

# 网络安全协议及分析

## 会话安全SSH

密码与网络安全系 刘虹

2025年春季学期

# 课程体系

第一章 概述

第二章 链路层扩展L2TP

第三章 IP层安全IPSec

第四章 传输层安全SSL和TLS

第五章 会话安全SSH

第六章 代理安全Socks

第七章 网管安全SNMPv3

第八章 认证协议Kerberos

第九章 应用安全

# 本章学习目标

- ▲ **SSH连接协议**
- ▲ **SSH协议的典型应用**
- ▲ **SSH协议的相关案例**

# 提纲

**一、 SSH连接协议**

**二、 SSH应用**

**三、 SSH协议相关案例**

# 功能及组成

## ▲ SSH协议组成：

### ▪ 传输层协议：

- 协商和数据处理。位于SSH协议套件的最底层。位于SSH协议套件的最底层，为SSH其它协议和高层应用提供安全保护。（包含了服务器身份认证）

### ▪ 用户认证协议：

- 规定了服务器认证客户端用户身份的流程和报文内容。

### ▪ 连接协议：

- 将安全通道多路分解为多个逻辑通道，以便多个高层应用共享SSH提供的安全服务。

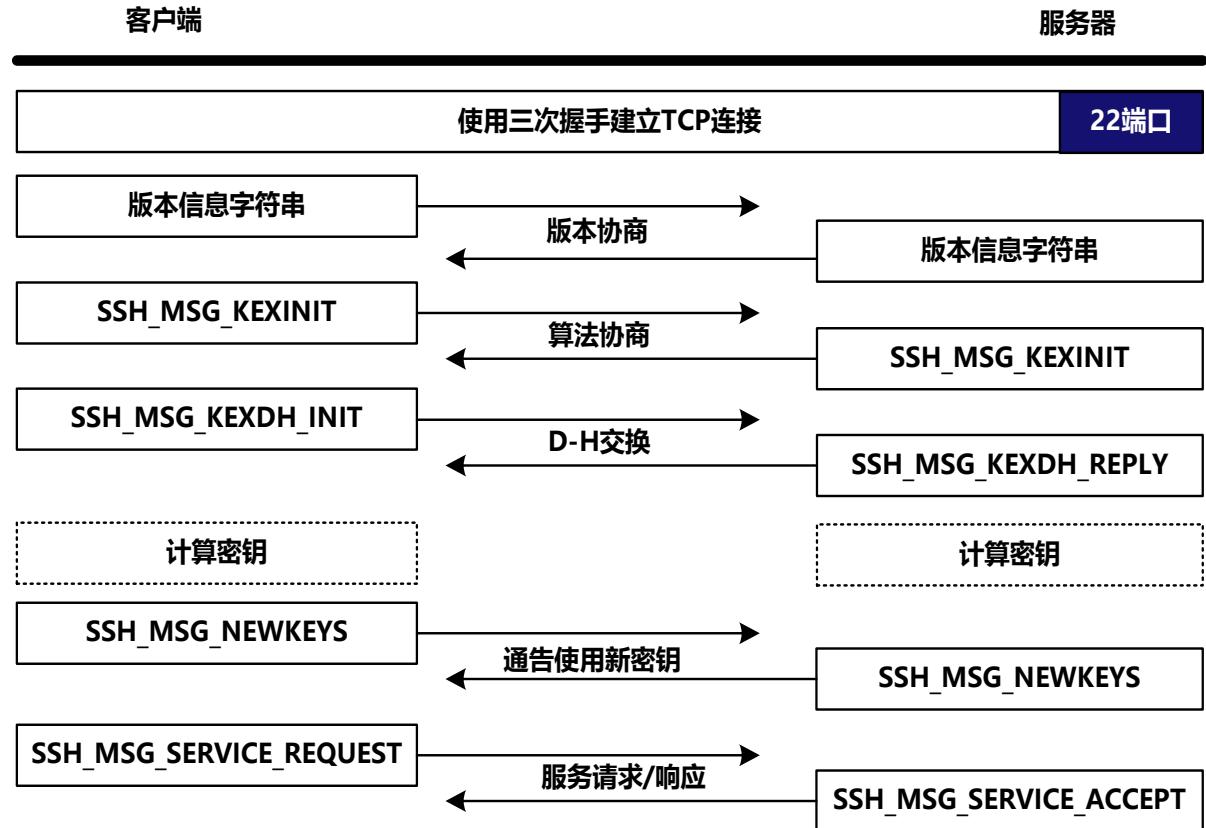
# SSH 传输层协议

## 版本交换

## 密钥交换

- 算法协商
- D-H 交换
- 密钥计算

## 服务请求/响应



# SSH身份认证协议

- 客户利用传输层协议向服务器提出**用户身份认证服务请求**，若服务器接受请求，则双方开始执行SSH身份认证协议。
  - 身份认证协议在传输层协议提供的安全通道上运行。（即身份认证协议的消息封装在传输层报文的数据区中）
  - 一次身份认证失败，客户端可再次提出认证请求，但重试的时间间隔和次数是有限制的。

