**Lab6-report**

**Linux Firewall Exploration Lab**

**Name:范心宇**

**Student Number:57117129**

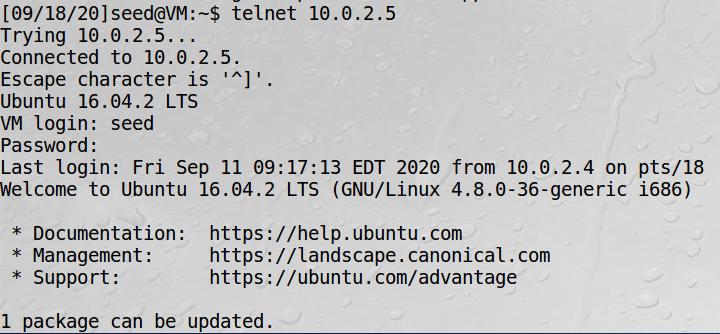
**Task 1: Using Firewall**

* **实验目的**

使用Linux系统防火墙。

* **实验过程**

1. 正常情况下虚拟机A与B进行telnet通信。



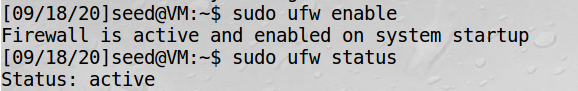
成功连接。

1. 修改ufw默认策略为allow。

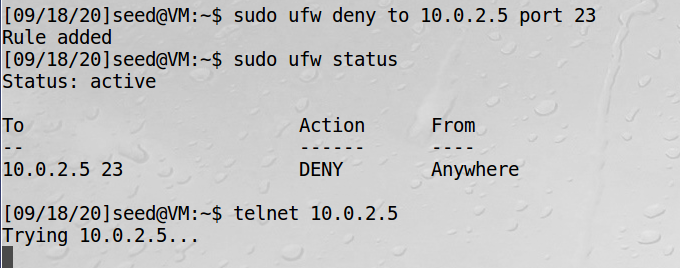


1. 打开ufw并设置规则。

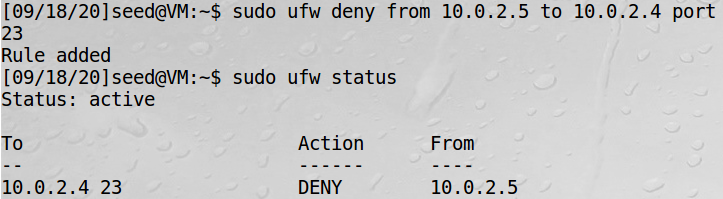
打开ufw：



阻止A向B发起telnet：



阻止B向A发起telnet：





阻止A访问一个外部网站：

可以看到现在已经无法访问了。

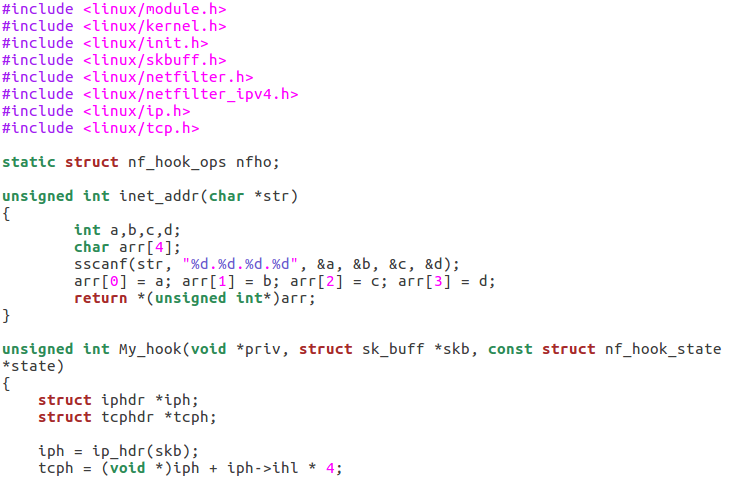
**Task 2: Implementing a Simple Firewall**

