目录遍历漏洞总结

一、目录遍历漏洞介绍

1.什么是目录遍历漏洞

先上一张图看看什么是目录遍历漏洞:

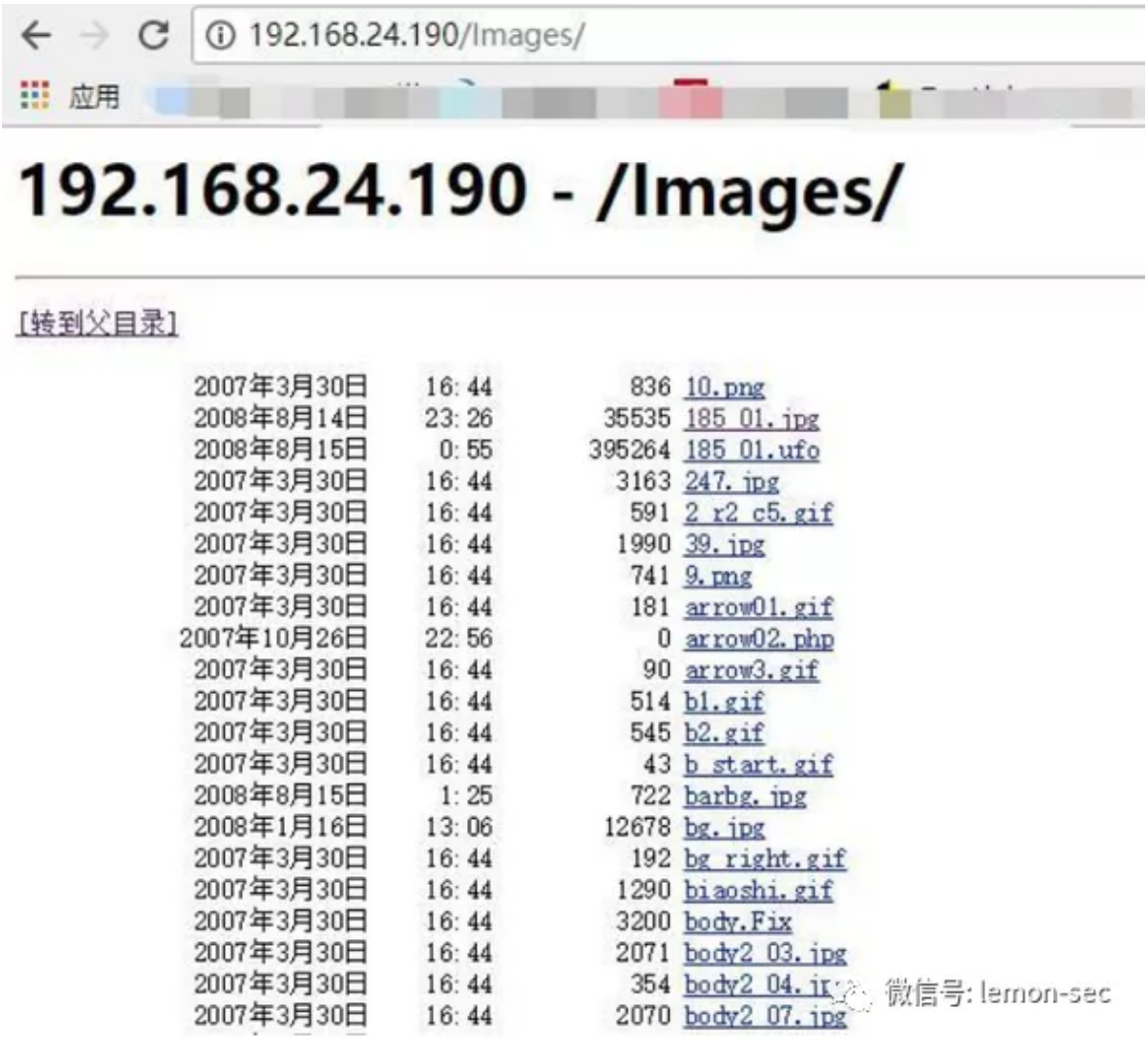
目录遍历就长成这样子。

一般遇到目录遍历漏洞，我们常做的就是去寻找有价值的东西去下载，比如下载数据库的敏感信息。

一般是没有index.php就可能出现像这样的一个目录遍历的漏洞，但是一般情况下index文件都会有的。

2.怎么寻找目录遍历漏洞

那么怎么去找目录遍历漏洞，一般是输入到文件目录，看页面响应。比如站点上的一张图片的的连接为：http://192.168.24.190/Images/185_01.jpg，我们把图片删除，只保留目录：http://192.168.24.190/Images/，最后浏览器看看:



那这样子就是存在目录遍历漏洞。

二、漏洞分析

1.目录遍历漏洞存在的原因

目前许多的Web应用程序一般会有对服务器的文件读取查看的功能，大多会用到提交的参数来指明文件名，形如:http://www.nuanyue.com/getfile=image.jgp。

当服务器处理传送过来的image.jpg文件名后，Web应用程序即会自动添加完整路径，形如“d://site/images/image.jpg”，将读取的内容返回给访问者。