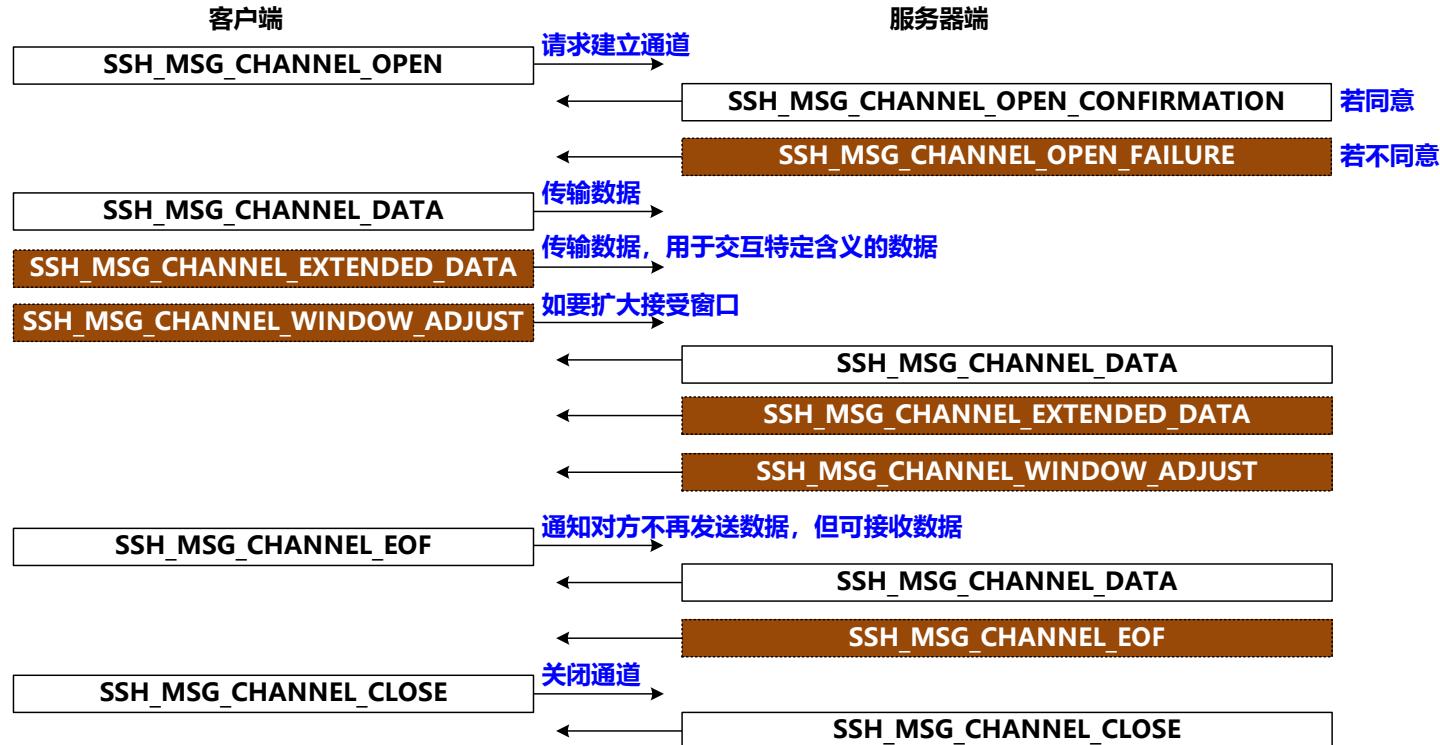
# SSH连接协议

- ▲ **连接协议如何运行：**
  - 在传输层提供的安全通道上进行
  - 客户端通过SSH传输层协议发出连接协议服务请求后，即可开始连接协议流程。
- ▲ **隧道：**传输层协议建立的安全通道称为“隧道”
- ▲ **通道：**所有的终端会话和被转发的连接称为“通道”
- ▲ **隧道和通道的关系：**一个隧道可以包含多条通道，即SSH的安全通道可以同时为多个不同的应用提供安全服务。

# SSH连接协议

- ▲ SSH通道可分为两类：
  - 交互式会话和TCP/IP端口转发，分别用于远程交互式会话和保护SMTP、IMAP等高层应用。
- ▲ SSH连接协议规定了四种通道类型：
  - session：运行远程程序
  - x11：x视窗系统，远程服务器运行，在本地显示
  - forwarded-tcpip：远程端口转发
  - direct-tcpip：本地端口转发

# 基本通道操作



# 基本通道操作

## SSH\_MSG\_CHANNEL\_OPEN消息格式

0

7 8

31

类型 (90)

通道类型... ...

发送方通道标识

初始窗口尺寸

最大报文长度

与通道类型相关的数据... ...

# 基本通道操作

## SSH\_MSG\_CHANNEL\_OPEN\_CONFIRMATION消息格式

0

7 8

31

类型 (91)

接收方通道标识

发送方通道标识

初始窗口尺寸

最大报文长度

与通道类型相关的数据... ...

# 基本通道操作

## SSH\_MSG\_CHANNEL\_OPEN\_FAILURE消息格式

0

7 8

31

类型 (92)

接收方通道标识

原因代码

描述信息 (ISO-10646 UTF-8编码形式)

语言标签

# 基本通道操作

## SSH\_MSG\_CHANNEL\_WINDOW\_ADJUST消息格式

0

7 8

31

类型 (93)

接收方通道标识

增加字节数

# 基本通道操作

## ◀ **SSH\_MSG\_CHANNEL\_DATA**消息格式

0

7 8

31

类型 (94)

接收方通道标识

数据... ...

# 基本通道操作

## SSH\_MSG\_CHANNEL\_EXTENDED\_DATA消息格式

0

7 8

31

类型 (95)

接收方通道标识

数据类型代码

数据... ...

# 基本通道操作

## SSH\_MSG\_CHANNEL\_EOF消息格式

- 通信方可发送通道文件末尾 (EndOfFile, EOF) 消息通知对方自己不再发送数据。

0

7 8

31

类型 (96)

接收方通道标识

# 基本通道操作

## SSH\_MSG\_CHANNEL\_CLOSE消息格式

- 通信的一方只有同时发送和收到了该消息，才能认为该通道已经被关闭，相应的通道标识也可以被重新使用。

0

7 8

31

类型 (97)

接收方通道标识

# 交互式会话通道

## 如利用通道交互式会话操作：

- `SSH_MSG_CHANNEL_OPEN`中指定通道类型为“session”
- 通道建立后，发送`SSH_MSG_CHANNEL_REQUEST`消息进行各种交互式操作。如该消息的标志为“TRUE”时需要对方应答，对方应返回以下消息之一：
  - `SSH_MSG_CHANNEL_SUCCESS`
  - `SSH_MSG_CHANNEL_FAILURE`

# 交互式会话通道

## SSH\_MSG\_CHANNEL\_REQUEST消息格式

0

7 8

31

类型 (98)

接收方通道标识

请求类型... ...

标志

与请求类型相关的数据... ...

# 交互式会话通道

## SSH\_MSG\_CHANNEL\_REQUEST消息格式中的“请求类型”

- |           |         |
|-----------|---------|
| ▪ 伪终端请求   | x11请求   |
| ▪ 发送环境变量  | 启动shell |
| ▪ 启动可执行命令 | 启动子系统   |
| ▪ 窗口更改    | 数据流控制   |
| ▪ 信号      | 退出状态    |
| ▪ 退出信号    |         |

# 交互式会话通道

## 伪终端请求

0 7 8

31

**类型 (98)**

接收方通道标识

请求类型... ... (" pty-req" )

**标志**

终端环境变量值

字符数表示的终端宽度

行数表示的终端高度

像素表示的终端宽度

像素表示的终端高度

编码的终端模式

# 交互式会话通道

## ◀ x11请求

- X Window系统：
  - 针对网络环境设计，所以采用了C/S模型。它的一个重要特点就是网络透明性，这意味着某台主机的使用者可以将网络中其他主机的程序执行结果显示在本机屏幕上。
- X11转发请求
  - 可以把远程的图形界面通过SSH的加密通道传送到本地
- X11通道转发
  - x11通道转发是所有交互式会话通道操作中最为特殊的一种

