<https://baike.baidu.com/item/ssh/10407?fr=aladdin>

SSH 为 [Secure Shell](https://baike.baidu.com/item/Secure%20Shell) 的缩写，由 IETF 的网络小组（Network Working Group）所制定；SSH 为建立在应用层基础上的安全协议。SSH 是目前较可靠，专为[远程登录](https://baike.baidu.com/item/%E8%BF%9C%E7%A8%8B%E7%99%BB%E5%BD%95" \t "_blank)会话和其他网络服务提供安全性的协议。利用 SSH 协议可以有效防止远程管理过程中的信息泄露问题。SSH最初是UNIX系统上的一个程序，后来又迅速扩展到其他操作平台。SSH在正确使用时可弥补网络中的漏洞。SSH客户端适用于多种平台。几乎所有UNIX平台—包括[HP-UX](https://baike.baidu.com/item/HP-UX" \t "_blank)、[Linux](https://baike.baidu.com/item/Linux" \t "_blank)、[AIX](https://baike.baidu.com/item/AIX" \t "_blank)、[Solaris](https://baike.baidu.com/item/Solaris/3517" \t "_blank)、[Digital](https://baike.baidu.com/item/Digital" \t "_blank) [UNIX](https://baike.baidu.com/item/UNIX)、[Irix](https://baike.baidu.com/item/Irix" \t "_blank)，以及其他平台，都可运行SSH。

**中文名**

安全外壳协议

**外文名**

Secure Shell

**简    称**

[SSH](https://baike.baidu.com/item/SSH/5400403)

**作    用**

提供安全性的协议

## 目录

1. 1 [功能](https://baike.baidu.com/item/ssh/10407?fr=aladdin#1)
2. 2 [验证](https://baike.baidu.com/item/ssh/10407?fr=aladdin#2)
3. 3 [详细](https://baike.baidu.com/item/ssh/10407?fr=aladdin#3)
4. 4 [层次](https://baike.baidu.com/item/ssh/10407?fr=aladdin#4)
5. 5 [结构](https://baike.baidu.com/item/ssh/10407?fr=aladdin#5)
6. 6 [应用](https://baike.baidu.com/item/ssh/10407?fr=aladdin#6)
7. 7 [扩展](https://baike.baidu.com/item/ssh/10407?fr=aladdin#7)
8. 8 [启动方法](https://baike.baidu.com/item/ssh/10407?fr=aladdin#8)
9. 9 [SSH安全技巧](https://baike.baidu.com/item/ssh/10407?fr=aladdin#9)
10. ▪ [服务器间跳转](https://baike.baidu.com/item/ssh/10407?fr=aladdin#9_1)
11. ▪ [省去用户名](https://baike.baidu.com/item/ssh/10407?fr=aladdin#9_2)
12. ▪ [主机别名](https://baike.baidu.com/item/ssh/10407?fr=aladdin#9_3)
13. ▪ [省略主机名](https://baike.baidu.com/item/ssh/10407?fr=aladdin#9_4)
14. ▪ [连接中转](https://baike.baidu.com/item/ssh/10407?fr=aladdin#9_5)
15. ▪ [别再输入密码](https://baike.baidu.com/item/ssh/10407?fr=aladdin#9_6)
16. ▪ [长连接](https://baike.baidu.com/item/ssh/10407?fr=aladdin#9_7)
17. ▪ [多条连接共享](https://baike.baidu.com/item/ssh/10407?fr=aladdin#9_8)
18. ▪ [加速连接](https://baike.baidu.com/item/ssh/10407?fr=aladdin#9_9)
19. ▪ [减少延迟](https://baike.baidu.com/item/ssh/10407?fr=aladdin#9_10)

## 功能

[编辑](javascript:;)

