

## 1、SSHD服务

介绍：SSH协议：安全外壳协议。为 Secure Shell的缩写。SSH为建立在应用层和传输层基础上的安全协议。

作用：在客户端与远程服务器主机数据传输前，SSH服务会对需要传输的数据进行加密，从而保证客户端与远程主机的会话中数据安全

相比较之前用Telnet方式来传输文件要安全很多，因为Telnet使用明文传输，SSH是加密传输。

SSH服务由服务端（openssh）与客户端（如xshell、secureCRT）两部分组成，SSH服务一般装机自带安装。

查看镜像中的openssh开头的程序包：

```
[root@base Packages]# ls /mnt/Packages/openssh
/mnt/Packages/openssh-7.4p1-11.el7.x86_64.rpm
/mnt/Packages/openssh-keycat-7.4p1-11.el7.x86_64.rpm
/mnt/Packages/openssh-askpass-7.4p1-11.el7.x86_64.rpm
/mnt/Packages/openssh-server-7.4p1-11.el7.x86_64.rpm
/mnt/Packages/openssh-clients-7.4p1-11.el7.x86_64.rpm
```

openssh-7.4p1-11.el7.x86\_64.rpm：包含OpenSSH服务器及客户端需要的核心文件。

openssh-clients-7.4p1-11.el7.x86\_64.rpm：OpenSSH客户端软件包。

openssh-server-7.4p1-11.el7.x86\_64.rpm：OpenSSH服务器软件包。

openssh-askpass-7.4p1-11.el7.x86\_64.rpm：支持对话框窗口的显示，是一个基于X系统的密码诊断工具

查看版本

```
[root@home ~]# ssh -V
```

```
OpenSSH_7.4p1, OpenSSL 1.0.2k-fips 26 Jan 2017
```

## 2、安装SSH服务

安装方法有两种：

1. 通过Yum安装（推荐使用）：

```
[root@base ~]# yum install openssh openssh-clients openssh-server -y
```

2. 本地直接安装rpm包文件：

```
[root@base ~]# rpm -ivh /mnt/Packages/openssh*.rpm
```

以上选择一种安装方式即可。

3. 确认软件包是否已经安装：

```
[root@home ~]# rpm -qa |grep openssh
openssh-server-7.4p1-11.el7.x86_64
openssh-7.4p1-11.el7.x86_64
openssh-clients-7.4p1-11.el7.x86_64
```

4. 查看软件安装生产的文件：

```
[root@base ~]# rpm -ql openssh
/etc/ssh
/etc/ssh/moduli
/usr/bin/ssh-keygen
/usr/libexec/openssh
/usr/libexec/openssh/ctr-cavstest
/usr/libexec/openssh/ssh-keysign
.....
```

5. OpenSSH配置文件：

OpenSSH常用配置文件有两个/etc/ssh/ssh\_config  
和/etc/ssh/sshd\_config。

ssh\_config为客户端配置文件，设置与客户端相关的应用可通过此文件实现。

sshd\_config为服务器端配置文件，设置与服务端相关的应用可通过此文件实现。

### 3、如何使用SSH来远程连接主机

语法：

```
ssh [远程主机用户名] @[远程服务器主机名或IP地址] -p port
```

-p: -p选项，指定登录端口（当服务端的端口非默认时，需要使用-p指定端口进行登录）。

当在Linux主机上远程连接另一台Linux主机时，如当前所登录的用户是root的话，当连接另一台主机时也是用root用户登录时，可以直接使用ssh IP，端口默认即可，如果端口不是默认的情况下，需要使用-p 指定端口。

```
[root@base ssh]# ssh 192.168.1.64
```

```
The authenticity of host '192.168.1.64 (192.168.1.64)' can't be
established.
```

```
ECDSA key fingerprint is
```

```
SHA256:QFj9fwEJausEeOSvUFlG53KxfwdlzLzzz3M/HAobEhY.
```

```
ECDSA key fingerprint is
```

```
MD5:e3:a3:29:b6:04:1d:2e:23:b6:bc:08:87:0f:dd:d6:d2.
```

```
Are you sure you want to continue connecting (yes/no)? yes
```

