Web 系统测试

4.5 Web安全测试—文件上传漏洞

河北师范大学软件学院
Software College of Hebei Normal University

目 录

什么是文件上传漏洞

➢什么是文件上传漏洞

- 文件上传漏洞是指由于程序员在对用户文件上传部分的控制不足或者处理缺陷，而导致的用户可以越过其本身权限向服务器上上传可执行的动态脚本文件
- 这里上传的文件可以是木马，病毒，恶意脚本或者WebShell等。这种攻击方式是最为直接和有效的，“文件上传”本身没有问题，有问题的是文件上传后，服务器怎么处理、解释文件。如果服务器的处理逻辑做的不够安全，则会导致严重的后果

概述

- Web应用程序通常会有文件上传功能
- 解析漏洞
  - 攻击者在利用上传漏洞时，通常会与Web容器的解析漏洞配合在一起

Web解析漏洞

➢IIS解析文件

- 在IIS根目录建立文件夹parsing.asp
- 在该文件夹内创建文本文档test.txt
- 在文本文档上写如下内容

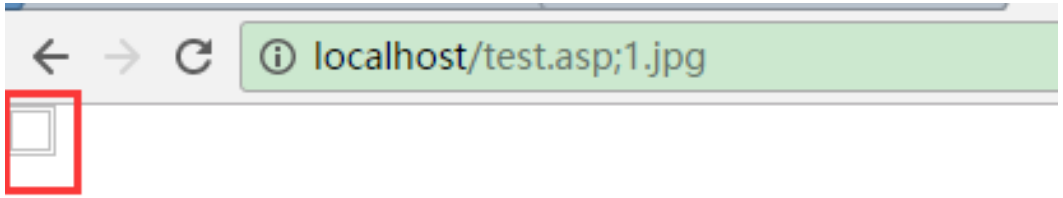
<%=Now()%>

访问http://localhost/parsing.asp/test.txt（老版本IIS显示当时时间）

- 原因：IIS不解析.txt文件，应该直接显示内容，而在parsing.asp中却被当做ASP脚本执行
- 老版本的IIS6中的目录解析漏洞，如果网站目录中有一个/.asp/目录，那么此目录下面的一切内容都会被当作asp脚本来解析

Web解析漏洞

- 当文件为文件名为*.asp;1.jpg
- 访问http://localhost/test.asp;1.jpg



- 原因：IIS在解析文件名的时候可能将分号后面的内容丢弃，当做.asp文件被解析
- 攻击者利用：在上传的时候给后面加入分号内容来避免黑名单过滤

Web解析漏洞

- 旧版Windows Server中存在空格和dot漏洞类似于 a.php. 和 a.php[空格] 这样的文件名存储后会被windows去掉点和空格，从而使得加上这两个东西可以突破过滤，成功上传，并且被当作php代码来执行

认识解析漏洞

➢Apache解析漏洞

- ●新建文本文件，写如下内容

```
<?php
        phpinfo();
?>
```

- ●保存文件为1.php.rar
- ●浏览器地址栏输入 http://localhost/1.php.rar，正常应该提示文件下载框，但此时显示phpinfo()的内容
- ●Apache解析文件时，当碰到不认识的扩展名时，将会从后向前解析直到碰到认识的扩展名为止，如果都不认识，则会暴露其源代码
- ●攻击者利用：将恶意脚本打包成xxx.php.rar

造成文件上传漏洞的原因

- 对于上传文件的后缀名（扩展名）没有做较为严格的限制
- 对于上传文件的MIMETYPE(用于描述文件的类型的一种表述方法) 没有做检查
- 权限上没有对于上传的文件目录设置不可执行权限
- web server对于上传文件或者指定目录的行为没有做限制

绕过上传漏洞

➢防止上传漏洞两种策略

- 客户端检测：客户端使用JS检测，在文件未上传时，就对文件进行验证
- 服务器端检测：检测文件扩展名是否合法，检测文件中是否嵌入恶意代码

文件上传漏洞预防

- 白名单与黑名单过滤
- MIME（MultiPupose Internet Mail Extensions）验证（用来设定某种扩展名文件的打开方式）类型验证
  - CSS文件MIME类型为text/css
  - gif图片MIME类型为text/gif
- 目录验证
  - 接收文件后，对目录进行判断