# 交互式会话通道

- 发送环境变量
  - 发送环境变量的标准请求类型字符串为“env”，与请求类型相关的数据部分则包括字符串类型的“变量名”和“变量值”字段。
- 启动shell
  - 在请求对方启动shell时，请求类型设置为“shell”，与请求类型相关的数据部分为空。
- 启动可执行命令
  - 在请求对方执行某个命令时，请求类型设置为“exec”，与请求类型相关的数据部分包含了字符串形式的命令。

# 交互式会话通道

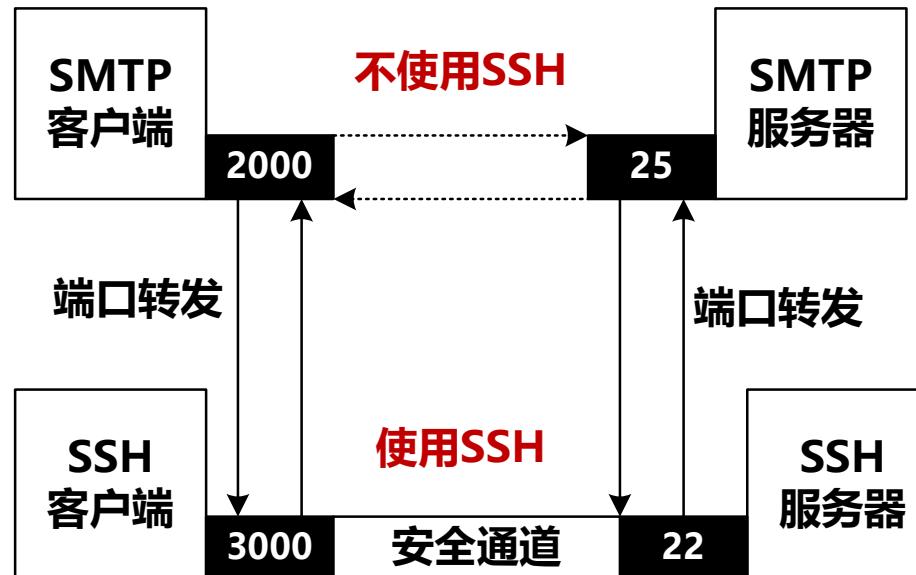
- ▲ 启动子系统
  - 在请求对方启动某个子系统时，请求类型设置为“subsystem”，与请求类型相关的数据部分则包含了字符串形式的子系统名。
- ▲ 窗口更改
  - 当客户端需要更改终端窗口尺寸时，将请求类型设置为“window-change”。
  - 与请求类型相关的数据部分则包括4个4B的新尺寸标识，即用列数描述的“终端宽度”、用行数描述的“终端高度”、用像素描述的宽度和高度信息。
- ▲ 数据流控制
  - 在交互式会话过程中，客户端可利用“control-s”（停止输出）和“control-Q”（启动输出）进行数据流控制，但必须得到服务器的认可。

# 交互式会话通道

- 信号
  - 当需要向远端发送信号时，可以发送类型为“signal”的通道请求，其中包含字符串形式的“信号名”字段。
- 返回退出状态
  - 远程命令执行完毕后，执行端会返回其执行状态。
- 退出信号
  - 上一个请求对应命令正常执行完毕的状态。当远程命令是由于收到某个信号而被强制终止，会返回“退出信息”请求。

# TCP/IP转发连接通道操作

- TCP/IP端口转发原理，用于保护SMTP、IMAP等应用安全



# TCP/IP转发连接通道操作

- ▲ TCP/IP端口转发流程
  - **direct-tcpip**:
    - 与通道类型相关的字段则分别包含了目标和发起方的信息。
    - 目标地址和发起方地址可以是IP地址，也可以是域名。
  - **forwarded-tcpip**:
    - 客户端可以在任意时刻向服务器发送一个“全局”请求消息SSH-MSG-GLOBAL-REQUEST以提出转发请求
- ▲ SSH连接协议同时支持IPv4和IPv6，一台主机也可能有多个网络接口，支持不同的协议族，或配置多个IP地址

# TCP/IP转发连接通道操作

- 客户端可以在任意时刻向服务器发送一个“全局”请求消息SSH-MSG-GLOBAL-REQUEST以提出转发请求。
- 该消息旨在告诉服务器：
  - 请在“拟绑定地址”的“拟绑定端口”监听，一旦在这个端口收到连接请求，请为每条连接开辟一个SSH通道。
  - 如果客户端将标志设置为TRUE且端口号设置为“0”，则服务器会建立这个安全通道并为其分配一个端口号，通过SSH\_MSG\_REQUEST\_SUCCESS返回。

# TCP/IP转发连接通道操作

- 当服务器收到远程主机连接至客户端的请求时，向客户端发送SSH-MSG-CHANNEL-OPEN消息。
- 该消息旨在告诉SSH客户端：
  - 刚才你让我在“发起方地址”和“发起方端口号”这里监听连接请求；现在来了一个连接请求，其信息由“连接地址”和“连接端口号”指示，我为他建立了一个安全通道。
  - 一旦SSH客户端同意建立这个通道，就可以在其中安全地传输数据了。

