 ComController，里面有身份认证，不能直接 getshell. .. 呜呜呜，还是要突破登录后台才行。

跟进 ComController 看下认证检查过程

```
class ComController extends BaseController {

    public $USER;

    public function _initialize(){

        C(setting());

        $user = cookie('user');

        $this->USER = $user;

        $url = U("login/index");

        if(!$user){

            header("Location: {$url}");

            exit(0);

        }

        $Auth  =   new Auth();

        $allow_controller_name=array('Upload');//放行控制器名称

        $allow_action_name=array();//放行函数名称

        if(!$Auth->check(CONTROLLER_NAME.'/'.ACTION_NAME,$this->USER['uid'])&&!in_array(CONTROLLER_NAME,$a

            $this->error('没有权限访问本页面!');

        }


        $user = member(intval($user['uid']));

        $this->assign('user',$user);
```

恩... 我可能深深爱上这个开发工程师了， 从 cookie 中获取认证信息，并赋值给 $user 对象，那我们就可以操控用户登录啦，在配合后台的任意文件上传，美滋滋～

失败案例

总结

相对于甲方不同，渗透测试中代码审计更多的是挖掘可利用的漏洞或利用链进行攻击，尽可能的获取更高的权限为目的。

个人觉得渗透测试时关注的漏洞优先级：

命令执行 > 代码执行 > 文件上传 > 文件包含 > SQL 注入 > 文件下载 > 逻辑漏洞 > SSRF > XSS ...

代码审计除了需要了解漏洞的原理、熟悉常见的编程语言、常见的危险函数、常见的协议、渗透技巧外，还需要一些开发调试工具（IDEA、PHPStrom、PyCharm ...），以上内容有不正之处，还请大家斧正。

注:

> de4dot 项目地址：https://github.com/0xd4d/de4dot/

全文完