

文件上传漏洞的攻击方法与防御措施研究

郝子希^{1 2}, 王志军¹, 刘振宇²

(1. 东华大学 计算机科学与技术学院, 上海 200050;

2. 上海计算机软件技术开发中心 上海市计算机软件评测重点实验室, 上海 201112)

摘要: 简述了当今社会信息安全的重要性, 说明了渗透测试技术中文件上传漏洞的基本原理, 列举了文件上传漏洞能够造成的危害, 对文件上传漏洞进行详细分析。由于文件上传漏洞一般伴随着服务器解析漏洞出现, 结合三种不同的 Web 应用容器(IIS、Apache、PHP)的解析漏洞, 解释文件上传漏洞与服务器解析漏洞之间的关系, 详细说明文件上传漏洞出现的原因; 从 Web 站点的两种上传文件验证方式—客户端验证和服务器端验证阐述了相应的攻击技巧, 通过对五种攻击方法(绕过客户端验证、绕过黑名单与白名单验证、绕过 MIME 验证、绕过目录验证和截断上传攻击)的具体实验描述了对文件上传漏洞的攻击过程, 并给出了实验代码; 最后针对实验中的攻击方法, 提出了四类文件上传漏洞的有效防御措施, 并对全文进行总结, 对未来提出展望。

关键词: 渗透测试; 文件上传; 网络安全; 网络攻击; Web 漏洞

中图分类号: TP309

文献标识码: A

文章编号: 1673-629X(2019)02-0129-06

doi: 10.3969/j.issn.1673-629X.2019.02.027

Research on Attack Method and Defensive Measure of File Upload Vulnerabilities

HAO Zi-xi¹, WANG Zhi-jun¹, LIU Zhen-yu²

(1. School of Computer Science and Technology, Donghua University, Shanghai 200050, China;

2. Key Laboratory of Computer Software Testing & Evaluating of Shanghai, Shanghai Computer Software Technology Development Center, Shanghai 201112, China)

Abstract: We briefly describe the importance of social network security in today's society, explain the basic principle of file upload vulnerability in penetration testing technology, list the hazards caused by file upload vulnerability, and analyze the file upload vulnerability in detail. Because the file upload vulnerability generally accompanies server parsing vulnerability, combined with the parsing vulnerabilities of three different Web application containers (IIS, Apache and PHP), we explain the relationship between file upload vulnerability and server parsing vulnerability, and the reasons why the vulnerability can be caused. Based on two ways to verify the uploaded files from Web sites: client authentication and server-side authentication, we illustrate the attack techniques, and based on five attack methods (bypassing client authentication, bypassing blacklists and whitelisting, bypassing MIME verification, bypassing directory verification, and truncating upload attack), we describe the process of attack on file upload vulnerability and give the experimental code. Finally, aiming at the attack methods in the experiment, we put forward four kinds of defensive measures for file upload vulnerability, summarizing full text, prospecting for future.

Key words: penetration test; file uploading; network security; network attack; Web vulnerability

0 引言

“互联网+”推动了各行各业与互联网的融合, 各行各业大到工业、金融、商贸, 小到娱乐、民生、家庭, 与互联网的联系越来越紧密, 形成创新 2.0 下的互联网

发展的新业态。但“互联网+”在带动国内互联网经济的同时, 也为通过网络传播的有害信息提供了更大的空间, 攻击者的攻击手段也在不断更新。2017 年 6 月 1 日, 国家开始实施《中华人民共和国网络安全法》并

收稿日期: 2018-03-08

修回日期: 2018-07-12

网络出版时间: 2018-11-15

基金项目: 国家自然科学基金(61502299)

作者简介: 郝子希(1993-) 男, 硕士研究生, 研究方向为信息安全; 王志军, 博士, 副教授, 研究方向为信息系统与信息安全管理; 刘振宇, 博士, 研究员, 研究方向为软件测试、软件质量、性能分析与调优。