第一次登录服务器时系统没有保存远程主机的信息，为了确认该主机身份会提示用户是否继续连

接，输入yes 后登录，这时系统会将远程服务器信息写入用户主目录下的\$HOME/.ssh/known\_hosts文件中，下次再进行登录时因为保存有该主机信息就不会再提示了，如图 1-5 所示。

```
[root@base2 ~]# exit
```

登出

```
Connection to 192.168.1.64 closed.
```

```
[root@base ~]# cat /root/.ssh/known_hosts
```

```
192.168.1.64 ecdsa-sha2-nistp256
```

```
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBNo
Wrh4bU6qDM0V3CVPnfyEhgdyCOJwKTmgMqIt0sTM0bjQd3ryJyq34HfP
lBse222KwrZ9S5tvheXEcU/PEk2Y=
```

图 1-5 密钥信息

RSA算法基于一个十分简单的数论事实：将两个大素数相乘十分容易，但是想要对其乘积进行因式分解却极其困难，因此可以将乘积公开作为加密密钥。

使用普通用户登录：

创建用户并设置密码为123456。

```
[root@base ~]# ssh 192.168.1.64
```

```
[root@base2 ~]# useradd kill && echo 123456 | passwd --stdin kill
```

更改用户 kill 的密码。

passwd：所有的身份验证令牌已经成功更新。

```
[root@base2 ~]# exit
```

```
[root@base ~]# ssh kill@base2    #远程主机名
```

ssh: Could not resolve hostname base2: Name or service not known

#需要配置hosts文件

```
[root@base ~]# cat /etc/hosts
```

```
127.0.0.1 localhost localhost.localdomain localhost4
```

```
localhost4.localhost4
```

```
::1 localhost localhost.localdomain localhost6
```

```
localhost6.localhost6
```

```
192.168.1.64 base2
```

## 4.支持远程执行命令

```
[root@home ~]# ssh root@192.168.1.64 ls /tmp
```

root@192.168.1.64's password:

a

```
systemd-private-dbff0e7c8f74450f92a752e3eaa46ea7-
```

```
chronyd.service-BjJaEQ
```

```
systemd-private-dbff0e7c8f74450f92a752e3eaa46ea7-
```

```
vgauthd.service-nsMonx
```

```
systemd-private-dbff0e7c8f74450f92a752e3eaa46ea7-
```

```
vmtoolsd.service-dmEe7d
```

```
[root@home opt]# ssh root@home2 "for i in {1..10};do touch b$i
```

```
;done"
```

root@home2's password:

## 5、SSH附带命令：scp远程复制命令

语法：

scp 参数 文件名 用户@远程主机名或IP地址： 目标目录

参数：

-p:复制时保留文件或目录属性

-r:复制目录

```
[root@base ~]# scp etc.tar root@192.168.1.64:/opt
```

## 6、SSH附带服务：sftp服务

上传命令 put

```
sftp> put file.zip
```

```
sftp> put file.zip /opt
```

下载命令 get

```
sftp> get a.sh
```

```
sftp> get a.sh /opt
```

### 1.创建用户组 sftp

```
[root@base2 ~]#groupadd sftp
```

### 2.创建用户

```
[root@base2 ~]#useradd -g sftp -s /sbin/nologin abc
```

-g 加入到sftp组

-s 禁止登录

### 3.设置密码

```
[root@base2 ~]#passwd abc
```

输入两次密码

### 4.修改服务端ssh的配置文件

```
[root@home ~]# vim /etc/ssh/sshd_config
```

修改sshd\_config如下:

#### 4.1.注释原来的Subsystem设置

```
132 #Subsystem sftp /usr/libexec/openssh/sftp-server
```

#### 4.2.添加一行, 启用internal-sftp

```
133 Subsystem sftp internal-sftp
```

#### 4.3.限制用户SFTP访问的根目录

以abc用户为例

```
134 Match User abc
```

```
ChrootDirectory /opt/abc
```

```
ForceCommand internal-sftp
```