传统的网络服务程序，如：[ftp](https://baike.baidu.com/item/ftp)、pop和[telnet](https://baike.baidu.com/item/telnet" \t "_blank)在本质上都是不安全的，因为它们在网络上用[明文](https://baike.baidu.com/item/%E6%98%8E%E6%96%87" \t "_blank)传送口令和数据，别有用心的人非常容易就可以截获这些口令和数据。而且，这些服务程序的[安全验证](https://baike.baidu.com/item/%E5%AE%89%E5%85%A8%E9%AA%8C%E8%AF%81)方式也是有其弱点的， 就是很容易受到“中间人”（man-in-the-middle）这种方式的攻击。所谓“中间人”的攻击方式， 就是“中间人”冒充真正的[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "_blank)接收你传给服务器的数据，然后再冒充你把数据传给真正的服务器。服务器和你之间的数据传送被“中间人”一转手做了手脚之后，就会出现很严重的问题。通过使用SSH，你可以把所有传输的数据进行加密，这样"中间人"这种攻击方式就不可能实现了，而且也能够防止DNS欺骗和IP欺骗。使用SSH，还有一个额外的好处就是传输的数据是经过压缩的，所以可以加快传输的[速度](https://baike.baidu.com/item/%E9%80%9F%E5%BA%A6/5456)。SSH有很多功能，它既可以代替[Telnet](https://baike.baidu.com/item/Telnet" \t "_blank)，又可以为[FTP](https://baike.baidu.com/item/FTP" \t "_blank)、[PoP](https://baike.baidu.com/item/PoP" \t "_blank)、甚至为[PPP](https://baike.baidu.com/item/PPP" \t "_blank)提供一个安全的"通道"[1]  。

## 验证

[编辑](javascript:;)

从客户端来看，SSH提供两种级别的安全验证。

**第一种级别（基于口令的安全验证）**

只要你知道自己帐号和口令，就可以登录到远程主机。所有传输的数据都会被加密，但是不能保证你正在连接的服务器就是你想连接的[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8)。可能会有别的服务器在冒充真正的服务器，也就是受到“中间人”这种方式的攻击。

**第二种级别（基于密匙的安全验证）**

需要依靠[密匙](https://baike.baidu.com/item/%E5%AF%86%E5%8C%99)，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，先在该服务器上你的主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。

用这种方式，你必须知道自己密匙的[口令](https://baike.baidu.com/item/%E5%8F%A3%E4%BB%A4)。但是，与第一种级别相比，第二种级别不需要在网络上传送口令。

第二种级别不仅加密所有传送的数据，而且“中间人”这种攻击方式也是不可能的（因为他没有你的私人密匙）。但是整个登录的过程可能需要10秒[2]  。

## 详细

[编辑](javascript:;)

如果你考察一下接入[ISP](https://baike.baidu.com/item/ISP)(Internet Service Provider，互联网服务供应商)或大学的方法，一般都是采用[Telnet](https://baike.baidu.com/item/Telnet" \t "_blank)或POP邮件客户进程。因此，每当要进入自己的账号时，你输入的密码将会以明码方式发送(即没有保护，直接可读)，这就给攻击者一个盗用你账号的机会—最终你将为他的行为负责。由于SSH的源代码是公开的，所以在[Unix](https://baike.baidu.com/item/Unix" \t "_blank)世界里它获得了广泛的认可。[Linux](https://baike.baidu.com/item/Linux" \t "_blank)其源代码也是公开的，大众可以免费获得，并同时获得了类似的认可。这就使得所有开发者(或任何人)都可以通过补丁程序或b u g修补来提高其性能，甚至还可以增加功能。开发者获得并安装SSH意味着其性能可以不断提高而无须得到来自原始创作者的直接技术支持。SSH替代了不安全的远程应用程序。SSH是设计用来替代[伯克利](https://baike.baidu.com/item/%E4%BC%AF%E5%85%8B%E5%88%A9)版本的r命令集的；它同时继承了类似的语法。其结果是，使用者注意不到使用SSH和r命令集的区别。利用它，你还可以干一些很酷的事。通过使用SSH，你在不安全的网络中发送信息时不必担心会被监听。你也可以使用POP通道和Telnet方式，通过SSH可以利用[PPP](https://baike.baidu.com/item/PPP" \t "_blank)通道创建一个虚拟个人网络( Virtual Private Network,[VPN](https://baike.baidu.com/item/VPN))。SSH也支持一些其他的身份认证方法，如Kerberos和安全ID卡等。

