完整性检验工具 Aide

V0.1

Coolc

2006-1-26

前言

Coolc 突然对完整性检验工具有了一些兴趣，于是索性自己做个简单教程，让大家以后自己鼓弄时也有个参考，也可以方便一下想通过完整性校验工具，而让自己系统更安全的朋友。

软件简介

AIDE，英文(Advanced Intrusion Detection Environment)直译为高级入侵检测环境。

AIDE，是一个文件完整性检测工具，AIDE 能够构造一个指定文件的数据库，它使用 aide.conf 作为其配置文件。AIDE 生成的数据库能够保存文件的各种属性，包括：权限(permission)、索引节点序号(inode number)、所属用户(user)、所属用户组(group)、文件大小、最后修改时间(mtime)、创建时间(ctime)、最后访问时间(atime)、增加的大小以及连接数。AIDE 还能够使用下列算法：sha1、md5、rmd160、tiger，以密文形式建立每个文件的校验码或散列。

用途

管理员在系统安装完毕，连接到网络上之前，可以通过该程序建立新系统的 AIDE 数据库。这个 AIDE 数据库是系统的一个快照和以后系统升级的准绳。数据库应该至少包含这些信息：关键的系统二进制可执行程序、动态连接库、头文件以及其它总是保持不变的文件。（当然也可以用一些变通的策略，例如/dev 下很多终端设备只是 permisson 变动，所以只要检查时去掉权限检查，就不会被报警淹没。）

一旦发现系统被侵入，系统管理员会使用 ls、lsof、ps、netstat、last 以及 who 等系统工具对系统进行检查，但是所有这些系统工具都可能被 rootkit 程序代替了。可以想象被修改的 ls 程序、ps 也不会显示任何入侵进程的信息，甚至本身就是一个肩负 backdoor 任务的程序。即使系统管理员恐怕永远也无法通过简单的文件属性来获知它们是否被修改过了，因为文件日期、大小等信息是非常容易改变的，如利用 touch。

但是如果有了 aide，那一切就都简单了。系统管理员只要运行 AIDE，就能够很快识别出哪些关键文件被攻击者修改过了。本文的附录中 coolc 给出了个例子，呵呵。

不过，要注意这也不是绝对的，因为 AIDE 可执行程序数据库自保护不好，

二进制文件本身可能被修改了或者数据库也被修改了。因此，应该把 AIDE 的数据库放到安全的地方，而且进行检查时要使用保证没有被修改过的程序，当然自己写一些增强程序就更好了 :)。下面 coolc 就介绍一下 aide 具体的安装和配置方法。

内容

安装

coolc 是在 slackware 上安装的，其他类型 linux 安装方法类似。
1、 **下载和解压缩最新的** aide **和** libmhash **软件包。**
建议从官方网站获得可靠的 aide **和** libmhash **软件包。**

通过 md5 **校验后，进行安装。**
http://sourceforge.net/projects/aide/
http://freshmeat.net/projects/mhash/

1) **创建** aide **软件包存放的目录。**
#mkdir -p /usr/local/src/harden

2) **解压缩源代码包， 在** id **目录下会生成两个新的目录** aide-0.10 **和** mhash-0.9.4
#tar zpxf aide-0.10.tar.gz
#tar zpxf mhash-0.9.4.tar.gz

2。 **配置预编译环境**
在 mhash-0.9.4 **和** aide-0.10 **目录依次分别做如下的操作**:

#cd mhash-0.9.4
#./configure &&make &&make install
#cp include/mutils/*.h /usr/local/include/mutils/

#cd aide-0.11
#./configure --prefix=/usr/local &&make &&make install

注意
在 slackware8.1 **下,由于默认编译器版本的原因，需要将** mhash **解压目录下** lib **子目录中的文件** stdfns.c **第** 128 **行进行修改。**

修改前

```
#define MIX32(a) \
        (((mutils_word32)((mutils_word8 *)(a))[0]) | \
        (((mutils_word32)((mutils_word8 *)(a))[1]) << 8)| \
        (((mutils_word32)((mutils_word8 *)(a))[2]) << 16)| \
        (((mutils_word32)((mutils_word8 *)(a))[3]) << 24))
```

改成

```
#define MIX32(a) (((mutils_word32)((mutils_word8 *)(a))[0]) |(((mutils_word32)((mutils_word8 *)(a))[1])
<< 8)|(((mutils_word32)((mutils_word8*)(a))[2]) << 16)|(((mutils_word32)((mutils_word8 *)(a))[3]) << 24))
```