网络出版地址: <http://kns.cnki.net/kcms/detail/61.1450.TP.20181115.1047.036.html>

对互联网和信息系统的的功能做出了明确的要求,网络安全的防御工作更加迫切。

近年来,虽然国内几乎所有企业都开发了自己的 Web 应用系统,但是这些 Web 应用是重点致力于方便客户使用,给予客户更方便、快捷的服务支持,因此现在的开发人员更注重的是 Web 应用的功能和性能,而在非常关键的安全性方面,却没有付诸足够的重视。渗透测试指的是一种通过模拟恶意攻击者的技术与方法,对目标系统进行攻击,绕过或挫败目标系统的安全控制措施,从而达到取得访问控制权的目的,并发现具备业务影响后果安全隐患的一种安全测试与评估方式^[1]。文件上传漏洞作为渗透测试过程中经常出现的 Web 漏洞,正在严重威胁着网络空间的安全。攻击者可以通过文件上传漏洞窃取服务器的信息,种植木马,在受害服务器网站上传播恶意信息,甚至获取服务器的管理员权限。

1 文件上传漏洞概述

文件上传漏洞指的是,由于程序员对信息系统中用户上传文件的模块的控制不足或者文件上传模块本身存在处理缺陷而导致的漏洞,攻击者可能利用此类漏洞越过其本身权限向服务器中上传可执行的动态脚本文件,并通过该文件对目标系统进行进一步的恶意操作。例如,某信息系统使用的是 Linux 服务器,并且以 php 作为服务器端的动态网站环境,那么在信息系统有上传文件功能的地方,就不能允许用户上传后缀为*.php 的文件,否则攻击者可以直接上传一个 php 语言编写的 Webshell,服务器就被攻击者入侵了。

文件上传后导致的常见安全问题一般有:

(1) 上传的文件是 Web 脚本语言,服务器的 Web 容器解释并执行了用户上传的脚本,导致代码执行;

(2) 上传的文件是 Flash 的策略文件 crossdomain.xml,攻击者用以控制 Flash 在该域下的行为,其他通过类似方式控制策略文件的情况类似;

(3) 上传的文件是病毒、木马文件,用于诱骗用户或者管理员下载、执行;

(4) 上传的文件是钓鱼图片或包含了脚本的图片,在某些版本的浏览器中会被作为脚本执行,被用于钓鱼和欺诈。

2 文件上传漏洞和常见的解析漏洞

在文件上传漏洞中,攻击者上传了一个可执行的脚本文件,并通过此文件进行进一步的恶意操作,这种攻击方法是最直接和有效的。但是通常在信息系统中,“文件上传”功能本身是没有问题的。但是如果攻击者上传的是 php 等脚本语言编写的恶意文件,那么

就需要注意恶意文件上传后,服务器怎样处理和解释该文件。如果攻击者上传恶意文件后没办法执行,那么攻击行为是没有意义的。所以,文件上传漏洞一般和服务器的解析漏洞一起存在。以下介绍几种常见的 Web 容器中存在的文件解析漏洞。

2.1 IIS 5.x-6.x 中的解析漏洞

IIS 的全称是 Internet Information Services,意思是互联网信息服务,它是一个万维网服务器,意味着你可以通过 IIS 发布网页,并且由 Asp、Java、VB 脚本产生页面。使用 IIS5.x-6.x 的服务器大多是 Windows Server 2003,运行的网站比较古老,开发语言一般为 Asp。由于版本限制的原因,该漏洞只能解析 asp 文件,而不能解析 aspx 文件^[2]。

该版本的解析漏洞有两个:

(1) 如果存在一个以*.asp 或*.asa 结尾的文件夹,则该文件夹中所有的文件都会被解析成 asp 文件。

例如,在网站目录下存在一个 folder.asp 的文件夹,然后上传一个 hacker.txt 到该文件夹内(假如 hacker.txt 文件中存在 Webshell),攻击者可以通过直接访问该文件在服务器上执行命令。

(2) IIS 解析文件名时从前往后解析,遇到分号自动停止。

假如存在一个名为 hacker.asp;.jpg 的图片,则它会被解析为一个 asp 文件,若该文件中存在 Webshell,同样可以被攻击者利用。

(注:asp 环境下除了 asp 还有三种后缀的文件:*.asa、*.cer、*.cdx,也是可执行的脚本文件)

