SSH原理与运用（一）：远程登录

一、什么是**SSH**？

简单说，SSH是一种网络协议，用于计算机之间的加密登录。

如果一个用户从本地计算机，使用SSH协议登录另一台远程计算机，我们就可以认为，这种登录是安全的，即使被中途截获，密码也不会泄露。

二、最基本的用法

SSH主要用于远程登录。假定你要以用户名user，登录远程主机host，只要一条简单命令就可以了。

```
$ ssh user@host
```

如果本地用户名与远程用户名一致，登录时可以省略用户名。

```
$ ssh host
```

SSH的默认端口是22，也就是说，你的登录请求会送进远程主机的22端口。使用p参数，可以修改这个端口。

```
$ ssh -p 2222 user@host
```

上面这条命令表示，ssh直接连接远程主机的2222端口。

三、中间人攻击

SSH之所以能够保证安全，原因在于它采用了公钥加密。

整个过程是这样的：（1）远程主机收到用户的登录请求，把自己的公钥发给用户。（2）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。

这个过程本身是安全的，但是实施的时候存在一个风险：如果有人截获了登录请求，然后冒充远程主机，将伪造的公钥发给用户，那么用户很难辨别真伪。因为不像https协议，SSH协议的公钥是没有证书中心（CA）公证的，也就是说，都是自己签发的。

可以设想，如果攻击者插在用户与远程主机之间（比如在公共的wifi区域），用伪造的公钥，获取用户的登录密码。再用这个密码登录远程主机，那么SSH的安全机制就荡然无存了。这种风险就是著名的"中间人攻击"（Man-in-the-middle attack）。

SSH协议是如何应对的呢？

四、口令登录

如果你是第一次登录对方主机，系统会出现下面的提示：

```
$ ssh user@host

The authenticity of host 'host (12.18.429.21)' can't be established.

RSA key fingerprint is 98:2e:d7:e0ᴅᴇ9f: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:e0ᴅᴇ9f: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。

五、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

六、**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。

==========