注: /home/abc此时的文件所有者和属组需要是root, 不然会报错

#### 4.4.重启SSH服务

```
systemctl restart sshd
```

### 5.使用客户端连接服务端

```
1  创建测试文件
2  [root@home opt]# cd abc
3  [root@home abc]# touch test
4  [root@home abc]# ll
5  总用量 0
6  -rw-r--r-- 1 root root 0 8月  10 16:59 test
7
8  测试客户端连接后, 能否限定内容家目录为/opt/abc
9  [root@home2 opt]# su - abc
10 [abc@home2 ~]$ sftp 192.168.245.129
11 sftp> ls
12 test
13 sftp> pwd
```

```
14 Remote working directory: /
15 sftp> get test
16 Fetching /test to test
17 sftp> cd /opt
18 Couldn't stat remote file: No such file or directory
```

## 6、SSHD服务配置和管理

### 6.1、配置文件详解

/etc/ssh/sshd\_config配置文件内容详解。

#### 1. Port 22

设置SSHD监听端口号。

(1) SSH 预设使用 22 这个port, 也可以使用多个port, 即重复使用 port 这个设定项!

(2) 例如想要开放SSHD端口为 22和222, 则多加一行内容为: Port 222 即可。

(3) 然后重新启动SSHD这样就好了。建议大家修改 port number 为其它端口, 防止别人暴力破解。

例: 修改SSHD服务默认监听的端口为222.

```
[root@base ~]# vim /etc/ssh/sshd_config    #修改ssh服务端配置文件。
```

改:

```
17 Port 22
```

为:

```
17 Port 222
```

```
[root@basessh]# systemctl restart sshd    #重启sshd服务。
```

测试:

```
[root@base74 ~]# netstat -tlnp | grep sshd
tcp      0      0 0.0.0.0:222          0.0.0.0:*           LISTEN
4139/sshd
tcp      0      0 :::222              :::*                 LISTEN      4139/sshd
```

修改完端口默认端口后，登录方法：

```
[root@base ~]# ssh -p 222 192.168.1.63
```

## 2. ListenAddress 0.0.0.0

设置SSHD服务器绑定的IP 地址，0.0.0.0 表示侦听所有地址

安全建议：如果主机不需要从公网ssh访问，可以把监听地址改为内网地址

这个值可以写成本地IP地址，也可以写成所有地址，即0.0.0.0 表示所有IP。

## 3. Protocol 2

选择的 SSH 协议版本，可以是 1 也可以是 2，CentOS 7.x 预设是仅支援V2版本，出于安全考虑，设置为最新的协议版本。

## 4. #HostKey /etc/ssh/ssh\_host\_key

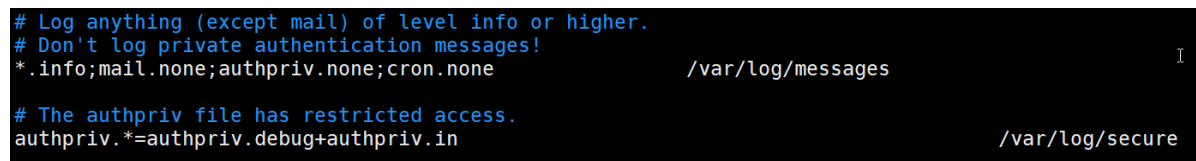
设置包含计算机私人密钥的文件

## 5. SyslogFacility AUTHPRIV

当有人使用 SSH 登入系统的时候，SSH 会记录信息，这个信息要记录的类型为AUTHPRIV，sshd服务日志存放在：/var/log/secure。

例：为什么sshd配置文件中没有指定日志，但日志却存放在了/var/log/secure？

```
[root@base ssh]# vim /etc/rsyslog.conf          #查看日志配置文件。
```



```
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# The authpriv file has restricted access.
authpriv.*=authpriv.debug+authpriv.in                    /var/log/secure
```

图 1-6 日志文件定义

## 6. #LogLevel INFO

定义登录日志记录的等级。

## 6.2、安全调优的重点

### 37 #LoginGraceTime 2m

(1) grace意思是系统给与多少秒来进行登录。

(2) 当使用者连上 SSH server 之后，会出现输入密码的画面，在该画面中。

(3) 在多久时间内没有成功连上 SSH server 就强迫断线！若无单位则默



认时间为秒。

可以根据实际情况来修改实际

### **38 # PermitRootLogin yes**

是否允许 root 登入，默认是允许的，但是建议设定成 no，真实的生产环境服务器，是不允许root账号直接登陆的，仅允许普通用户登录，需要用到root用户再切换到root用户。