2.2 Apache 解析漏洞

Apache 解析文件时,如果遇到不认识的文件扩展名,则会从后往前解析,直到遇到认识的扩展名为止。例如,上传一个名为“hacker.php.aa.xx”的文件,由于 Apache 不认识.xx 的扩展名,向前解析,又不认识.aa 的扩展名,再向前解析,就会将该文件解析为.php 文件。在 Apache 的安装目录下的/conf/mime.type 文件中,有详细的文件名列,它定义了 Apache 可以解析哪些扩展名的文件,如图 1 所示。

2.3 PHP CGI 解析漏洞

在动态环境为 php 的服务器中,通过从客户端访问服务器端文件构造恶意的 Url,也可以实现执行服务器上脚本文件的目的。假如在服务器中存在一个路径为 www.test.com/123.jpg 的图片,攻击者访问时在路径后面加上/hacker.php 变为 www.test.com/123.jpg/hacker.php,则服务器会将图片 123.jpg 当作 PHP 脚本解析。实际上 Url 最后的 hacker.php 是不存在的,所以路径中“hacker”可以是任意的。通过这个漏洞,攻击者将 Webshell 写进一张合法的图片制作成图片木

马,并上传至服务器。上传成功后在该图片的路径后加上/xxx.php 即可入侵成功^[3]。

因为 Nginx 服务器通常作为 php 的解析容器,而 Nginx 和 php 配合很容易导致该解析漏洞,所以尽管该漏洞与 Nginx 关系不大,也通常被认为是 Nginx 的解析漏洞^[4]。



图 1 Apache 可以解析的文件扩展名
(mime.types 文件)

3 针对不同防护手段的绕过技巧

3.1 针对客户端验证的绕过方法

客户端验证,就是在前端页面编写 JavaScript 代码来对用户上传文件的文件名进行合法性检测。这种验证原理是在载入上传的文件时使用 JavaScript 对文件名进行校验,如果文件名合法^[5],则允许载入,否则不允许。

但是此类验证非常容易被攻破。攻击者使用抓包软件拦截 HTTP 请求,并修改请求内容即可绕过。例如,先把木马文件改成 hacker.jpg 的合法文件,成功载入到等待上传区,然后点击发送上传请求。用 Burpsuite 将上传请求拦截后将文件名改回 hacker.php,再发送给服务器,则实际上传的文件后缀名为.php,实现了对客户端验证的绕过,如图 2 所示。

需要注意,如果 HTTP 请求修改前后的文件名长度发生了变化,那么在请求头中 Content-Length 的值也需要修改。Content-Length 代表实体正文长度。例如,如果修改之前文件名为 123.jpg,正文长度为 200,修改后文件名为 1.php,那么少了两个字符,Content-Length 就要改为 198,否则会上传失败。

由此看来,任何客户端验证都是不安全的。客户端验证是防止用户输入错误而进行的输入有效性检查,服务器端验证才可以真正防御攻击者。

```

Referer: http://127.0.0.1/register/
Connection: keep-alive
Content-Type: multipart/form-data; boundary=-----
Content-Length: 803

-----1815867388750600931284579519
Content-Disposition: form-data; name="userType"

学生
-----1815867388750600931284579519
Content-Disposition: form-data; name="userName"

123
-----1815867388750600931284579519
Content-Disposition: form-data; name="headpic"; filename="hacker.jpg"
Content-Type: application/x-php

Connection: keep-alive
Content-Type: multipart/form-data; boundary=-----
Content-Length: 803

-----1815867388750600931284579519
Content-Disposition: form-data; name="userType"

学生
-----1815867388750600931284579519
Content-Disposition: form-data; name="userName"

123
-----1815867388750600931284579519
Content-Disposition: form-data; name="headpic"; filename="hacker.php"
Content-Type: application/x-php

```

图 2 修改前后的 HTTP 请求

3.2 针对服务器端验证的绕过方法

3.2.1 绕过黑名单与白名单验证

(1) 黑名单过滤方式。

黑名单过滤是一种不安全的过滤方式。黑名单过滤在服务器端定义了一系列不允许上传的文件扩展名。在接收用户上传文件时,判断用户上传文件的扩展名与黑名单中的是否匹配。匹配则不允许上传,不匹配则允许上传。黑名单验证 PHP 代码如下:

```
<? php

$Blacklist = array( 'asp' , 'php' , 'jsp' , 'php5' , 'asa' ,
'aspx' ); //黑名单
.....

foreach ( $Blacklist as $key => $value) {

if( $value == $extension) { //判断是否为黑名单中的扩展名

    $boo = true;

break; //退出循环

}

}

if( ! $boo)

{ .....//允许上传}

else {

echo “文件不合法”;

}

.....

? >
```