但是因为受版权和加密算法的限制，可以预计将来会有越来越多的人使用SSH而不是Telnet或者POP3等[3]  。

## 层次

[编辑](javascript:;)

SSH 主要由三部分组成：

**传输层协议 [SSH-TRANS]**

提供了服务器认证，保密性及完整性。此外它有时还提供压缩功能。 SSH-TRANS 通常运行在[TCP/IP](https://baike.baidu.com/item/TCP%2FIP" \t "_blank)连接上，也可能用于其它可靠数据流上。 SSH-TRANS 提供了强力的加密技术、密码主机认证及完整性保护。该协议中的认证基于主机，并且该协议不执行[用户认证](https://baike.baidu.com/item/%E7%94%A8%E6%88%B7%E8%AE%A4%E8%AF%81" \t "_blank)。更高层的用户认证协议可以设计为在此协议之上。

**用户认证协议 [SSH-USERAUTH]**

用于向服务器提供客户端用户鉴别功能。它运行在[传输层](https://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E5%B1%82" \t "_blank)协议 SSH-TRANS 上面。当SSH-USERAUTH 开始后，它从低层协议那里接收会话[标识符](https://baike.baidu.com/item/%E6%A0%87%E8%AF%86%E7%AC%A6" \t "_blank)（从第一次[密钥](https://baike.baidu.com/item/%E5%AF%86%E9%92%A5" \t "_blank)交换中的交换哈希H ）。会话标识符唯一标识此会话并且适用于标记以证明私钥的所有权。 SSH-USERAUTH 也需要知道低层协议是否提供保密性保护。

**连接协议 [SSH-CONNECT]**

将多个加密隧道分成逻辑通道。它运行在用户认证协议上。它提供了交互式登录话路、远程命令执行、转发 TCP/IP 连接和转发 X11 连接。

## 结构

[编辑](javascript:;)

SSH是由客户端和[服务端](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E7%AB%AF" \t "_blank)的软件组成的，有两个不兼容的版本分别是：1.x和2.x。 用SSH 2.x的客户程序是不能连接到SSH 1.x的服务程序上去的。OpenSSH 2.x同时支持SSH 1.x和2.x。

服务端是一个[守护进程](https://baike.baidu.com/item/%E5%AE%88%E6%8A%A4%E8%BF%9B%E7%A8%8B)(daemon)，他在后台运行并响应来自客户端的连接请求。服务端一般是sshd进程，提供了对远程连接的处理，一般包括公共密钥认证、密钥交换、[对称密钥](https://baike.baidu.com/item/%E5%AF%B9%E7%A7%B0%E5%AF%86%E9%92%A5" \t "_blank)加密和非安全连接。

客户端包含ssh程序以及像[scp](https://baike.baidu.com/item/scp" \t "_blank)（远程拷贝）、slogin（远程登陆）、[sftp](https://baike.baidu.com/item/sftp" \t "_blank)（安全文件传输）等其他的应用程序。

他们的工作机制大致是本地的客户端发送一个连接请求到远程的服务端，服务端检查申请的包和IP地址再发送密钥给SSH的客户端，本地再将密钥发回给服务端，自此连接建立。SSH 1.x和SSH 2.x在连接协议上有一些差异。

一旦建立一个安全传输层连接，客户机就发送一个服务请求。当用户认证完成之后，会发送第二个服务请求。这样就允许新定义的协议可以与上述协议共存。连接协议提供了用途广泛的各种通道，有标准的方法用于建立安全交互式会话外壳和转发（“[隧道技术](https://baike.baidu.com/item/%E9%9A%A7%E9%81%93%E6%8A%80%E6%9C%AF)”）专有 TCP/IP 端口和 X11 连接。