初看，这只是文件交互的一种简单的过程，但是由于**文件名可以任意更改而服务器支持“~/”，“/..”等特殊符号的目录回溯**，从而使攻击者越权访问或者覆盖敏感数据，如网站的配置文件、系统的核心文件，这样的缺陷被命名为路径遍历漏洞。在检查一些常规的Web应用程序时，也常常有发现，只是相对隐蔽而已。

目录遍历漏洞的发现，主要是对Web应用程序的文件读取交互的功能块，进行检测，面对这样的读取方式:“http://www.nuanyue.com/test/downfile.jsp?filename=fan.pdf”。我们可以使用 “../”来作试探，比如提交Url: `getfile=/fan/fan/*53.pdf`，而系统会解析成 `d://site/test/pdf/fan/fan/../../*53.pdf`，通过“../”跳转目录“/fan”，即“d://site/test/pdf/*53.pdf”，返回了读取文件的正常的页面。

2.目录遍历漏洞存在的位置

目录遍历漏洞隐藏一般在文件读取或者展示图片功能块这样的通过参数提交上来的文件名，从这可以看出来过滤交互数据是完全有必要的。恶意攻击者当然后会利用对文件的读取权限进行跨越目录访问，比如访问一些受控制的文件，“../../../../../../etc/passwd”或者“../../../../boot.ini”，当然现在部分网站都有类似Waf的防护设备，只要在数据中会有/etc /boot.ini等文件名出直接进行拦截。

三、目录遍历漏洞绕过

路径遍历漏洞是很常见的，在Web应用程序编写过程，会有意识的对传递过来的参数进行过滤或者直接删除，存在风险的过滤方式，一般可以采用如下方式进行突破：

以下是一些绕过的方法，当然在实际运行过程中，可以组合使用。

1.加密参数传递的数据

在Web应用程序对文件名进行加密之后再提交，比如：“downfile.jsp?filename=ZmFuLnBkZg-”，在参数filename用的是Base64加密，而攻击者要想绕过，只需简单的将文件名加密后再附加提交即可。所以说，采用一些有规律或者轻易能识别的加密方式，也是存在风险的。

2.编码绕过

尝试使用不同的编码转换进行过滤性的绕过，比如Url编码，通过对参数进行Url编码提交，“downfile.jsp?filename=%66%61%6E%2E%70%64%66”来绕过。

3.目录限定绕过

在有些Web应用程序是通过限定目录权限来分离的。当然这样的方法不是十分可取的，攻击者可以通过某些特殊的符号“~“来绕过。形如这样的提交“downfile.jsp?filename=~/../boot”。通过这样一个符号，就可以直接跳转到硬盘目录下了。

4.绕过一些文件后缀过滤

一些Web应用程序在读取文件前，会对提交的文件后缀进行检测，攻击者可以在文件名后放一个空字节的编码，来绕过这样的文件类型的检查。例如：../../../../boot.ini%00.jpg，Web应用程序使用的Api会允许字符串中包含空字符，当实际获取文件名时，则由系统的Api会直接截短，而解析为“../../../../boot.ini”。在类Unix的系统中也可以使用Url编码的换行符，例如：../../../etc/passwd%0a.jpg如果文件系统在获取含有换行符的文件名，会截短为文件名。也可以尝试%20，例如:../../../index.jsp%20。

5.绕过来路验证

在一些Web应用程序中，会有对提交参数的来路进行判断的方法，而绕过的方法可以尝试通过在网站留言或者交互的地方提交Url再点击或者直接修改Http数据包中的Referer即可，这样可以进行绕过的原因是Http数据包中的Referer是由客户端浏览器发送的，服务器是无法控制的，而将此变量当作一个值得信任源是错误的。

四、漏洞防范

在防范遍历路径漏洞的方法中，**最有效的是权限的控制。**谨慎的处理向文件系统API传递过来的参数路径。主要是因为大多数的目录或者文件权限均没有得到合理的配置，而Web应用程序对文件的读取大多依赖于系统本身的API，在参数传递的过程，如果没有得严谨的控制，则会出现越权现象的出现。在这种情况下，Web应用程序可以采取以下防御方法，最好是组合使用。

1.数据净化

数据净化，对网站用户提交过来的文件名进行硬编码或者统一编码，对文件后缀进行白名单控制，对包含了恶意的符号或者空字节进行拒绝。

2.使用chrooted环境

Web应用程序可以使用chrooted环境访问包含被访问文件的目录，或者使用绝对路径+参数来控制访问目录，使其即使是越权或者跨越目录也是在指定的目录下。

五、总结

路径遍历漏洞允许恶意攻击者突破Web应用程序的安全控制，直接访问攻击者想要的敏感数据，包括配置文件、日志、源代码等，配合其它漏洞的综合利用，攻击者可以轻易的获取更高的权限，并且这样的漏洞在发掘上也是很容易的，只要对Web应用程序的读写功能块直接手工检测，通过返回的页面内容来判断，是很直观的，利用起来也相对简单。