在以上代码中,黑名单定义的危險的文件扩展名有 6 种,但实际上攻击者还是可以绕过黑名单检测^[6]。

(a) 攻击者可以找到 Web 开发人员忽略的扩展名, 如* .cer;

(b) 如果代码中没有对扩展名进行大小写转换操作,那就意味着可以上传如 AsP、pHp 这样扩展名的文件,而该类扩展名依然可以被 Windows 平台中的 Web

容器解析;

(c) 在 Windows 系统,可以上传如下文件名 “*.asp.”或“*.asp_(此处下划线为空格)”,在上传后,Windows 会自动去掉文件名后的点和空格。

(2) 白名单过滤方式。

相对于黑名单,白名单拥有更好的防御机制。白名单与黑名单相反,仅允许上传白名单中定义的扩展名。白名单验证 PHP 代码如下:

```
<? php
$ Whitelist = array( 'rar' , 'jpg' , 'png' , bmp, 'gif' ,
'doc' );
.....
foreach( Whitelist as $ key => $ value ) {
if( $ value == $ extension ) {
$ boo = true;
}
}
if( boo ) {
... //允许上传
}
else{
echo “文件不合法”;
}
? >
```

虽然白名单可以防御未知扩展名的上传,但是并不能完全防御上传漏洞。例如前文提到的 IIS5.x-6.x 解析漏洞,攻击者可以把文件改为 hacker.asp;.jpg 来绕过检测,并执行木马程序。

3.2.2 绕过 MIME 验证

在 HTTP 请求头中存在一个 Content-Type 字段^[7],它规定着文件的类型,即浏览器遇到此文件时使用相应的应用程序来打开。在上传时,服务器端一般会对上传文件的 MIME 类型进行验证,php 代码如下(以图片格式 jpg 为例):

```
if( $_FILES[ 'file' ][ 'type' ] == “image/jpeg” ) { //判断
是否是 jpg 格式
$ imageTempName = $_FILES[ 'file' ][ 'temp_name' ];
$ imageName = $_FILES[ 'file' ][ 'name' ];
$ last = substr( $ imageName , strpos( $ imageName , “.” ) );
if( ! is_dir( “uploadFile” ) ) {
mkdir( “uploadFile” );
}
$ imageName = md5( $ imageName ) . $ last;
Move _ upload _ file ( $ imageTempName , “./uploadFile/”.
$ imageName ); //指定上传文件到 uploadFile 目录
echo( “文件上传成功” );
} else {
echo( “文件类型错误,上传失败” );
exit;
```

}

但是,MIME 验证也可以被中间人攻击^[8]。攻击者用抓包软件拦截到上传文件的请求时,可以看到上传文件的部分的 Content-Type 是 application/x-php 类型,这样上传文件对于有 MIME 类型验证的服务器肯定是上传不了的。将请求这里的值改为 image/jpeg,即可成功绕过该验证上传文件,如图 3 所示。



图 3 修改前后的 HTTP 请求

3.2.3 绕过目录验证

用户上传文件时,Web 程序一般是允许用户上传文件到一个指定的文件夹。不过有时 Web 开发人员为了方便,通常会做这样一个操作:如果用户上传的目录存在,则将用户的文件存入该文件夹;如果目录不存在,则创建该目录,并写入文件^[9]。示例 PHP 代码如下:

```
//HTML 代码
<form action="upload.php" method="post" enctype="mul-
tipart/form-data">
<input type="file" name="file"/><br/>
<input type="hidden" name="Extension" value="up"/>
<input type="submit" value="提交" name="submit"/>
</form>

//PHP 代码
if( $_FILES[ 'file' ][ 'type' ] == “image/jpeg” )
$ imageTempName = $_FILES[ 'file' ][ 'temp_name' ];
$ imageName = $_FILES[ 'file' ][ 'name' ];
$ last = substr ( $ imageName , strpos ( $ imageName ,
“.” ) ); //判断图片类型是否符合
if( $ last! = “.jpg” ) {
Exit( “图片类型错误” );
}
$ Extension = $_POST[ 'Extension' ]; //获取文件上传目录
if( ! is_dir( Extension ) ) {
mkdir( $ Extension );
}
$ imageName = md5( $ imageName ) . $ last;
move _ upload _ file ( $ imageTempName , “./uploadFile/”.
$ imageName );
echo( “文件上传成功” );
exit( );
```

```

}
//Upload.php 中有如下代码
if(! is_dir( $Extension)) {
mkdir ( $Extension);
}