SSH被设计成为工作于自己的基础之上而不利用超级服务器([inetd](https://baike.baidu.com/item/inetd" \t "_blank))，虽然可以通过inetd上的[tcpd](https://baike.baidu.com/item/tcpd" \t "_blank)来运行SSH进程，但是这完全没有必要。启动SSH服务器后，sshd运行起来并在默认的22端口进行监听（你可以用 # ps -waux | grep sshd 来查看sshd是否已经被正确的运行了）如果不是通过inetd启动的SSH，那么SSH就将一直等待连接请求。当请求到来的时候SSH守护进程会产生一个子进程，该子进程进行这次的连接处理[4]  。

## 应用

[编辑](javascript:;)

**SSH另类应用：用ssh做**[socks5](https://baike.baidu.com/item/socks5" \t "_blank)代理[5]

1. 下载MyEntunnel。

2.下载PuTTY，解压到MyEntunnel程序的目录下。

3.运行MyEntunnel.exe，设置：SSH Server里头填上ssh ftp的地址或IP，填好用户名和密码，点Connect，系统栏里面的小锁变成绿色就连接成功了。

4.设置浏览器。IE是不支持socks代理的，用[firefox](https://baike.baidu.com/item/firefox" \t "_blank)好了，打开firefox的代理设置页，在socks主机处填上[127.0.0.1](https://baike.baidu.com/item/127.0.0.1" \t "_blank)端口填原先设置的，默认7070。

**Win主机环境运行SSH命令的方法**

对于Win主机用户，可以[下载工具](https://baike.baidu.com/item/%E4%B8%8B%E8%BD%BD%E5%B7%A5%E5%85%B7" \t "_blank)putty来进行shell管理。具体的命令依赖于登录到远端主机所使用的系统和Shell。

一些常用的shell命令如下：

cd[目录名]转换路径

cd.. 返回上级目录

ls显示当前目录下所有文件

rm[-r]-f[][文件名]删除文件，加[-r]可以删除文件下所有子文件，如rm[ -r]-f [abc]删除abc文件夹及文件夹下的所有文件

tar -xzf [解压下载的压缩包]

unzip[文件名]解压文件

cp -rpf .A/\* B 将A文件夹中的所有文件拷贝到其上级目录B中

wget(远程下载文件到服务器上）

## 扩展

[编辑](javascript:;)

SSH协议框架中设计了大量可扩展的冗余能力，比如用户自定义算法、客户自定义密钥规则、高层扩展功能性应用协议。这些扩展大多遵循 IANA 的有关规定，特别是在重要的部分，像命名规则和消息编码方面。

SSH采用面向连接的[TCP](https://baike.baidu.com/item/TCP" \t "_blank)协议传输 应用22号端口 安全系数较高。

## 启动方法

[编辑](javascript:;)

**方法一：使用批处理文件**

在服务器端安装目录下有两个批处理文件“start-ssh.bat”和“stop-ssh.bat”。运行“start-ssh.bat”文件就可以启动SSH服务，要停止该服务只要执行“stop-ssh.bat”文件即可。

**方法二：使用SSH服务配置程序**

在安装目录下，运行“fsshconf.exe”程序，它虽是SSH服务器的配置程序，但也可以用来启动和停止SSH服务。在弹出的“F-Secure SSH Server Configuration”窗口中，点击左面列表框中的“Server Settings”后，在右边的“Service status”栏中会显示服务器状态按钮，如果服务器是停止状态，则按钮显示为“Start service”，点击该按钮就可启动SSH服务，再次点击可停止SSH服务。

**方法三：使用NET命令**

在服务器端的“命令提示符”窗口中，输入“net start ″F-secure SSH Server″”命令，就可以启动SSH服务，要停止该服务，输入“net stop ″F-Secure SSH Server″”命令即可。其中“F-Secure SSH Server”为SSH服务器名，“net start”和“net stop”为Windows系统启动和停止系统服务所使用的命令。