# 提纲

一、 SSH连接协议

二、 SSH应用

三、 SSH协议相关案例

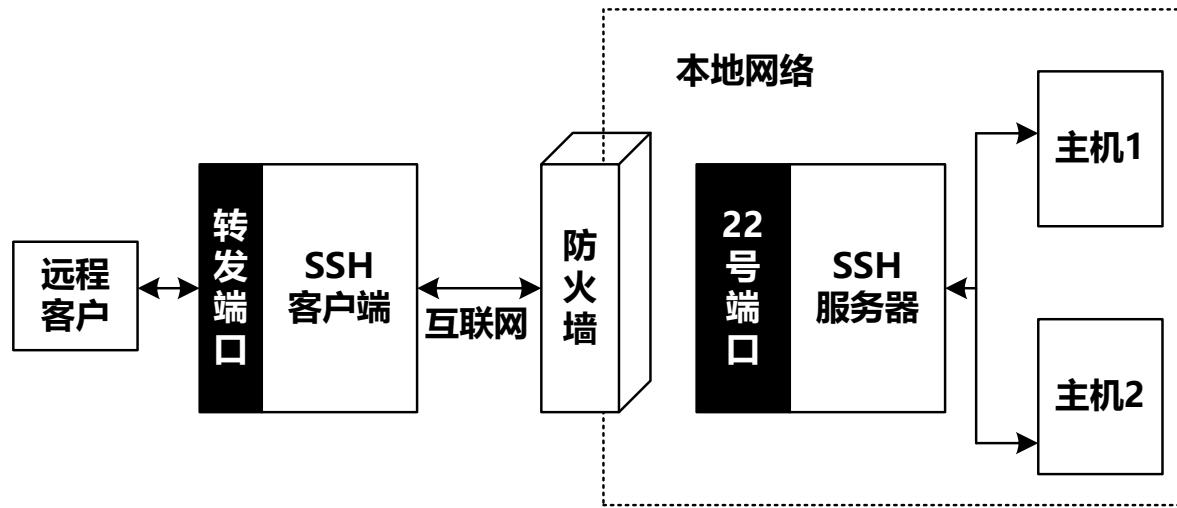
# SSH应用

- 用于安全的交互式会话或远程执行命令
  - **SSH文件传输协议SFTP**: 基于SSH2, 是SSH的子系统
    - 使用SSH\_MSG\_CHANNEL\_OPEN建立通道
    - 使用SSH\_MSG\_CHANNEL\_REQUEST消息请求启动SFTP子系统
    - SFTP报文作为SSH传输层报文的数据区进行传输
  - **FTPS**: 基于SSL的FTP
  - **SSH上的FTP**: 使用SSH1的TCP/IP端口转发
- 用于保护各类应用安全: 构建VPN

# SSH应用

## ▲ 基于SSH的VPN

- 使用SSH的TCP/IP端口转发功能构建VPN，使用该机制，防火墙的访问控制列表ACL可配置成仅允许目前端口为22的通信量通过。



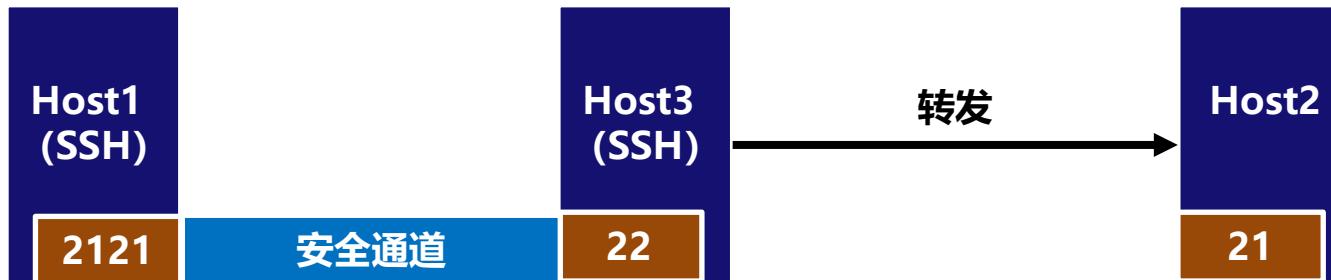
# SSH应用

- 基于SSH的VPN：基于SSH构建VPN，依托的是端口转发功能
  - 绑定端口：
    - 使用绑定端口，可以使得不加密的网络连接由SSH保护，从而提高安全性。比如，假设某台主机具备SSH模块，并且期望其所有8080号端口的数据都通过SSH传向远程主机，则可以设置SSH绑定8080号端口
  - 配置：`$ ssh -D 8080 user@host`

# SSH应用

## ▲ 基于SSH的VPN

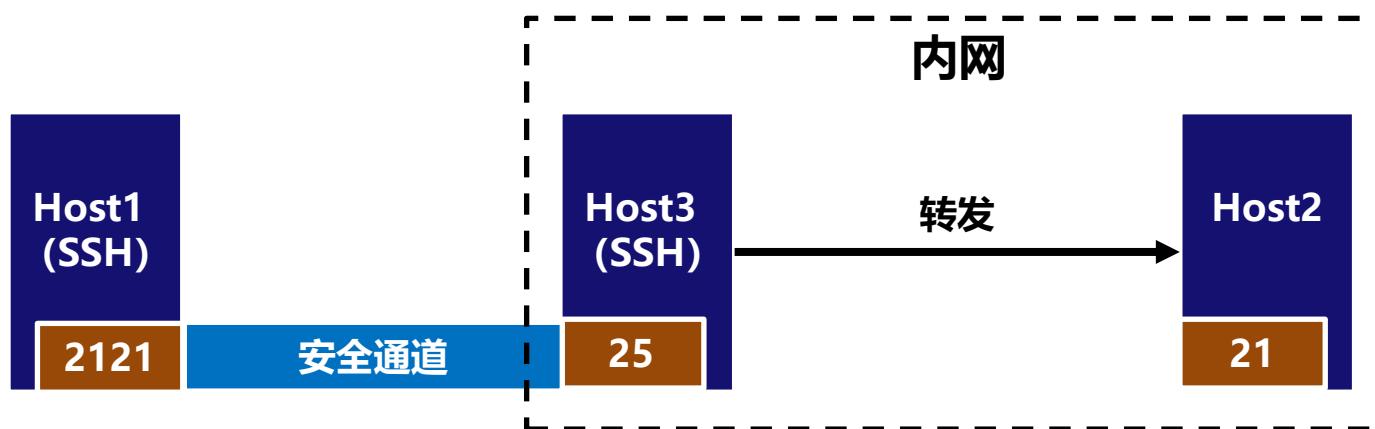
- 本地转发
- 配置: \$ssh-L 2121: host2: 21 host3



# SSH应用

## 基于SSH的VPN

- 远程转发
- 配置: \$ssh -R 2121: host2: 21 host1



# SSH应用

- **SSH隧道**: 隧道的概念是将一种网络协议封装在了另外一个协议里面。
  - 动态SSH隧道
  - 本地SSH隧道
  - 远程SSH隧道

# SSH应用

## ▲ SSH隧道

- 目的: 建立远程PC和不同网络的本地系统之间的SSH连接

SSH 服务端(两个网卡):

IP 192.168.1.104 和远程系统连接

IP 192.168.10.1 和本地网络192.168.10.2进行连接。

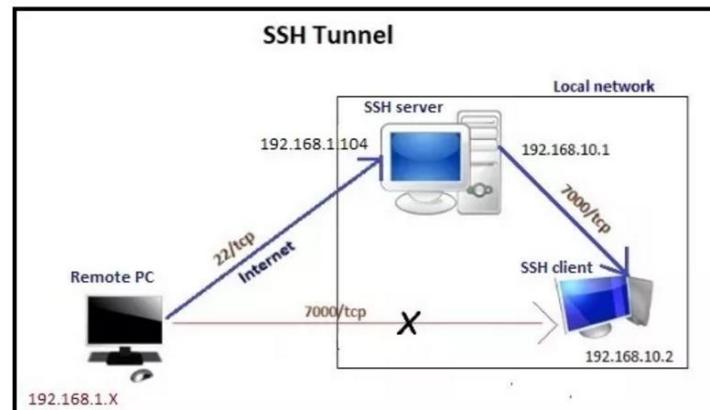
SSH 客户端:

IP 192.168.10.2远程系统(不在内网当中)