```

在 upload.php 中的这一段代码是引发漏洞的关键点。在 html 中有一个隐藏的标签 `<input type="hidden" name="Extension" value="up" />`, 这个标签标示的是上传文件时默认的文件夹, 而同样该标签的参数对攻击者来说是可控的。依然可以用 burp 对请求进行拦截, 对该标签的参数进行修改。例如前文提到的 Web 容器 IIS6.0, 将新建的文件夹的名称命名为 123.asp 并在其中包含木马文件, 则可直接获得 web-shell^[10-11]。

3.2.4 截断上传攻击

截断上传攻击的核心是“%00”字符, 也被称为 Url 终止符。通俗的讲就是如果 Url 中包含这个字符, 那么这个字符之后的所有内容都会被丢弃。同样在 HTTP 请求中也适用^[12]。

```

//test.asp
<%
Username=request( "username")
Response.write username
%>

```

以上代码的作用是接收 username 的值并输出。访问 Url: HTTP://www.test.com: 8080/test.asp? username=hacker%00admin, 可以发现, 输出的内容只有“hacker”, %00 后面的字符都被截断了, 这就是截断攻击的原型^[13]。

例如, 如下网页有一个文件上传模块, 先上传一张普通图片, 得到提示如果要得到 flag, 必须要上传 PHP 文件才可以。然后再上传一个 1.php 文件, 提示*.php 是不被允许的文件类型, 仅支持上传 jpg gif png 后缀的文件。

针对这种情况可以使用截断攻击。先上传一个 1.php.jpg 的文件(此处的下划线为空格, 方便修改请求), 在上传时将请求拦截, 用 Burpsuite 的 hex 视图在上传路径的 1.php 后的%20(空格)改成 Url 终止符%00 再上传, 提示获取 flag 成功, 如图 4 和图 5 所示。

尽管上传的是一个*.php 文件, 但是如果不进行%00 截断, 上传的文件在服务器上以<1.php.jpg>格式保存, 也就是说这是一个图片文件, PHP 无法解析这个文件。当进行%00 截断后, 服务器就会将%00 后的<.jpg>丢弃, 这时文件将以<1.php>的形式保存在服务器上, 恶意脚本也就成功上传了。

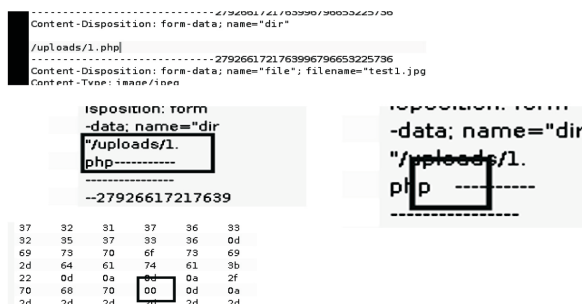


图4 修改 HTTP 请求



图5 获得 flag

4 防御措施

通过分析, 文件上传漏洞形成的主要原因有两点:

(1) 上传文件过滤不严, 攻击者可能直接上传恶意脚本;

(2) 攻击者可能通过上传“合法文件”, 利用 Web 解析漏洞使之能够被执行^[14]。

文件上传漏洞的防御, 主要围绕以下几点: 文件上传路径; 文件访问权限; 文件执行权限。另外, 由于业务领域不同, 根据上传文件类型的不同也需要进行不同的防御。以下提出了防范文件上传漏洞的几种常见方法:

(a) 文件上传的目录设置为不可执行。只要 Web 容器无法解析目录下面的文件, 即使攻击者直接上传了 webshell, 服务器本身也不会受到影响, 这一点至关重要。

(b) 判断文件类型。即前文提到的 MIME 验证, 在文件类型检查中, 强烈推荐白名单方式, 黑名单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脚本代码。