二。 安装

安装的过程很简单， 依次运行:

```
#cd mhash-0.9.4
#make;make install
#cd aide-0.10
#make;make install
```

配置

接下来的步骤是配置 aide.conf。 aide.conf **配置文件的格式是非常简单的。 在设置该文件之前， 建议阅读该配置。 或者阅读** man **帮助文件**:

注意:

1、拷贝默认配置模板到目录。

```
#cp doc/aide.conf /usr/local/etc/
```

配置文件的缺省位置是: /usr/local/etc/aide.conf

2、初始化 checksum 数据库。

```
root@coolc:~/aide-0.11-rc2# aide -i

AIDE, version 0.9.4

### AIDE database at aide.db.new initialized.
```

将数据库拷贝到指定的位置

root@coolc:/usr/local/bin# cp ./aide.db.new /usr/local/etc/aide.db

下面是一个简短的 aide.conf **范例，建议此处可以多用心调配，说个小技巧，你可以用** tripwire **的模版来自己改 ，呵呵。**

@@define TOPDIR /usr/local/aide

#定义宏变量 TOPDIR，您可以随意定义自己喜欢的宏。

###

@@ifndef TOPDIR

@@define TOPDIR /

@@endif

###

#aide 里的宏可以使用类似 c 里的控制语句 undef 、else、 ifdef 来控制宏是否生效。

@@ifdef DEBUG

@@define DEBUG ison

@@undef NOT_DEBUG

@@else

@@define NOT_DEBUG true

@@undef DEBUG

@@endif

database=file:@@{TOPDIR}/etc/aide.db

＃存放数据库的位置

database_out=file:aide.db.new

＃生成新数据库时的位置

verbose=20

＃输出模式分为 20 个级（1－20），20 为最详细的输出

report_url=stdout

＃输出报告的模式。此处我定义的为标准输出，您还可以定义为 syslog，mail 等模式

#常见规则

#Device =p+u+g+s+i

Device =u+g+s+i

```
Dynamic           =  p+i+n+u+g+sha1
Growing           =  p+u+g+i+n+S
IgnoreNone        =  p+i+n+u+g+s+m+a+c+S+md5+sha1+rmd160+tiger
ReadOnly          =  p+i+u+g+md5
Temporary         =  p+u+g+i
Rule = p+i+u+g+n+s+md5

/dev/.* Device

   ##########################################
  #                                          ##
##########################################  #
#                                           # #
#    Aide Binaries and Data Files           # #
#                                           ##
##########################################
/your/aide/path



   ################################################
  #                                                ##
################################################  #
#                                                 # #
#   OS  Binaries  and  Libraries                  # #
#                                                  ##
################################################

   /bin/.*    ReadOnly
   /lib/.*    ReadOnly
   /sbin/.*    ReadOnly

   /usr/local/sbin/.*    ReadOnly
   /usr/local/bin/.*    ReadOnly

   /usr/bin/.*    ReadOnly
   /usr/sbin/.*    ReadOnly
   /usr/lib/.*    ReadOnly
   /usr/libexec/.*    ReadOnly

   /usr/X11R6/bin/.*    ReadOnly
   /usr/X11R6/lib/.*    ReadOnly

   /usr/games/.*    ReadOnly
```

/opt/www/htdig/bin/.* ReadOnly

注意就如 coolc **前面提到的，**linux **下很多文件变化频繁，比如** ssh **下的** radom_seed **文件，可以通过如下写法屏蔽，如：**

!/etc/ssh2/random_seed

当然这也给黑客留了空子，这就需要管理根据系统特性灵活处理。比如只观测 inode **和属主等属性的变化，或者自身容忍少量的抱错，呵呵。个人觉得第一要点是配置文件不要放在服务器上，比如** coolc **就喜欢放置一个错误的配置文件，用以误导黑客：）** **。**

初始化

提示下面成功信息:

AIDE， version 0.10

AIDE database initialized。

修改 aide.conf **配置文件**:

#vi /usr/local/src/id/aide-0.10/doc/aide.conf

修改成下面的配置:

database=/usr/local/aide/etc/aide.db
database_out=/usr/local/aide/etc/aide.db.new
database_new=/usr/local/aide/etc/aide.db.new