# SSH应用

## SSH隧道

- PC试图连接到不在同一网络的机器(192.168.10.2)。为了和SSH客户端建立连接，远程PC会通过SSH隧道与SSH客户端进行通信，进而访问本地网络，和SSH客户端连接。
- SSH服务是必须得启动的。



# SSH应用

## SSH隧道

- SSH服务端的IP设置，他有两个网卡，一个IP是192.168.1.104，另外一个是192.168.10.1。

```
raj@ubuntu:~$ ifconfig
ens33    Link encap:Ethernet HWaddr 00:0c:29:d7:e7:43
          inet addr:192.168.1.104 Bcast:192.168.1.255 Mask:255.255.255.0
          inet6 addr: fe80::836f:2737:911b:8a26/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:12658 errors:0 dropped:0 overruns:0 frame:0
            TX packets:118 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:10548981 (10.5 MB) TX bytes:14008 (14.0 KB)

ens38    Link encap:Ethernet HWaddr 00:0c:29:d7:e7:4d
          inet addr:192.168.10.1 Bcast:192.168.10.255 Mask:255.255.255.0
          inet6 addr: fe80::1266:d6b6:8e79:fb52/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:95 errors:0 dropped:0 overruns:0 frame:0
            TX packets:98 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:11912 (11.9 KB) TX bytes:11976 (11.9 KB)

lo      Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING MTU:65536 Metric:1
            RX packets:204 errors:0 dropped:0 overruns:0 frame:0
            TX packets:204 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:15028 (15.0 KB) TX bytes:15028 (15.0 KB)
```

# SSH应用

## SSH隧道

- SSH客户端的IP设置

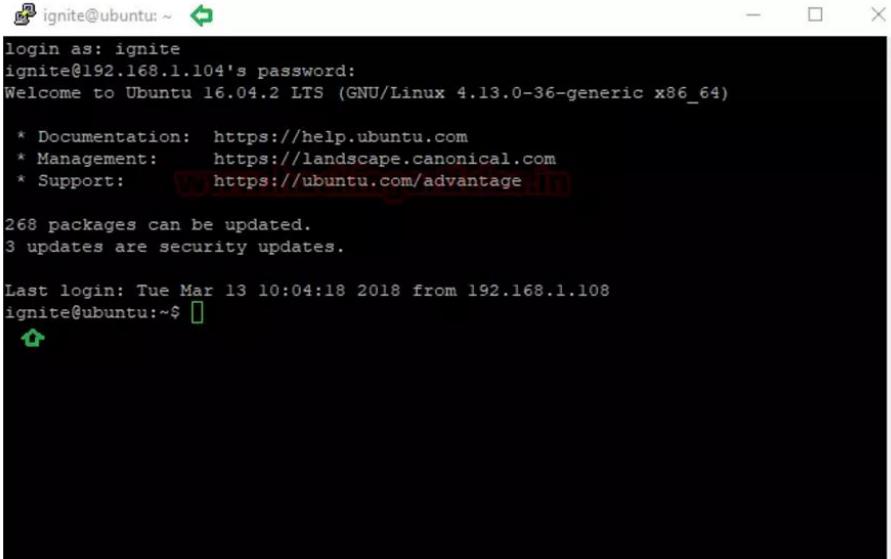
```
root@ignite:~# ifconfig ↵
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:1037 errors:0 dropped:1 overruns:0 frame:0
            TX packets:298 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:142045 (142.0 KB)  TX bytes:48788 (48.7 KB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:345 errors:0 dropped:0 overruns:0 frame:0
            TX packets:345 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:27024 (27.0 KB)  TX bytes:27024 (27.0 KB)
```

# SSH应用

## SSH隧道：通过windows进行动态SSH隧道建立

- 远程PC正在尝试通过22端口与SSH服务端进行连接，并且成功的登陆进去。
- 使用putty建立SSH服务端(ubuntu)和远程PC(Windows)的通信。



```
ignite@ubuntu: ~
ignite@192.168.1.104's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

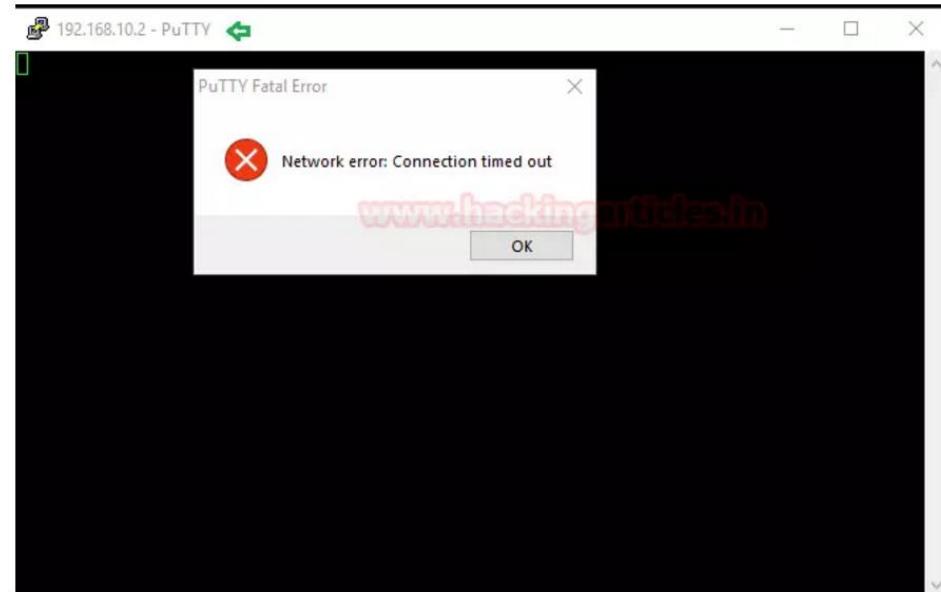
268 packages can be updated.
3 updates are security updates.

Last login: Tue Mar 13 10:04:18 2018 from 192.168.1.108
ignite@ubuntu:~$
```

