晴川のLinux笔记-Linux命令Ⅲ

Linux简单管理

ssh 简介及基本操作

简介

SSH，全名 secure shell，其目的是用来从终端与远程机器交互，SSH 设计之处初，遵循了如下原则：

- 机器之间通讯的内容必须经过加密。
- 加密过程中，通过 public key 加密，private 解密。

密钥

SSH 通讯的双方各自持有一个公钥私钥对，公钥对对方是可见的，私钥仅持有者可见，你可以通过"ssh-keygen"生成自己的公私钥，默认情况下，公私钥的存放路径如下：

- 公钥：$HOME/.ssh/id_rsa.pub
- 私钥：$HOME/.ssh/id_rsa

基于口令的安全验证通讯原理

建立通信通道的步骤如下：

```
1.  远程主机将公钥发给用户 ---- 远程主机收到用户的登录请求，把自己的公钥发给客户端，客户端检查这个 public key 是否在自己的 $HOME/.ssh/known_hosts 中，如果没有，客户端会提示是否要把这个 public key 加入到 known_hosts 中。
2.  用户使用公钥加密密码 ------ 用户使用这个公钥，将登录密码加密后，发送回来
3.  远程主机使用私钥加密 ------ 远程主机用自己的私钥，解密登录密码，如果密码正确，就同意用户登录。
4.  客户端把 PUBLIC KEY(client)，发送给服务器。
5.  至此，到此为止双方彼此拥有了对方的公钥，开始双向加密解密。
```

PS：当网络中有另一台冒牌服务器冒充远程主机时，客户端的连接请求被服务器 B 拦截，服务器 B 将自己的公钥发送给客户端，客户端就会将密码加密后发送给冒牌服务器，冒牌服务器就可以拿自己的私钥获取到密码，然后为所欲为。因此当第一次链接远程主机时，在上述步骤中，会提示您当前远程主机的"公钥指纹"，以确认远程主机是否是正版的远程主机，如果选择继续后就可以输入密码进行登录了，当远程的主机接受以后，该台服务器的公钥就会保存到 ~/.ssh/known_hosts 文件中。

StrictHostKeyChecking 和 UserKnownHostsFile

> - 如何让连接新主机时，不进行公钥确认?
>   - SSH 客户端的 StrictHostKeyChecking 配置指令，可以实现当第一次连接服务器时，自动接受新的公钥。
>   - [例子:] ssh -o StrictHostKeyChecking=no 192.168.0.110
> - 如何防止远程主机公钥改变导致 SSH 连接失败
>   - 将本地的 known_hosts 指向不同的文件，就不会造成公钥冲突导致的中断了,提示信息由公钥改变中断警告，变成了首次连接的提示。结合自动接收新的公钥
>   - [例子:] ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null 192.168.0.110

基于密匙的安全验证通讯原理

这种方式你需要在当前用户家目录下为自己创建一对密匙，并把公匙放在需要登录的服务器上。当你要连接到服务器上时，客户端就会向服务器请求使用密匙进行安全验证。服务器收到请求之后，会在该服务器上你所请求登录的用户的家目录下寻找你的公匙，然后与你发送过来的公匙进行比较。如果两个密匙一致，服务器就用该公匙加密"质询"并把它发送给客户端。客户端收到"质询"之后用自己的私匙解密再把它发送给服务器。与第一种级别相比，第二种级别不需要在网络上传送口令。

PS：简单来说，就是将客户端的公钥放到服务器上，那么客户端就可以免密码登录服务器了，那么客户端的公钥应该放到服务器上哪个地方呢？默认为你要登录的用户的家目录下的 .ssh 目录下的 authorized_keys 文件中（即：~/.ssh/authorized_keys）。 我们的目标是：用户已经在主

机 A 上登录为 a 用户，现在需要以不输入密码的方式以用户 b 登录到主机 B 上。

可以把密钥理解成一把钥匙，公钥理解成这把钥匙对应的锁头， 把锁头（公钥）放到想要控制的 server 上，锁住 server, 只有拥有钥匙（密钥）的人， 才能打开锁头，进入 server 并控制 而对于拥有这把钥匙的人，必需得知道钥匙本身的密码，才能使用这把钥匙（除非这把钥匙没设置密码）, 这样就可以防止钥匙被了配了（私钥被人复制）

当然，这种例子只是方便理解罢了， 拥有 root 密码的人当然是不会被锁住的，而且不一定只有一把锁（公钥）, 但如果任何一把锁，被人用其对应的钥匙（私钥）打开了，server 就可以被那个人控制了 所以说，只要你曾经知道 server 的 root 密码，并将有 root 身份的公钥放到上面， 就可以用这个公钥对应的私钥"打开" server, 再以 root 的身分登录，即使现在 root 密码已经更改!

步骤如下:

1. 以用户 a 登录到主机 A 上，生成一对公私钥。
2. 把主机 A 的公钥复制到主机 B 的 authorized_keys 中，可能需要输入 b@B 的密码。

   ssh-copy-id -i ~/.ssh/id_dsa.pub b@B
3. 在 a@A 上以免密码方式登录到 b@B

   ```
   ssh b@B
   ```

tips:

假如 B 机器修改端口后，将主机 A 上的公钥复制到 B 机的操作方法是（下面方法中双引号是必须的）:

ssh-copy-id "-p 端口号 b@B"

SSH 端口转发

SSH 还同时提供了一个非常有用的功能，这就是端口转发。它能够将其他 TCP 端口的网络数据通过 SSH 链接来转发，并且自动提供了相应的加密及解密服务。这一过程有时也被叫做“隧道”(tunneling)，这是因为 SSH 为其他 TCP 链接提供了一个安全的通道来进行传输而得名。