维护

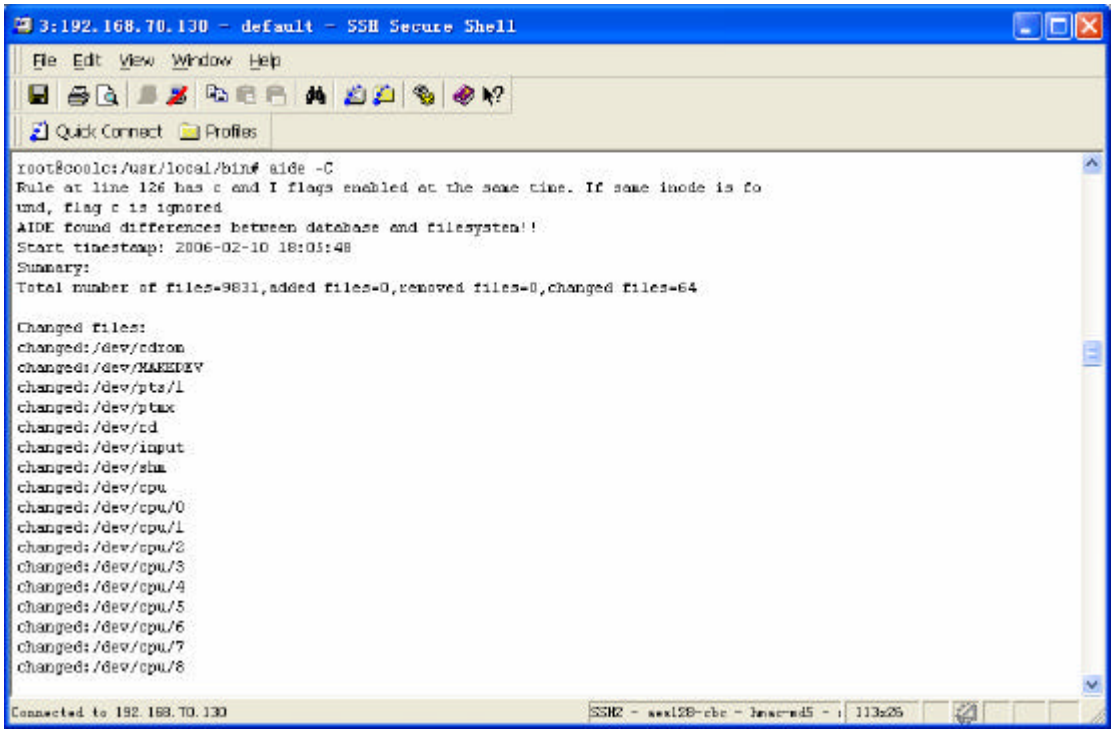
Checksum 检查

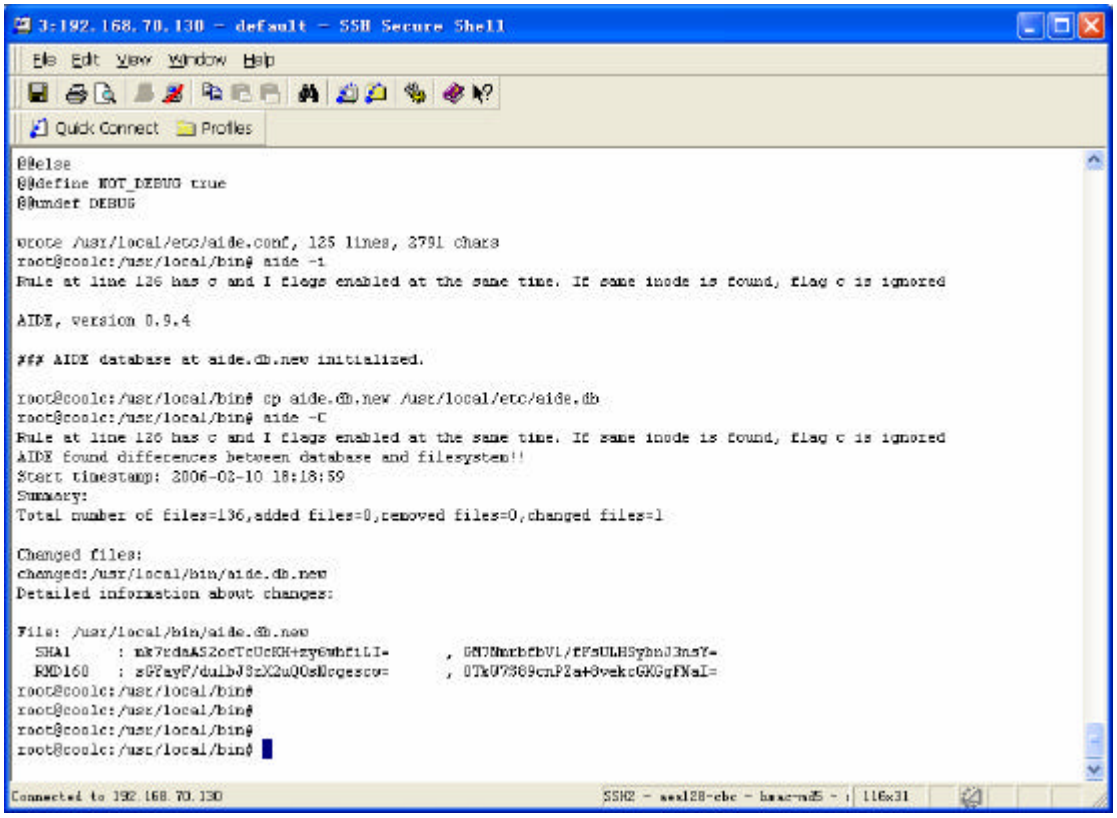
需要 checksum **检查数据库， 运行**:
root@coolc:/usr/local/bin# aide -C