43 #PasswordAuthentication yes

密码验证当然是需要的！所以这里写 yes，也可以设置为no，在真实的生产服务器上，根据不同安全级别要求，有的是设置不需要密码登陆的，通过认证的秘钥来登陆。

### **64 # PermitEmptyPasswords no**

是否允许空密码的用户登录，默认为no，不允许空密码登录。

106 # PrintLastLog yes

显示上次登入的信息！默认为 yes 。

例：

```
[root@base ~]# ssh 192.168.1.63    #登录主机。
```

```
Last login: Tue Nov  4 19:57:31 2014 from 192.168.1.107
```

#PrintLastLog yes项定义即是该回显上次登录的信息。

115 # UseDNS yes

一般来说，为了要判断客户端来源是正常合法的，因此会使用 DNS 去反查客户端的主机名，但通常在内网互连时，该设置为no，因此使联机速度会快些。

例：给sshd服务添加一些警告信息。

```
[root@base ~]# cat /etc/motd
```

```
[root@base ~]# echo 'Warning ! 今天好冷!'"> /etc/motd
```

测试：

```
[root@base ~]# ssh 192.168.1.64    #登录主机。
```

```
root@192.168.1.64's password:
```

```
Last login: Thu Jun 23 14:02:38 2016 from 192.168.1.1
```

```
Warning ! 今天好冷!
```

## 实战：配置安全的SSHD服务

SSH的服务认证有两种

### 1.基于口令的安全验证

防护方法：

(1) 密码足够的复杂，密码的长度要大于8位最好大于20位。密码的复杂度是密码要尽可能有数字、大小写字母和特殊符号混合组成。

(2) 修改默认端口号。

(3) 不允许root账号直接登陆到系统，添加普通账号，使用普通账号登录系统，授予root的权限，必要时再从普通用户切换到root用户。

(4) 不允许密码登陆，只能通过认证的密钥来登陆系统。

### 2.基于密钥的安全认证

原理：1.首先你需要在客户端创建一对密钥，并将公钥存放在需要访问的服务器上，将私钥存放在SSH客户端主机上。

2.客户端向远程服务器发出连接请求，请求使用密钥进行安全验证。

远程服务器在收到请求之后，会先在SSH连接

用户的家目录下查找事先存放的用户的公用密钥，然后将它与客户端发送过来的公钥密钥进行匹配。

如果两者一致，SSH服务器就用公钥密钥加密“质询”，并把它发送给SSH客户端，客户端收到“质询”后，用自己的私钥进行解密

再把它发送给远程服务器，认证通过后，建立连接

实验环境：

服务端：base IP：192.168.1.63

客户端：base2 IP：192.168.1.64

客户端生成密钥对，然后把公钥传输到服务端

```
[root@base2 ~]# ssh-keygen          #-t 指定算法加密 ssh-keygen -t dsa
```

这命令则会生成dsa算法加密的密钥

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa): #提示输入密钥

文件的保存路径，选择默认，回车继续

Enter passphrase (empty for no passphrase):     #下面要求输入密码，这里的passphrase 密码是对生成的私钥文件(/root/.ssh/id\_rsa)的保护口令，如果不设置可以回车。

Enter same passphrase again:     #直接回车。

Your identification has been saved in /root/.ssh/id\_rsa.

Your public key has been saved in /root/.ssh/id\_rsa.pub.

The key fingerprint is:

da:2c:d8:53:92:6e:ff:4a:54:14 23:28:b3:bb:3b root@base2

The key's randomart image is:

+--[ RSA 2048 ]-----+

```
|      .o+      |
|      o ... +   |
|      + ...     |
|      ..        |
|      o.S       |
|      +.B       |
|      . B.+     |
|      .E=       |
|      .ooo.     |
```

+-----+

[root@base2 ~]# cd /root/.ssh/     #切换工作目录至家目录下的.ssh目录下。

[root@base2 .ssh]# ls     #可查看到生成的id\_rsa、id\_rsa.pub文件。  
id\_rsa id\_rsa.pub known\_hosts  
私钥   公钥

## 2. 发布公钥到服务端。

使用ssh-copy-id 命令将客户端生成的公钥发布到远程服务器  
192.168.1.63 base 。

[root@base2 .ssh]# ssh-copy-id -i ~/.ssh/id\_rsa.pub

root@192.168.1.63

The authenticity of host '192.168.1.63(192.168.1.63)' can't be established.

