SSH原理与应用

ssh在程序员的生活中还是非常常见的，ssh具有很多种功能，也可以用在很多种场合。

什么是SSH

SSH是一种网络协议，用于计算机之间的加密登录

当我们在一台电脑上面，运用ssh登录了另一台计算机，我们便可以认为，这种登录是安全的了，因为即使中途被截获，我们的密码也不会泄漏。

最早的时候，互联网通信都是明文通信，一旦被截获，内容就暴露无疑。1995年，芬兰学者Tatu Ylonen设计了SSH协议，将登录信息全部加密，成为互联网安全的一个基本解决方案，迅速在全世界获得推广，目前已经成为Linux系统的标准配置。

需要指出的是，SSH只是一种协议，存在多种实现，既有商业实现，也有开源实现。本文针对的实现是OpenSSH，它是自由软件，应用非常广泛。

此外，本文只讨论SSH在Linux Shell中的用法。如果要在Windows系统中使用SSH，会用到另一种软件PuTTY，这需要另文介绍。

用法

1. 登录远程服务器 `ssh root@host`
2. 如果当前用户与远程用户同名 `ssh host`
3. ssh默认的端口是22，如果我们要修改登录的默认端口 `ssh -p xx root@host`

中间人攻击

ssh采用的是非对称加密，也就是要采用公钥和私钥的方式进行加密。

整个通信的过程是这样的： 1. 远程主机收到用户的登录请求，将公钥发送给用户 2. 用户使用这个公钥，将登录的密码进行加密，发送给后台 3. 远程主机，用自己的私钥进行解密，判断用户名密码是否正确

整个过程看起来是很完美的，但是容易产生一种中间人攻击的现象：

我们发送出去的登录的信息，被中途截获了，一个中间人，将他的公钥发送过来，这样用户加密之后，他便可以用自己的私钥解密了，这样他就拥有了我们的密码，并且可以一直在中间监听我们的通话。

当然，这是基于口令的通信方式，我们也可以采用基于密钥的加密方式：

第二种级别（基于密匙的安全验证）需要依靠密匙，也就是你必须为自己创建一对密匙，并把公用密匙放在需要访问的服务器上。 如果你要连接到SSH服务器上，客户端软件就会向服务器发出请求，请求用你的密匙进行安全验证。服务器收到请求之后，先在你在该服务器的家目录下寻找你的公用密匙，然后把它和你发送过来的公用密匙进行比较。如果两个密匙一致，服务器就用公用密匙加密“质询”（challenge）并把它发送给客户端软件。客户端软件收到“质询”之后就可以用你的私人密匙解密再把它发送给服务器。

这样我们便可以防止中间人攻击的现象了。

口令登录

```
$ ssh user@host
The authenticity of host 'host (12.18.429.21)' can't be established.
RSA key fingerprint is 98:2e:d7:e0:de:9f:ac:67:28:c2:42:2d:37:16:58:4d.
Are you sure you want to continue connecting (yes/no)?
```

这段话的意思是，无法确认host主机的真实性，只知道它的公钥指纹，问你还想继续连接吗？ 所谓"公钥指纹"，是指公钥长度较长（这里采用RSA算法，长达1024位），很难比对，所以对其进行MD5计算，将它变成一个128位的指纹。上例中是98:2e:d7:e0:de:9f:ac:67:28:c2:42:2d:37:16:58:4d，再进行比较，就容易多了。 很自然的一个问题就是，用户怎么知道远程主机的公钥指纹应该是多少？回答是没有好办法，远程主机必须在自己的网站上贴出公钥指纹，以便用户自行核对。 假定经过风险衡量以后，用户决定接受这个远程主机的公钥。

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

系统会出现一句提示，表示host主机已经得到认可。

```
Warning: Permanently added 'host,12.18.429.21' (RSA) to the list of known hosts.
```

然后，会要求输入密码。

```
Password: (enter password)
```

如果密码正确，就可以登录了。 当远程主机的公钥被接受以后，它就会被保存在文件$HOME/.ssh/known_hosts之中。下次再连接这台主机，系统就会认出它的公钥已经保存在本地了，从而跳过警告部分，直接提示输入密码。 每个SSH用户都有自己的known_hosts文件，此外系统也有一个这样的文件，通常是/etc/ssh/ssh_known_hosts，保存一些对所有用户都可信赖的远程主机的公钥。

公钥登录

使用密码登录，每次都必须输入密码，非常麻烦。好在SSH还提供了公钥登录，可以省去输入密码的步骤。

所谓"公钥登录"，原理很简单，就是用户将自己的公钥储存在远程主机上。登录的时候，远程主机会向用户发送一段随机字符串，用户用自己的私钥加密后，再发回来。远程主机用事先储存的公钥进行解密，如果成功，就证明用户是可信的，直接允许登录shell，不再要求密码。 这种方法要求用户必须提供自己的公钥。如果没有现成的，可以直接用ssh-keygen生成一个：

```
$ ssh-keygen
```

运行上面的命令以后，系统会出现一系列提示，可以一路回车。其中有一个问题是，要不要对私钥设置口令（passphrase），如果担心私钥的安全，这里可以设置一个。 运行结束以后，在$HOME/.ssh/目录下，会新生成两个文件：id_rsa.pub和id_rsa。前者是你的公钥，后者是你的私钥。 这时再输入下面的命令，将公钥传送到远程主机host上面：

```
$ ssh-copy-id user@host
```

好了，从此你再登录，就不需要输入密码了。 如果还是不行，就打开远程主机的/etc/ssh/sshd_config这个文件，检查下面几行前面"#"注释是否取掉。

```
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
```

然后，重启远程主机的ssh服务。

```
    // ubuntu系统
   service ssh restart
   // debian系统
   /etc/init.d/ssh restart
```

authorized_keys文件

远程主机将用户的公钥，保存在登录后的用户主目录的$HOME/.ssh/authorized_keys文件中。公钥就是一段字符串，只要把它追加在authorized_keys文件的末尾就行了。

这里不使用上面的ssh-copy-id命令，改用下面的命令，解释公钥的保存过程：

```
$ ssh user@host 'mkdir -p .ssh && cat >> .ssh/authorized_keys' < ~/.ssh/id_rsa.pub
```

这条命令由多个语句组成，依次分解开来看：（1）"$ ssh user@host"，表示登录远程主机；（2）单引号中的mkdir .ssh && cat >> .ssh/authorized_keys，表示登录后在远程shell上执行的命令：（3）"$ mkdir -p .ssh"的作用是，如果用户主目录中的.ssh目录不存在，就创建一个；（4）'cat >> .ssh/authorized_keys' < ~/.ssh/id_rsa.pub的作用是，将本地的公钥文件~/.ssh/id_rsa.pub，重定向追加到远程文件authorized_keys的末尾。 写入authorized_keys文件后，公钥登录的设置就完成了。

配置ssh config

```
vi ~/.ssh/config

// 文件内容如下

Host js //别名, 可以直接执行 ssh js

HostName 172.16.6.84 //Host别名指向的服务器 IP

User zhangsan //登录所用的用户名

PreferredAuthentications publickey //鉴权方式

IdentityFile ~/.ssh/zhangsan.pem //认证所需的密钥
```

这样我们便可以通过 `ssh js` 来代替曾经的 `ssh xxx@111.11.11.11` 并且采用公钥+私钥的加密方式，不用输入密码，非常的方便。

参考文献

- SSH原理与运用
- 网络安全协议比较