AIDE, version 0.9.4

All files match AIDE database. Looks okay!

aide **然后会向你报告任何的变化，如下所示。**





或者使用下面的命令仔细的比较最初的的数据库和后生成的库之间的区别。
现在的数据库:
#aide --compare

升级 checksum 数据库

如果你已经完成了检查和修复任务， 你需要重新更新一下数据库:
#aide --update

性能影响

为了保证软件不会影响正常服务，coolc 对本软件的性能进行了简单的测试。从结果看对系统 MEM 、CPU、IO 都有影响。总体上看影响如下

主机指标

DELL PE2850

至强 CPU 4CPU
内存 2G
200G 硬盘

采用的策略

采用 coolc 自己调配的配置模板，检测文件共 28329 个。

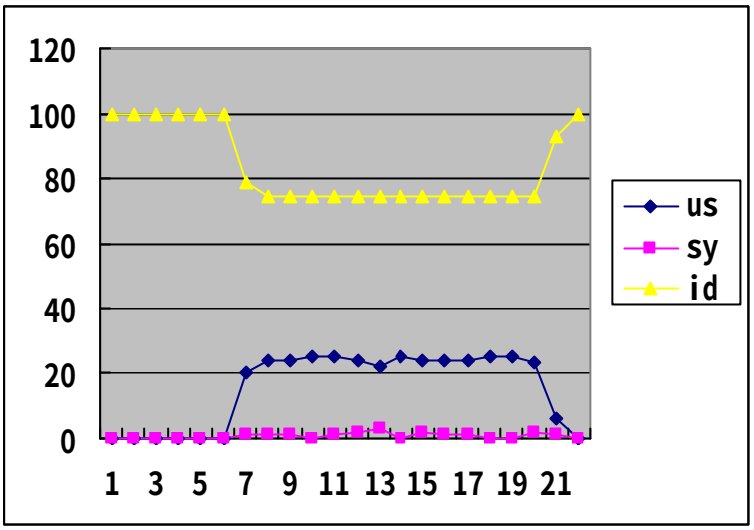
评测性能

主要关注规则误报率、CPU 负载、IO 负载、Mem 负载此几项指标。通过 vmstat 取值计算。

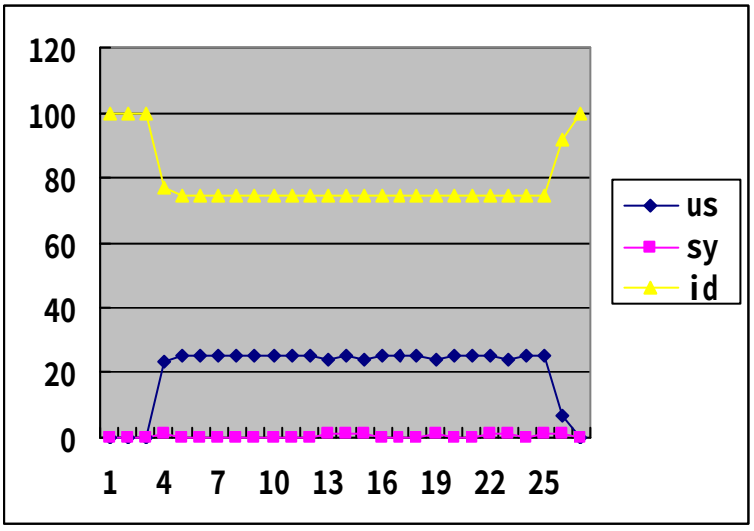
结果如图

CPU 负载

初始化时

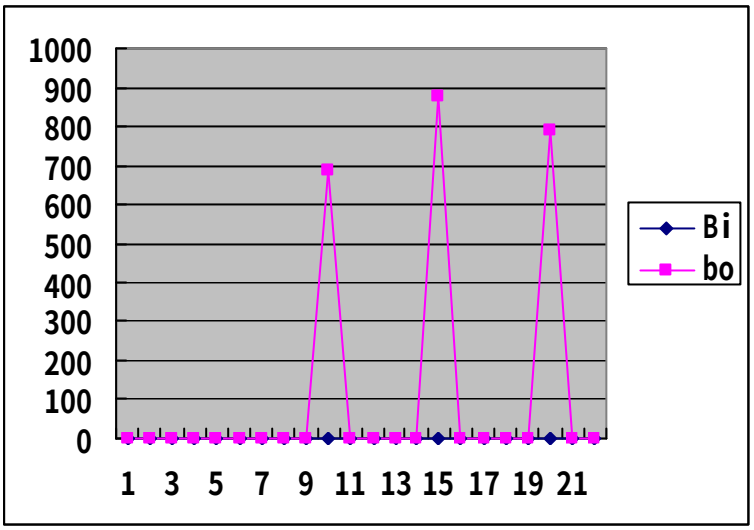


检查时化时

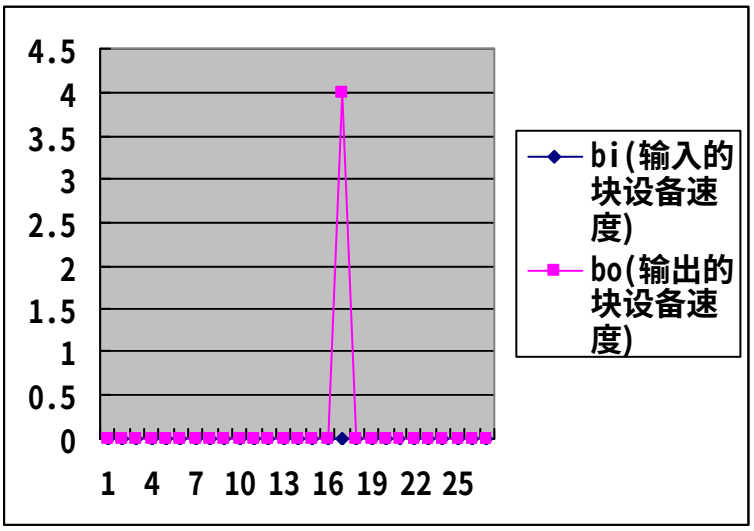


IO 负载

Aide 初始化时