如何检验有没有解析漏洞

- 上传要求（正确的）的文件类型
- 上传带有脚本的伪造成 txt,jpg 文件上传验证

文件上传防御方式

- 前端限制
- 在表单中使用

  onsumbit=check()调用js函数来检查上传文件的扩展名

```
function check(){

        var filename=document.getElementById("file");

        var str=filename.value.split(".");

        var ext=str[str.length-1];

        if(ext=='jpg'||ext=='png'||ext=='jpeg'||ext=='gif'){

            return true;

        }else{

            alert("这不是图片！")

            return false;

        }

        return false;
```

文件上传防御方式

➢前端检测绕过方法

- 绕过前台脚本检测扩展名，就是将所要上传文件的扩展名更改为符合脚本检测规则的扩展名，通过工具，截取数据包，并将数据包中文件扩展名更改回原来的，达到绕过的目的
- 如果是JS脚本检测，在本地浏览器客户端禁用JS即可。可使用火狐浏览器的NoScript插件、IE中禁用掉JS等方式实现

文件上传防御方式

➢检查扩展名

- ●在文件被上传到服务端的时候，对于文件名的扩展名进行检查，如果不合法，则拒绝这次上传
- ●检查扩展名是否合法的时候，有两种策略：

1.黑名单策略，文件扩展名在黑名单中的为不合法

2.白名单策略，文件扩展名不在白名单中的均为不合法

文件上传防御方式

➢黑名单策略略

```php
$postfix = end(explode('.','$_POST['filename']);

if($postfix=='php'||$postfix=='asp'||$postfix=='sh'){

  echo "invalid file type";

  return;

}
```

文件上传防御方式

➢白名单策略

```php
$postfix = end(explode('.','$_POST['filename']);

if($postfix=='jpg'||$postfix=='png'||$postfix=='gif'){

  //save the file and do something next

} else {

  echo "invalid file type";

  return;

}
```

文件上传防御方式

➢黑名单、白名单哪种更安全？

- 白名单策略是更加安全的，通过限制上传类型为只有我们接受的类型，可以较好的保证安全，因为黑名单我们可以使用各种方法来进行注入和突破
- 原理：当浏览器将文件提交到服务器端的时候，服务器端会根据设定的黑白名单对浏览器提交上来的文件扩展名进行检测，如果上传的文件扩展名不符合黑白名单的限制，则不予上传，否则上传成功

文件上传防御方式

- 导致文件上传漏洞的根本原因在于服务把用户上传的本应是数据的内容当作了代码，一般来说，用户上传的内容都会被存储到特定的一个文件夹下，比如我们很多人习惯于放在 ./upload/ 下面要防止数据被当作代码执行，我们可以限制web server对于特定文件夹的行为

文件上传防御方式

- 在默认情况下，对与 .php文件Apache会当作代码来执行，对于html,css,js文件，则会直接由HTTP Response交给客户端程序对于一些资源文件，比如txt，doc，rar等等，则也会以文件下载的方式传送的客户端。我们希望用户上传的东西仅仅当作资源和数据而不能当作代码
- 因此可以使用服务器程序的接口来进行限制

文件上传防御方式

- 禁止脚本执行的方式

1.指定特定扩展名的文件的处理方式,原理是指定Response的Content-Type可以加上如下几行

```
AddType text/plain .pl .py .php
```

这种情况下，以上几种脚本文件会被当作纯文本来显示出来，你也可以换成其他的Content-Type

文件上传防御方式

➢禁止脚本执行的方式

2.如果要完全禁止特定扩展名的文件被访问，用下面的几行

```
Options -ExecCGI
AddHandler cgi-script .php .pl .py .jsp .asp .htm .shtml .sh .cgi识别
```

在这种情况下，以上几种类型的文件被访问的时候，会返回403 Forbidden的错误

文件上传防御方式

➢禁止脚本执行的方式

3.也可以强制web服务器对于特定文件类型的处理，下面的方法直接强行让apache将文件识别为你指定的类型

```
<FilesMatch "\.(php|pl|py|jsp|asp|htm|shtml|sh|cgi)$">
ForceType text/plain
</FilesMatch>
```