# SSH应用

## SSH隧道

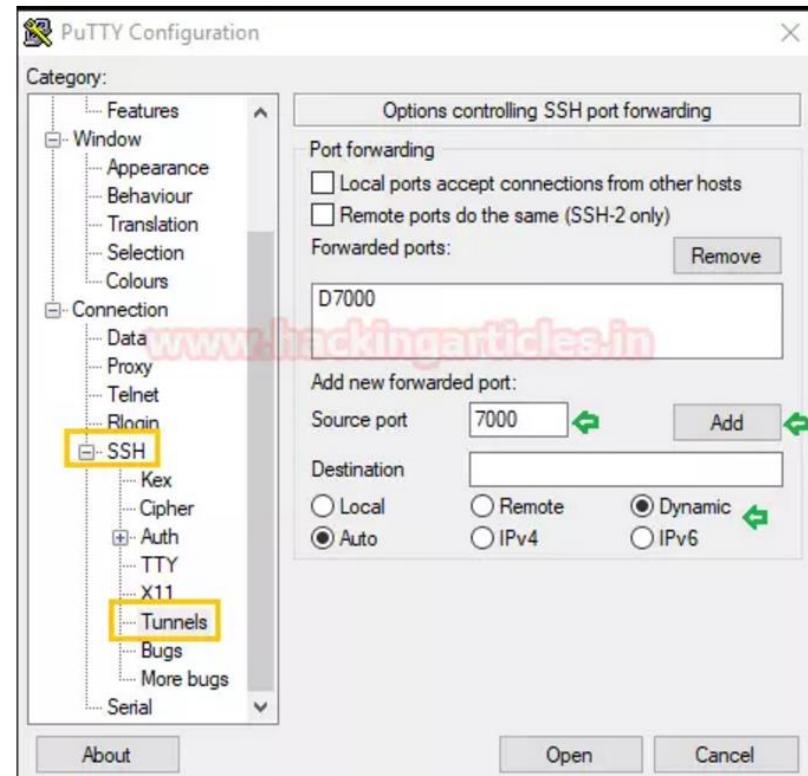
- 远程计算机使用相同的方法和SSH客户机进行连接，结果失败，原因是他们处在不同的网络当中。



# SSH应用

## SSH隧道：动态SSH隧道建立过程

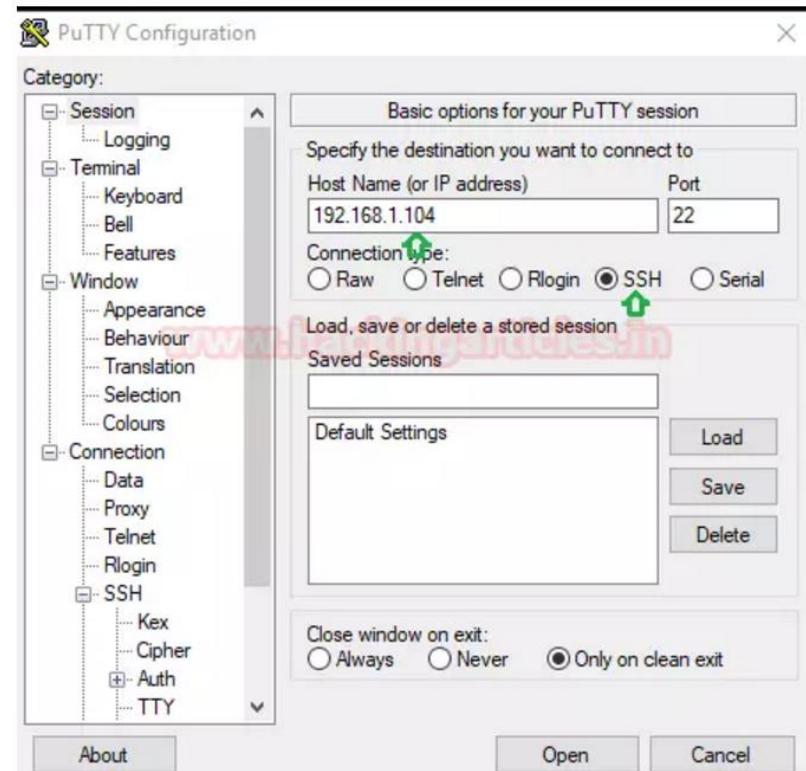
- 在左边框选择SSH->Tunnel项目，然后在源端口上填写7000，类型选择动态连接，然后点击添加。