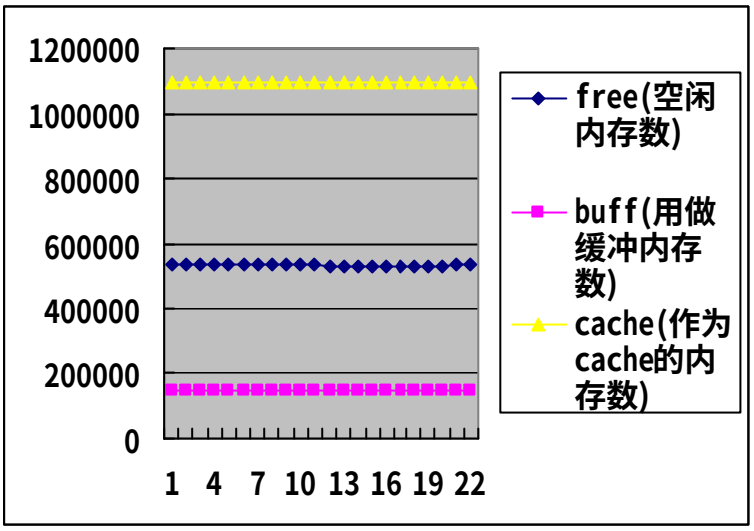


Aide 进行检查时 峰值仅为 4

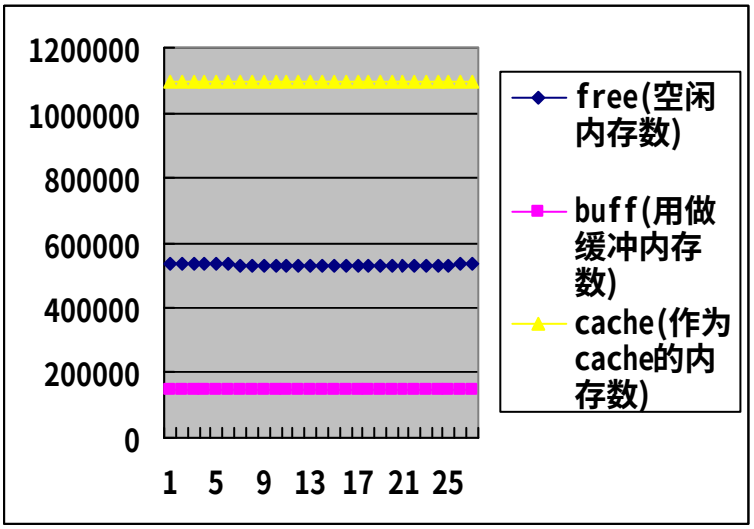


Mem 负载

Aide 初始化时



Aide 检查时



总结

性能：从目前测试情况看本程序对 CPU 资源有占用此处需要优化（一下吃掉了我一个 CPU），对于内存的资源挤占较少，可以忽略影响。对于 IO 的影响，还有优化空间，不适合在高 IO 负载的主机上运行。

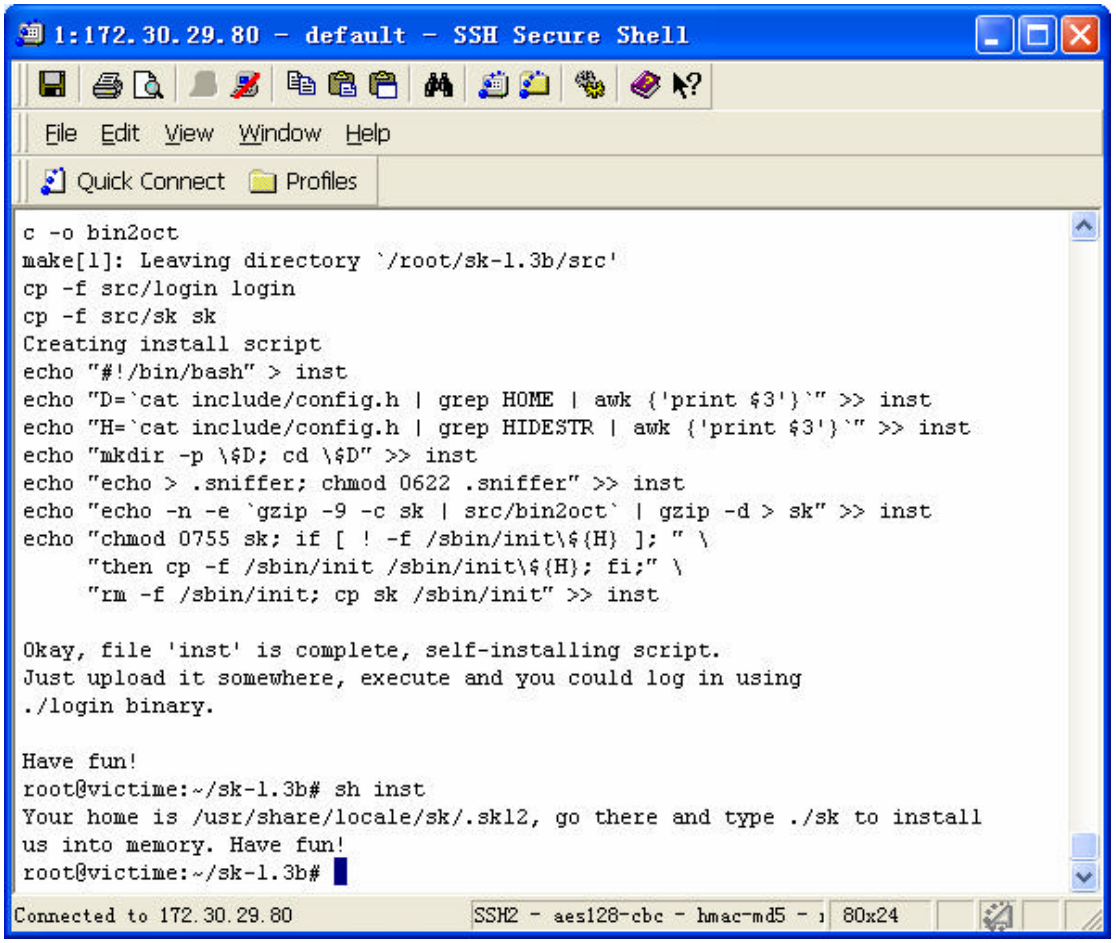
速度：整个检查 2w 多个文件，一般 20 秒内可以完成，coolc 相当满意，呵呵。