符合上面正则的全部被认为是纯文本，也可以继续往里面加入其他类型

文件上传防御方式

➢禁止脚本执行的方式

4.只允许访问特定类型的文件

```
<Files ^(*.jpeg|*.jpg|*.png|*.gif)>
order deny,allow
deny from all
</Files>
```

- 在一个上传图片的文件夹下面，就可以加上这段代码，使得该文件夹里面只有图片扩展名的文件才可以被访问，其他类型都是拒绝访问。
- 这又是一个白名单的处理方案，永远记得，白名单是最有保障的安全措施

文件上传防御方式

- 其他方式——绕过
- 原理：部分程序员的思维不严谨，并使用逻辑不完善的上传文件合法性检测手段，导致可以攻击者找到方式绕过其检测方式

```
1. 后缀名大小写绕过
用于只将小写的脚本后缀名(如php)过滤掉的场合；
例如:将Burpsuite截获的数据包中的文件名【evil.php】改为【evil.Php】

2. 双写后缀名绕过
用于只将文件后缀名过滤掉的场合，例如"php"字符串过滤的；
例如:上传时将Burpsuite截获的数据包中文件名【evil.php】改为【evil.pphphp】，那么过滤了第一个"php"字符串

3. 特殊后缀名绕过
用于检测文件合法性的脚本有问题的场合；
例如:将Burpsuite截获的数据包中【evil.php】名字改为【evil.php6】，或加个空格改为【evil.php 】等。
```

防范文件上传漏洞常见方法

➢防范文件上传漏洞常见的几种方法：

1.文件上传的目录设置为不可执行

只要web容器无法解析该目录下面的文件，即使攻击者上传了脚本文件，服务器本身也不会受到影响，因此这一点至关重要。

2.判断文件类型

在判断文件类型时，可以结合使用MIME Type、后缀检查等方式。在文件类型检查中，强烈推荐白名单方式，黑名单的方式已经无数次被证明是不可靠的。此外，对于图片的处理，可以使用压缩函数或者resize函数，在处理图片的同时破坏图片中可能包含的HTML代码

防范文件上传漏洞常见方法

3.使用随机数改写文件名和文件路径

文件上传如果要执行代码，则需要用户能够访问到这个文件。在某些环境中，用户能上传，但不能访问。如果应用了随机数改写了文件名和路径，将极大地增加攻击的成本。再来就是像**shell.php.rar.rar**和**crossdomain.xml**这种文件，都将因为重命名而无法攻击

系统开发阶段的防御

- 系统开发人员应有较强的安全意识，尤其是采用PHP语言开发系统。在系统开发阶段应充分考虑系统的安全性。对文件上传漏洞来说，最好能在客户端和服务器端对用户上传的文件名和文件路径等项目分别进行严格的检查。客户端的检查虽然对技术较好的攻击者来说可以借助工具绕过，但是这也可以阻挡一些基本的试探。服务器端的检查最好使用白名单过滤的方法，这样能防止大小写等方式的绕过，同时还需对%00截断符进行检测，对HTTP包头的content-type也和上传文件的大小也需要进行检查

系统运行阶段的防御

- 系统上线后运维人员应有较强的安全意思，积极使用多个安全检测工具对系统进行安全扫描，及时发现潜在漏洞并修复
- 定时查看系统日志，web服务器日志以发现入侵痕迹
- 定时关注系统所使用到的第三方插件的更新情况，如有新版本发布建议及时更新，如果第三方插件被爆有安全漏洞更应立即进行修补
- 对于整个网站都是使用的开源代码或者使用网上的框架搭建的网站来说，尤其要注意漏洞的自查和软件版本及补丁的更新，上传功能非必选可以直接删除
- 除对系统自生的维护外，服务器应进行合理配置，非必选一般的目录都应去掉执行权限，上传目录可配置为只读

总结

- 什么是文件上传漏洞
- 文件上传漏洞解析
- 文件上传漏洞预防

Question