# SSH应用

## SSH隧道

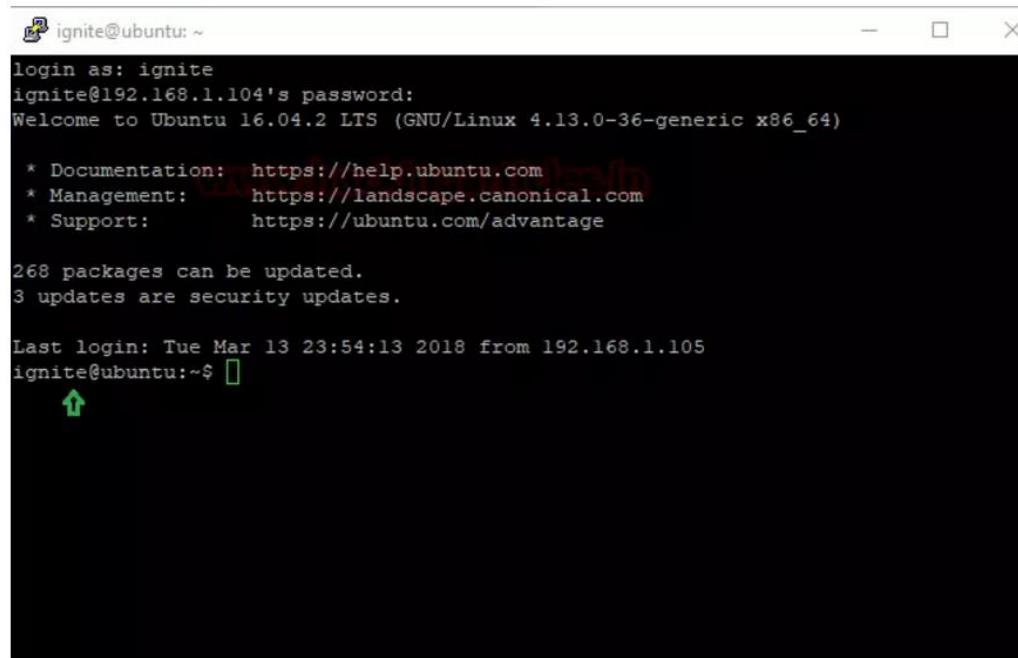
- 通过22端口连接192.168.1.104，填写完之后点击open：



# SSH应用

## SSH隧道

- 首先他会和SSH服务端进行连接



```
ignite@ubuntu: ~
login as: ignite
ignite@192.168.1.104's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

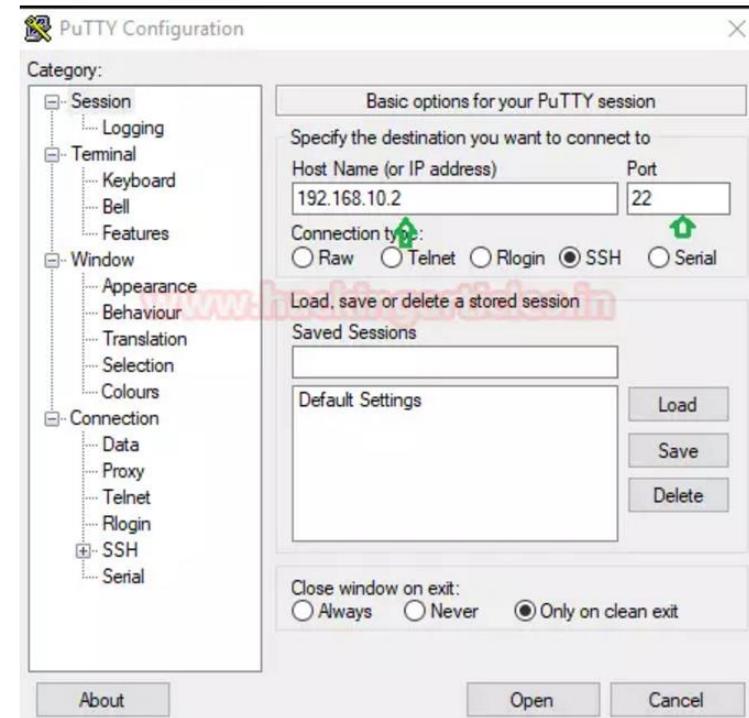
268 packages can be updated.
3 updates are security updates.

Last login: Tue Mar 13 23:54:13 2018 from 192.168.1.105
ignite@ubuntu:~$
```

# SSH应用

## SSH隧道

- 现在继续在putty中登陆，这是填写的就是SSH客户机的IP，以及22端口，填写完成之后，点击open：

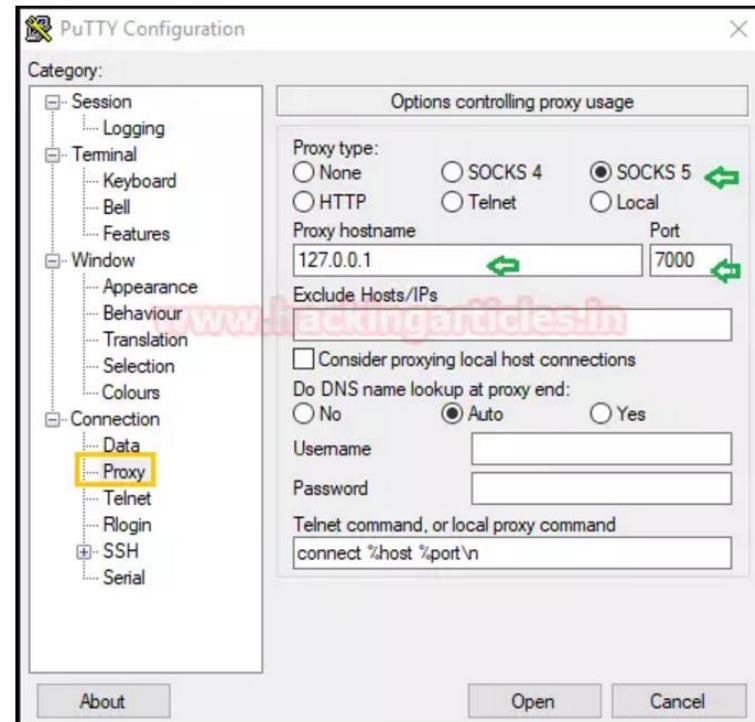


# SSH应用

## SSH隧道

- 打开之前putty运行的窗口，在Proxy(代理模块)进行如下操作：

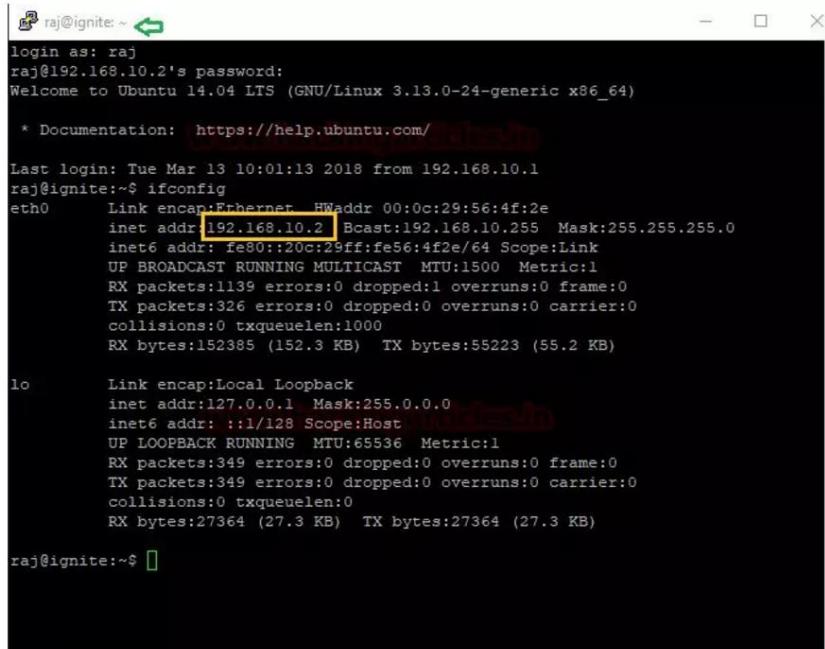
选择代理类型:SOCKS 5  
代理主机:127.0.0.1  
代理端口:7000  
点击建立连接



# SSH应用

## SSH隧道

- 通过7000端口与SSH客户机进行连接



```
raj@ignite: ~
login as: raj
raj@192.168.10.2's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation: https://help.ubuntu.com/

Last login: Tue Mar 13 10:01:13 2018 from 192.168.10.1
raj@ignite:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2 Bcast:192.168.10.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
              RX packets:1139 errors:0 dropped:1 overruns:0 frame:0
              TX packets:326 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:152385 (152.3 KB) TX bytes:55223 (55.2 KB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
              UP LOOPBACK RUNNING MTU:65536 Metric:1
              RX packets:349 errors:0 dropped:0 overruns:0 frame:0
              TX packets:349 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:0
              RX bytes:27364 (27.3 KB) TX bytes:27364 (27.3 KB)

raj@ignite:~$
```