RSA key fingerprint is d9:17:d7:db:38:7c:e8:56:9c:4b:7e:00:7f:9e:1c:74.

Are you sure you want to continue connecting (yes/no)? yes     #输入

yes

Warning: Permanently added '192.168.1.64' (RSA) to the list of known hosts.

root@192.168.1.63's password: #输入192.168.1.63主机登录密码。

Now try logging into the machine, with "ssh '192.168.1.63'", and check in:

.ssh/authorized\_keys

to make sure we haven't added extra keys that you weren't expecting

#这个时候可以通过ssh 无密钥直接登陆主机

注意：如果服务器不是监听22端口，则需要指定端口传输密钥：

```
[root@base2 .ssh]# ssh-copy-id -i ~/.ssh/id_rsa.pub -p 222
```

root@192.168.245.1.63

### 3.非交互式生成指定密钥

```
1 [root@home .ssh]# ssh-keygen -t dsa -q -P "" -f  
~/.ssh/id_dsa
```

-q:安静输出

-P:提供输入

-f:生成路径

### 实例：三方免密脚本

```
1 #/bin/bash  
2  
3 # 注意事项  
4 print_n(){  
5     echo -e "\e[1m\033[31m请在脚本后接 ip总数 然后再后面接  
上目标ip\033[0m\e[0m"  
6     echo "例如: "  
7     echo -e "sh ssh-key.sh \e[1m\033[33m3\033[0m\e[0m  
\033[34mip1\033[0m \033[35mip2\033[0m  
\033[36mip3\033[0m"
```

```
8     exit 25
9 }
10
11 # 检测参数
12 if [[ -z $1 ]];then
13     print_n
14 fi
15
16 # 参数赋值
17 n=0
18 for fid in `echo $*` ;do
19     if [[ $n == 0 ]];then
20
21         ids2[$n]=1
22     else
23         if `ping -w 1 $fid &>/dev/null`;then
24             echo "$fid 网络可通"
25         else
26             echo "$fid 网络不通"
27             exit 28
28         fi
29         ids2[$n]=$fid
30     fi
31     ids[$n]=$fid
32     ((n++))
33 done
34
35 # echo ${ids[*]} ${#ids[*]}
36 # echo ${ids2[*]} ${#ids2[*]}
37
38
39
40 # 检测到expect
41 if `expect -v &>/dev/null` ;then
42     echo "检测到expect"
43 else
44     echo "没有检测到expect"
45     if `yum install -y expect` ;then
46         echo "安装expect成功"
47     else
```

```
48         echo "安装expect失败"
49         exit 25
50     fi
51 fi
52
53 # 生成密钥
54 ssh_keygen_lib(){
55     cat >ssh-keygen-n.exp << ww
56     spawn ssh-keygen
57     expect {
58     "y/n" { send "y\r";exp_continue }
59     ":" { send "\r" ;exp_continue}
60     }
61     ww
62     expect ssh-keygen-n.exp
63     rm -rf ssh-keygen-n.exp
64     return 0
65 }
66
67 # 分发密钥
68 ssh_copy_id_lib(){
69     cat >ssh-copy-id.exp << ww
70     spawn ssh-copy-id $1
71     expect {
72     "yes/no" { send "yes\r";exp_continue }
73     "password:" { send "123456\r" ;exp_continue}
74     }
75     ww
76     expect ssh-copy-id.exp
77     rm -rf ssh-copy-id.exp
78     return 0
79 }
80 ssh_sh(){
81     expect << ww
82
83     spawn ssh $1
84     expect "from"
85     send "sh /opt/$0 ${ids2[*]} \r"
86     send "\r"
87     expect eof
```

```
88 ww
89     return 0
90 }
91
92 # run
93 ssh_keygen_lib
94 for ((i=1;i<${#ids[*]};i++)) ;do
95     if [[ $i != 0 ]];then
96         ssh_copy_id_lib ${ids[$i]}
97     fi
98     if [[ $1 > 1 ]] ;then
99         scp $0 ${ids[$i]}:/opt/$0
10        ssh_sh ${ids[$i]}
10    fi
10 done
10
10
```