(c) 使用随机数改写文件名和文件路径攻击者如果要执行上传的文件, 则需要能够访问到这个文件。在某些环境中, 攻击者能上传, 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。像 shell.php.rar.rar 和 crossdomain.xml 这种文件, 都将因为重命名而无法攻击^[15]。

(d) 单独设置文件服务器的域名。由于浏览器同源策略的关系, 一系列客户端攻击将失效: 上传 cross-domain.xml、上传包含 JavaScript 的 XSS 利用等问题将

得到解决。

5 结束语

在服务器工作环境中,Shell 指的是“为用户提供使用界面”的软件,类似于 Windows 系统中的 cmd.exe,网络管理员可以通过 Shell 对服务器进行一些管理行为的操作。而对于攻击者来说,文件上传漏洞一直都是获取服务器 Shell 的重要途径,对系统维护人员来说,文件上传漏洞的巨大危害,可以直接导致操作系统对攻击者暴露且被控制,并通过操作系统横向渗透入网络内部的其他设备。文中给出的方法基本可以解决上传漏洞,但不能说完全防御,因为没有绝对的安全,攻击者总是可以从不同角度对服务器进行攻击。所以,Web 开发人员不仅要在代码层面对 Web 程序做好安全防护工作,更要对服务器操作系统、网络通信层面进行保护和严密的监控。及时更新服务器操作系统和 WAF 的补丁^[16],对网络设备的日志进行及时、全面的审计,对用户的非法行为进行记录并做相应的处理^[17]。不能只在某一个方面做好防护,安全是一个整体。与此同时,安全人员需要注意文件上传漏洞和其他 Web 漏洞配合所做出的“组合攻击”,这将成为今后 Web 安全研究的重点。

参考文献:

- [1] 诸葛建伟,陈立波,孙松柏,等. Metasploit 渗透测试魔鬼训练营[M]. 北京:机械工业出版社,2013:2-31.
- [2] 张炳帅.Web 安全深度剖析[M]. 北京:电子工业出版社,2015:106-128.
- [3] 池阳,高健,周福才.基于 ISAPI 过滤器的 Web 防护系统[J].信息安全,2014(7):35-40.
- [4] 韦鲲鹏,葛志辉,杨波.PHP Web 应用程序上传漏洞的攻防研究[J].信息安全,2015(10):53-60.
- [5] 王威.PHP 网站安全性的分析研究及其在图片上传系统中的应用[D].北京:北京邮电大学,2011.
- [6] 白兴瑞,刘耀炎.高校 WEB 站点的上传漏洞分析及防范[J].衡水学院学报,2011,13(4):34-36.
- [7] AMITY H, SALTZMAN R. Testing web applications for file upload vulnerabilities: U.S. 9009841 [P]. 2015-04-14.
- [8] YAO Chunlong, YIN Fengjiao, LI Xu, et al. Security analysis of PHP encoder[J]. Journal of Networks, 2013, 8(10): 2353-2360.
- [9] 刘鹏.PHP Web 应用程序安全性研究及安全漏洞检测工具开发[D].西安:西安电子科技大学,2012.
- [10] 石刘洋,方勇.基于 Web 日志的 Webshell 检测方法研究[J].信息安全研究,2016,2(1):66-73.
- [11] XU Mingkun, CHEN Xi, HU Yan. Design of software to search ASP Web Shell [J]. Procedia Engineering, 2012, 29: 123-127.
- [12] SCHECHTER S, KRISHNAN M, SMITH M D. Using path profiles to predict HTTP requests [J]. Computer Networks and ISDN Systems, 1998, 30(1-7): 457-467.
- [13] 邱小果.编程实现基于 Cookie 验证的 HTTP 请求的发送[J].微型机与应用,2003,22(7):35-37.
- [14] 龙金辉,王坤杰.PHP 网站文件上传的研究[J].电脑编程技巧与维护,2014(14):85.
- [15] 李万彪.ASP.NET 2.0 中文件上传安全解决方案[J].电脑知识与技术,2009,5(8):1827-1828.
- [16] 马艳发.基于 WAF 入侵检测和变异 WebShell 检测算法的 Web 安全研究[D].天津:天津理工大学,2016.
- [17] 公安部信息安全等级保护评估中心.信息安全等级测评师培训教程[M].北京:电子工业出版社,2010:42-107.