# SSH应用

- SSH隧道：在80端口通过kali linux进行动态SSH隧道建立
  - kali linux用户如何通过SSH隧道与内网客户机进行连接的。
- 输入ssh -D 7000 ignite@192.168.1.104
- 输入完这条命令之后会提示你输入登录密码

```
root@kali:~# ssh -D 7000 ignite@192.168.1.104      ↵
ignite@192.168.1.104's password: [REDACTED]
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

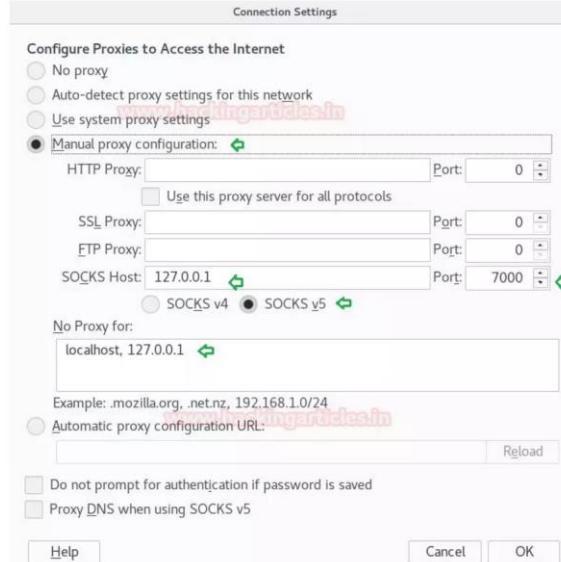
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

268 packages can be updated.
3 updates are security updates.

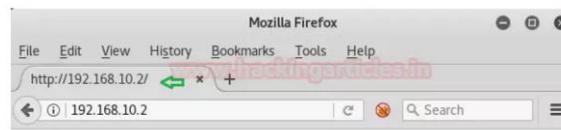
Last login: Tue Mar 13 23:57:45 2018 from 192.168.1.105
ignite@ubuntu:~$
```

# SSH应用

- SSH隧道：在80端口通过kali linux进行动态SSH隧道建立
  - 设置网络代理：
  - 打开浏览器中的网络代理板块，选择手动设置代理，然后开启SOCKS v5。



于是我们通过浏览器就可以访问到了内网机器192.168.10.2的80端口。



# SSH应用

- SSH隧道：在22端口上通过kali linux进行动态ssh隧道建立
  - 与SSH服务端进行连接

```
root@kali:~# ssh -D 7000 ignite@192.168.1.104 ↵
ignite@192.168.1.104's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

268 packages can be updated.
3 updates are security updates.

Last login: Tue Mar 13 23:57:45 2018 from 192.168.1.105
ignite@ubuntu:~$
```

# SSH应用

## SSH隧道：在22端口上通过kali linux进行动态ssh隧道建立

- 通过apt安装tsocks: apt install tsocks
- tsocks是用于拦截传出网络的连接，并且将所有连接都进行socks服务重定向的库。

```
root@kali:~# apt install tsocks
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gir1.2-networkmanager-1.0 gir1.2-nmgtk-1.0 gnome-themes-standard keepnote lib
  libfreerdp-gdi1.1 libfreerdp-locale1.1 libfreerdp-primitives1.1 libfreerdp-ut
  libpoppler68 libpoppler72 libproj12 libqgis-analysis2.14.21 libqgis-core2.14.
  libradare2-2.3 libtxc-dxtn-s2tc libvpx4 libwinpr-crt0.1 libwinpr-crypto0.1 li
  libwinpr-interlocked0.1 libwinpr-library0.1 libwinpr-path0.1 libwinpr-pool0.1
  libx264-148 multiarch-support php7.0-mysql python-functools32 python-httplib
Use 'apt autoremove' to remove them.
The following NEW packages will be installed:
  tsocks
```

# SSH应用

## SSH隧道：在22端口上通过kali linux进行动态ssh隧道建立

- 打开tsokcs.conf文件编辑 socks服务器的IP以及端口，在我们这种情况下，我们需要添加下面两行代码，然后保存就可以了。



# SSH应用

## SSH隧道：在22端口上通过kali linux进行动态ssh隧道建立

- 通过代理可以连接到SSH的客户机：  
`tsocks ssh raj@192.168.10.2`
- 输入密码，SSH隧道建立

```
root@kali:~# tsocks ssh raj@192.168.10.2 ↵
The authenticity of host '192.168.10.2 (192.168.10.2)' can't be established.
ECDSA key fingerprint is SHA256:JSfyM0DY2DlxXdaVStLUUx17WaTUIzqTKe0uKnCiSo.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.10.2' (ECDSA) to the list of known hosts.
raj@192.168.10.2's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation: https://help.ubuntu.com/

Last login: Wed Mar 14 00:49:44 2018 from 192.168.10.1
raj@ignite:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1675 errors:0 dropped:1 overruns:0 frame:0
          TX packets:582 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:215941 (215.9 KB)  TX bytes:97157 (97.1 KB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:373 errors:0 dropped:0 overruns:0 frame:0
          TX packets:373 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:29404 (29.4 KB)  TX bytes:29404 (29.4 KB)
```

# 提纲

一、 SSH连接协议

二、 SSH应用

三、 SSH协议相关案例

# SSH案例

## 自动配置SSH无密码访问的方法

- 由Linux bash语言编写测试程序
- 解决了ssh无密码访问配置的技术问题，实现了服务器集群系统在系统测试或运维测试中对多节点的无密码访问批量操作，无需记忆和输入每台服务器的密码，提高了远程操作效率。

# SSH案例

## 4 防止SSH破解的方法

- SSH蜜罐系统记录黑名单中的IP；核查该IP的所有密码，判断与本机的密码是否一致；如果一致，则立刻更改密码。
- 在不修改现有SSH服务的前提下，可以通过添加一层SSH蜜罐系统，将骇客引导进入蜜罐系统，并记录骇客的密码字典和IP地址，根据字典的变化进行判断，及时更新本机密码。

# 小结

- **SSH是一个应用层安全协议，使用知名端口号22。**
  - 它最初是设计用于保护会话安全，但最终也能够为其他各种应用提供安全服务。
- **SSH由传输层协议、用户认证协议和连接协议构成。**
  - 传输层是SSH的协商层和数据承载层，完成服务器身份认证、算法协商和密钥交换功能，同时定义了SSH报文格式；
  - 用户认证协议完成用户身份认证功能，支持口令认证、证书认证以及基于主机的认证这三种方法，单独规定的键盘交互式认证方法则更为通用；
  - 连接协议规定了利用SSH保护会话及应用安全的方法及流程。

# 小结

- **SSH使用D-H交换生成共享密钥，实际使用的4个会话密钥则由该共享密钥导出。**
  - **SSH使用的会话密钥包括用于客户发往服务器通信流的加密密钥和MAC密钥，以及反方向的加密和MAC密钥。**
- **SSH连接协议规定了“通道”的概念，并给出4种通道类型**
  - **session**用于交互式会话
  - **“x11”**用于转发x11连接
  - **“forwarded-tcpip”和“direct-tcpip”**用于TCP/IP端口转发。



办公地点：理科大楼B1209

联系方式：17621203829

邮箱：liuhongler@foxmail.com