弱点：自身保护太差了，这也是他和 tripwire 相比一大弱点，所以你的程序、配置文件、aide 的 db 千万不要放在主机上，如果人家随便帮咱们 update 一下，咱们可就白忙活了。当然也有变通的方法，比如写个小东东定期检测 aide.db 等的 checksum。

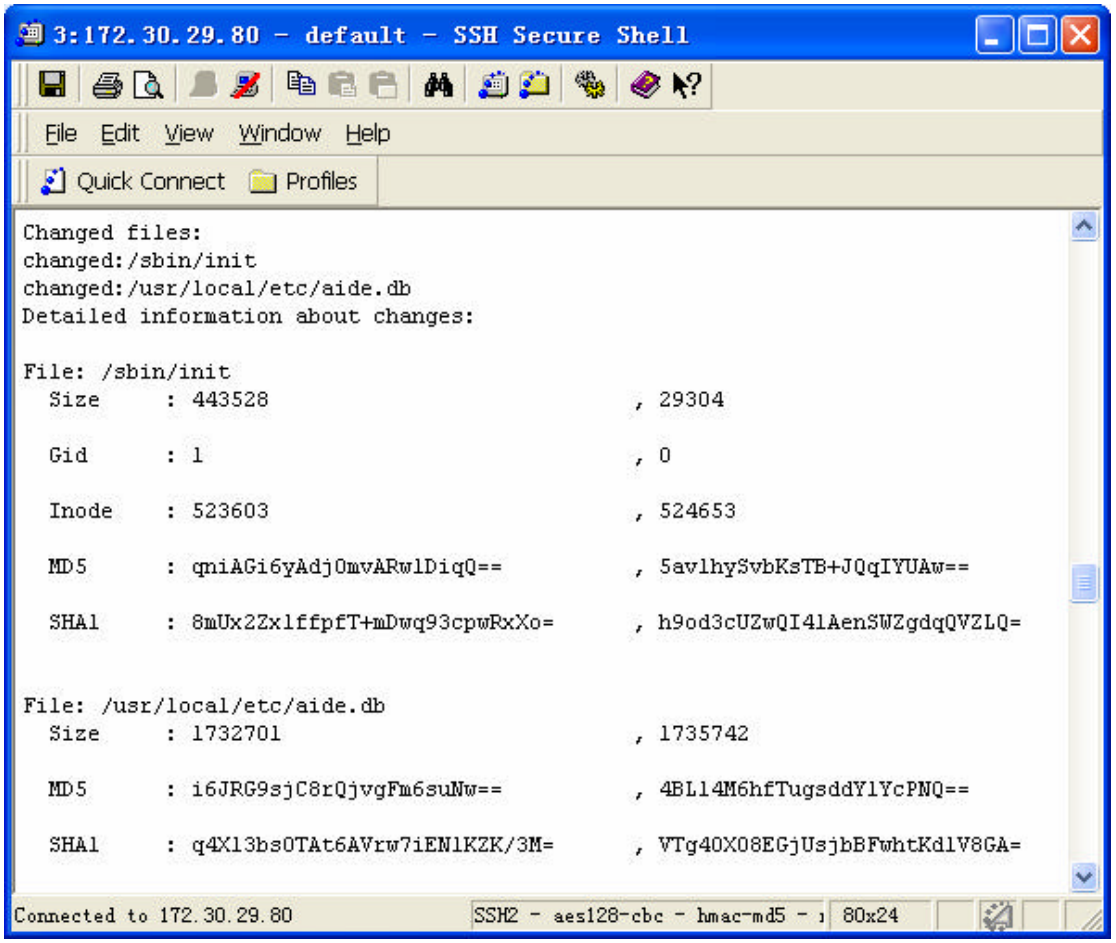
附录

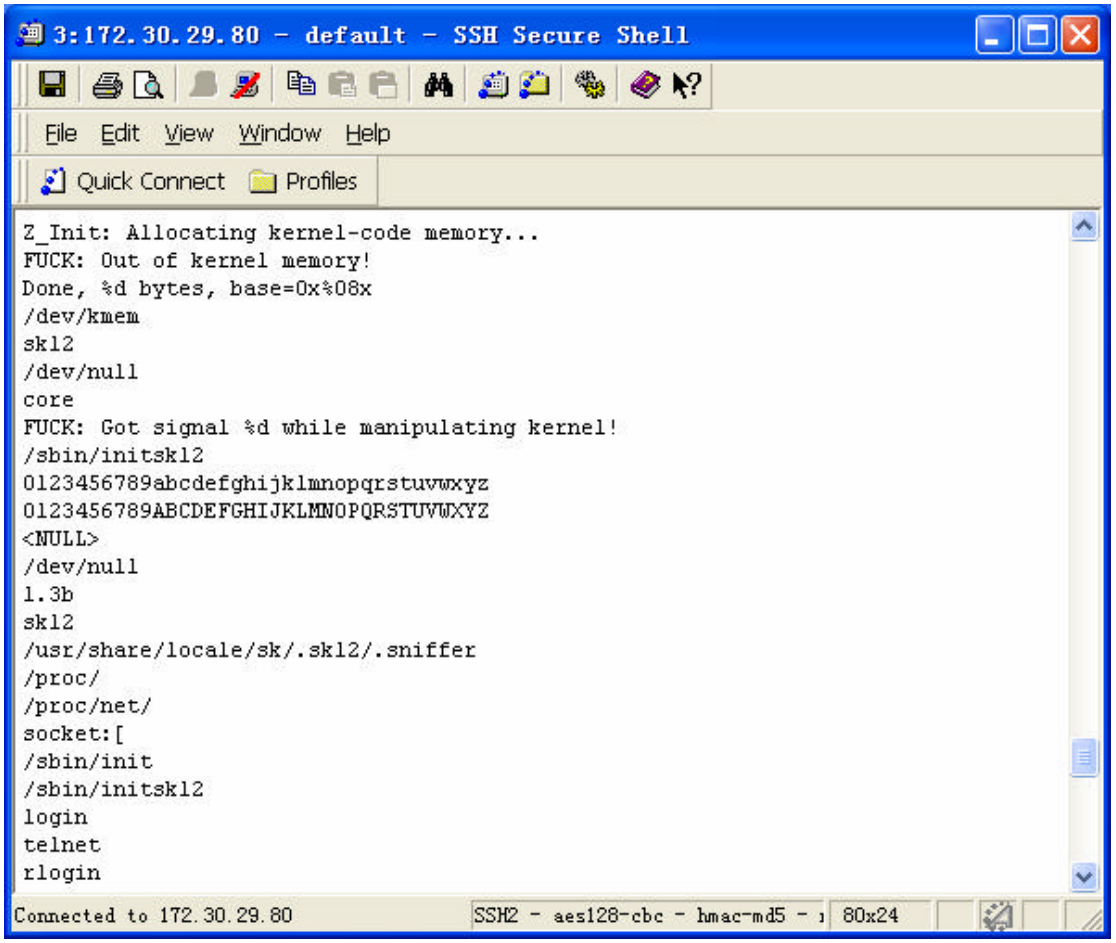
对 sk-1.3b 的测试

为了验证软件的有效性，coolc 作了个简单的模拟测试，先个自己的机器装一个 sk 后门。



然后通过运行 aide **很方便的就发现了异常行为。在** sbin **上给我添加了**/sbin/initsk12 **并且改动了** init**，然后** strings **一下，呵呵，又有了很多发现。后门的事情就比较好办了，可见这个小东东还是满不错好用的，呵呵。**


3:172.30.29.80 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
Changed files:
changed:/sbin/init
changed:/usr/local/etc/aide.db
Detailed information about changes:
File: /sbin/init
Size : 443528 , 29304
Gid : 1 , 0
Inode : 523603 , 524653
MD5 : qniAGi6yAdjOmvARwlDiqQ== , 5avlhySvbKsTB+JQqIYUAw==
SHA1 : 8mUx2ZxlffpfT+mDwq93cpwRxXo= , h9od3cUZwQI4lAenSWZgdqQVZLQ=
File: /usr/local/etc/aide.db
Size : 1732701 , 1735742
MD5 : i6JRG9sjC8rQjvgFm6suNw== , 4BL14M6hfTugsddYlYcPNQ==
SHA1 : q4Xl3bs0TAt6AVrw7iENlKZK/3M= , VTg40X08EGjUsjbBFwhtKdlV8GA=
Connected to 172.30.29.80
SSH2 - aes128-cbc - hmac-md5 -
80x24


3:172.30.29.80 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
Z_Init: Allocating kernel-code memory...
FUCK: Out of kernel memory!
Done, %d bytes, base=0x%08x
/dev/kmem
sk12
/dev/null
core
FUCK: Got signal %d while manipulating kernel!
/sbin/initsk12
0123456789abcdefghijklmnopqrstuvwxyz
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
<NULL>
/dev/null
1.3b
sk12
/usr/share/locale/sk/.sk12/.sniffer
/proc/
/proc/net/
socket:[
/sbin/init
/sbin/initsk12
login
telnet
rlogin
Connected to 172.30.29.80
SSH2 - aes128-cbc - hmac-md5 -
80x24