启动了SSH服务后，一定要关闭Telnet服务，这样服务器就处在安全环境之中了，不用再怕数据被窃取[6]  。

## SSH安全技巧

[编辑](javascript:;)

SSH 是较可靠，专为远程登录会话和其他网络服务提供安全性的协议。利用 SSH 协议可以有效防止远程管理过程中的信息泄露问题。S S H最初是U N I X系统上的一个程序，后来又迅速扩展到其他操作平台。S S H在正确使用时可弥补网络中的漏洞。客户端包含ssh程序以及像scp(远程拷贝)、slogin(远程登陆)、sftp(安全文件传输)等其他的应用程序。SSH有很多非常酷的特性，如果它是你每天的工作伴侣，那么我想你有必要了解以下10条高效使用SSH的秘籍，它们帮你节省的时间肯定会远远大于你用来配置它们的时间。

### 服务器间跳转

　　有些时候，你可能没法直接连接到某台服务器，而需要使用一台中间服务器进行中转，这个过程也可以自动化。首先确保你已经为服务器配置了公钥访问，并开启了agent forwarding，可以通过2条命令来连接目标服务器，不会有任何提示输入：  
　　$ ssh gateway  
　　gateway $ ssh db  
　　然后在你的本地SSH配置中，添加下面这条配置：  
　　Host db  
　　HostName  
　　ProxyCommand ssh gateway netcat -q 600 %h %p  
　　可以通过一条命令来直接连接目标服务器了:  
　　$ ssh db  
　　这里你可能会需要等待长一点的时间，因为SSH需要进行两次认证，注意netcat也有可能被写成nc或者ncat或者前面还需要加上g，你需要检查你的中间服务器来确定实际的参数。

### 省去用户名

　　如果你在远程服务器上的用户名和你本地的用户名不同，你同样可以在SSH配置中进行设置：  
　　Host www\* mail  
　　HostName %h  
　　User simon  
　　就算我的本地用户名是 smylers，我仍然可以这样连接我的服务器：  
　　$ ssh www2  
　　SSH会使用simon账户连接你的服务器，同样，Putty可以保存这个信息在你的session中。

### 主机别名

　　你也可以在你的SSH配置中直接定义主机别名，就像下面这样：  
　　Host dev  
　　HostName  
　　你还可以使用通配符来进行分组：  
　　Host dev intranet backup  
　　HostName %h  
　　Host www\* mail  
　　HostName %h  
　　在Putty中你可以为每个主机名保存单独的session，然后双击建立连接(但是它可能没办法支持通配符)。

### 省略主机名

输入服务器的完整主机名来建立一个新的SSH连接实在是太乏味无聊了，尤其是当你有一组拥有相同域名但是子域名不同的服务器需要管理时。  
　　或许你的网络已经配置了可以直接使用短域名，比如intranet，但是如果你的网络不支持，实际上你可以自己搞定这个问题，而不用求助网络管理员。  
　　解决办法根据你用的操作系统而略有差异，下面是我的Ubuntu系统的配置：  
　　prepend domain-search

然后你需要重启网络:$ sudo restart network-manager  
　　不同的系统，这两条命令可能会略有差异。

### 连接中转

有时候你可能需要从一个服务器连接另外一个服务器，比如在两个服务器之间直接传输数据，而不用通过本地电脑中转：  
　　www1 $ scp -pr templates www2:$PWD  
　　(顺便说一下，当你需要在两台服务器间拷贝文件时，$PWD变量时非常有用的)，因为即使你已经在两台服务器上添加了你本地电脑的公钥，scp默认仍然会提示你输入密码：这是因为你用来作为跳板的那台服务器上并没有你的私钥，所以，第二台服务器会拒绝你的公钥，但是一定不要通过将你的私钥拷贝到中转服务器上来解决这个问题，你可以使用agent forwarding来解决这个问题，只要在你的.ssh/config文件中加入下面这行代码就可以了：ForwardAgent yes或者是在Putty中勾上“Allow agent forwarding”选项，本地SSH就变成了第一台服务器的SSH代理，从第一台服务器在连接其它服务器就变和和在你本地一样简单，注意，如果要开启这个选项，前提是这个中间服务器值得你信任。