SSH 端口转发自然需要 SSH 连接，而 SSH 连接是有方向的，从 SSH Client 到 SSH Server 。 而我们所要访问的应用也是有方向的，应用连接的方向也是从应用的 Client 端连接到应用的 Server 端。 比如需要我们要访问 Internet 上的 Web 站点时，Http 应用的方向就是从我们自己这台主机 (Client) 到远处的 Web Server。

> - SSH 连接和应用的连接这两个连接的方向一致，那我们就说它是本地转发。
> - SSH 连接和应用的连接这两个连接的方向不同，那我们就说它是远程转发。

SSH 正向连接

正向连接就是 client 连上 server，然后把 server 能访问的机器地址和端口（当然也包括 server 自己）镜像到 client 的端口上。

```
何时使用本地 Tunnel？

> * 比如说你在本地访问不了某个网络服务（如 www.google.com)，而有一台机器（如：xx.xx.xx.xx) 可以，那么你就可以通过这台机器的 ssh 服务来转发
```

使用方法

```
ssh -L <local port>:<remote host>:<remote port> <SSH hostname>
ssh -L [客户端 IP 或省略]:[客户端端口]:[服务器能访问的 IP]:[服务器能访问的 IP 的端口] [登陆服务器的用户名 @服务器 IP] -p [服务器 ssh 服务端口（默认 22）]
```

ssh -L 1433:target_server:1433 user@ssh_host

windows 下使用本地转发 xshell

```
(1)ssh 远程连接到 Linux
(2) 打开代理设置面板，点击：view -> Tunneling Pane，在弹出的窗口选择 Forwarding Rules
(3) 在空白处右键：add。
在弹出的 Forwarding Rule，
Type 选择"Local(Outgoing)";
Source Host 使用默认的 localhost; Listen Port 添上端口 8888;
Destination Host 使用默认的 localhost；Destination Port 添上 80;

Destination Host 设置为 localhost 为要访问的机器，可以设置为登陆后的机器可以访问到的 IP
```

SSH 反向连接

反向连接就是 client 连上 server，然后把 client 能访问的机器地址和端口（也包括 client 自己）镜像到 server 的端口上。

```
何时使用反向连接？

比如当你下班回家后就访问不了公司内网的机器了，遇到这种情况可以事先在公司内网的机器上执行远程 Tunnel，连上一台公司外网的机器，等你下班回家后就可以通过公司外网的机器去访问公司内网的机器了。
```

使用方法

```
ssh -R <remote port>:<local host>:<local port> <SSH hostname>
ssh -R [服务器 IP 或省略]:[服务器端口]:[客户端能访问的 IP]:[客户端能访问的 IP 的端口] [登陆服务器的用户名 @服务器 IP] -p [服务器 ssh 服务端口（默认 22）]
```

外网机器 A 要控制 内网机器 B

A 主机：外网，ip：122.122.122.122，sshd 端口：2222（默认是 22)

B 主机：内网，sshd 端口：2222（默认是 22)

无论是外网主机 A，还是内网主机 B 都需要跑 ssh daemon

首先在内网机器 B 上执行

```
    ssh -NfR 1234:localhost:2222 user1@122.122.122.122 -p 2222
```

这句话的意思是将 A 主机的 1234 端口和 B 主机的 2222 端口绑定，相当于远程端口映射 (Remote Port Forwarding)。

外网机器 A 会 listen 本地 1234 端口

```
---- 外网机器 A sshd 会 listen 本地 1234 端口
    #netstat -tanp | grep sshd
    #Proto Recv-Q Send-Q   Local Address   Foreign Address   State       PID/Program name
    #tcp     0      0      127.0.0.1:1234    0.0.0.0:*    LISTEN      4234/sshd

---- 在外网机器 A 登录内网机器 B（非 root 用户的话，直接 user@localhost 即可）
    #ssh user@localhost -p1234
```

SSH 反向连接自动重连

上面的反向连接（Reverse Connection）不稳定，可能随时断开，需要内网主机 B 再次向外网 A 发起连接，这时需要个"朋友"帮你在内网 B 主机执行这条命令。可以使用 Autossh 或者 while 进行循环。

(1) 在 B 机器上将 B 机器公钥放到外网机器 A 上（实现 B 机器自动登录到 A 机器）

(2) 用 Autossh 或者 while 循环 保持 ssh 反向隧道一直连接，CentOS 需要使用 epel 源下载

在 CentOS6 和 CentOS7 都可以执行下面的命令安装 epel 仓库

while

编写脚本写入如下内容

```
#!/bin/bash
# 远程机器的 IP 和端口
remote_ip=122.122.122.122
remote_port=2222

while [[ 1==1  ]]
do
    ssh  -o ServerAliveInterval=15 -o ServerAliveCountMax=3 -N -R:1234:localhost:22 -p ${remote_port} root@${remote_ip}
    sleep 3
done
```

执行脚本后，即可以通过登陆 122.122.122.122 机器访问本地 1234 端口进行访问此机器

注;可以将此脚本放在后台中运行，并加到系统自启动程序中

autossh

```
#yum -y install epel-release
```

安装号 autossh 后使用如下方法进行反向连接

```
#autossh -M 5678 -NfR 1234:localhost:2222 user1@122.122.122.122 -p2222
```

比之前的命令添加的一个 -M 5678 参数，负责通过 5678 端口监视连接状态，连接有问题时就会自动重连

windows下xshell使用

- 私钥，在 Xshell 里也叫用户密钥
- 公钥，在 Xshell 里也叫主机密钥

  利用 xshell 密钥管理服务器远程登录，

  (1)Xshell 自带有用户密钥生成向导：点击菜单栏的工具 ->新建用户密钥生成向导 (2) 添加公钥 (Pubic Key) 到远程 Linux 服务器