### 别再输入密码

如果你还在通过密码方式登录SSH，那么你或许应该试试SSH Keys，首先使用OpenSSH为自己生成一对密钥：  
　　$ ssh-keygen  
　　跟随指示，完成之后，你应该可以在你的.ssh目录下看到两个文件，id\_rsa就是你的私钥，而id\_ras.pub则是你的公钥，现　在你需要将你的公钥拷贝到服务器上，如果你的系统有ssh-copy-id命令，拷贝会很简单：  
　　$ ssh-copy-id

否则，你需要手动将你的公钥拷贝的服务器上的~/.ssh/authorized\_keys文件中：  
　　$ < ~/.ssh/id\_rsa.pub ssh ‘mkdir -p .ssh; cat >> .ssh/authorized\_keys; chmod go-w .ssh .ssh/authorized\_keys’  
　　试试重新连接到SSH服务器，或是拷贝文件，是不是已经不需要再输入密码了?

### 长连接

如果你发现自己每条需要连接同一个服务器无数次，那么长连接选项就是为你准备的：  
　　ControlPersist 4h  
　　你每次通过SSH与服务器建立连接之后，这条连接将被保持4个小时，即使在你退出服务器之后，这条连接依然可以重用，因此，在你下一次(4小时之内)登录服务器时，你会发现连接以闪电般的速度建立完成，这个选项对于通过scp拷贝多个文件提速尤其明显，因为你不在需要为每个文件做单独的认证了。

### 多条连接共享

如果你需要在多个窗口中打开到同一个服务器的连接，而不想每次都输入用户名，密码，或是等待连接建立，那么你可以配置SSH的连接共享选项，在本地打开你的SSH配置文件，通常它们位于~/.ssh/config，然后添加下面2行：  
　　ControlMaster auto  
　　ControlPath /tmp/ssh\_mux\_%h\_%p\_%r  
　　试试断开你与服务器的连接，并建立一条新连接，然后打开一个新窗口，再创建一条连接，你会发现，第二条连接几乎是在瞬间就建立好了。  
　　[Windows](https://baike.baidu.com/item/Windows)用户  
　　如果你是Windows用户，很不幸，最流行的开源SSH客户端Putty并不支持这个特性，但是Windows上也有OpenSSH的实现，比如这个Copssh，如果你觉得下面的一些技巧对你很有帮助，或许你应该试试Copssh。  
　　文件传输  
　　连接共享不止可以帮助你共享多个SSH连接，如果你需要通过SFTP与服务器传输文件，你会发现，它们使用的依然是同一条连接，如果你使用的Bash，你会发现，你甚至SSH甚至支持Tab对服务器端文件进行自动补全，共享连接选项对于那些需要借助SSH的工具，比如rsync，git等等也同样有效。

### 加速连接

　　如果你确保你和某个服务器之间的连接是安全的(比如通过公司内网连接)，那么你就可以通过选择arcfourencryption算法来让数据传输更快一些：  
　　Host dev  
　　Ciphers arcfour  
　　注意这个加速是以牺牲数据的“加密”性为代价的，所以如果你连接的是位于网上的服务器，千万不要打开这个选项，并且确保你是通过VPN建立的连接。

### 减少延迟

如果每次连接服务器都意味着你需要等待几十秒而无所事事，那么你或许应该试试在你的SSH配置中加入下面这条：  
　　GSSAPIAuthentication no  
　　如果这条命令有效的话，你应该通知你的系统管理员让他在服务器上禁用这个选项，这样其他人就不用再分别添加这条配置到它们的本地配置了。[5]