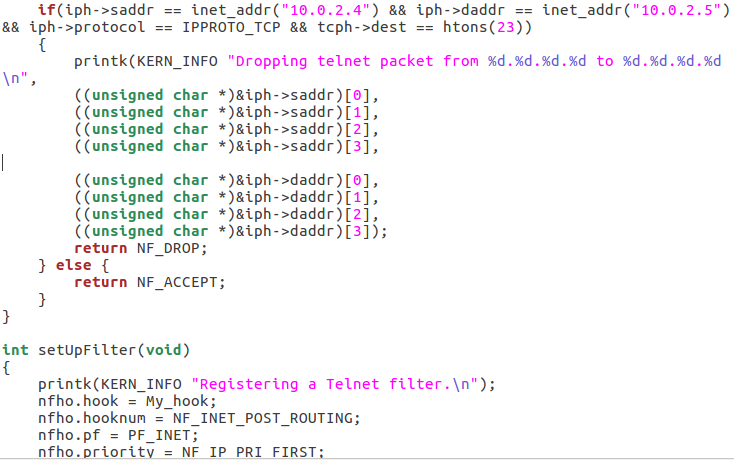
* **实验目的**

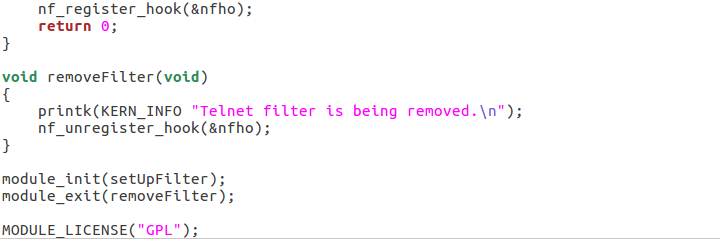
开发一个简单的防火墙。

* **实验过程**

1. 编写Netfilter代码。

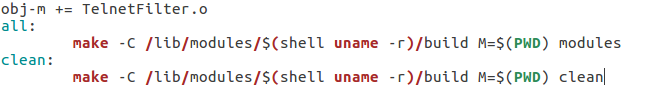




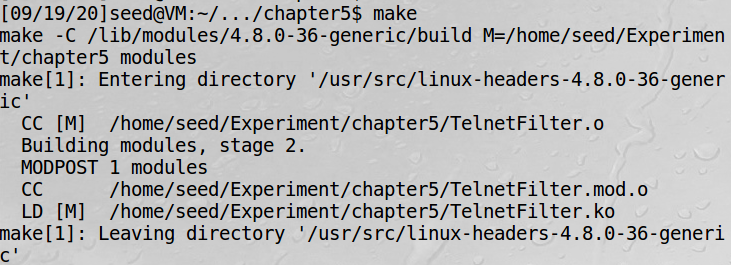


1. 编译动态可加载内核模块。

编写Makefile:



编译：



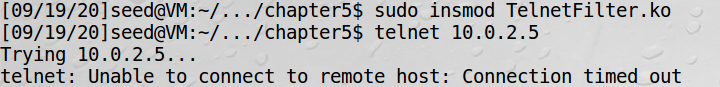
1. 运行代码观察结果。

阻止A向B发起telnet连接：

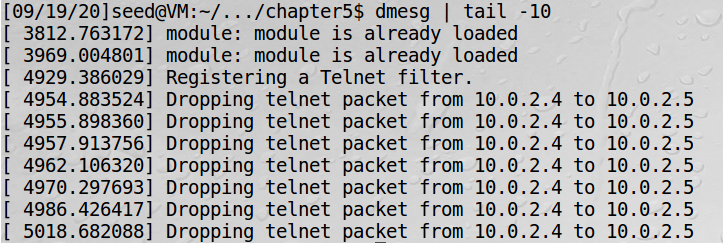




运行：



查看/var/log/syslog文件：



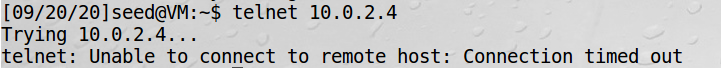
可以看到成功拦截了A发往B的telnet请求。

阻止B向A发起telnet连接：

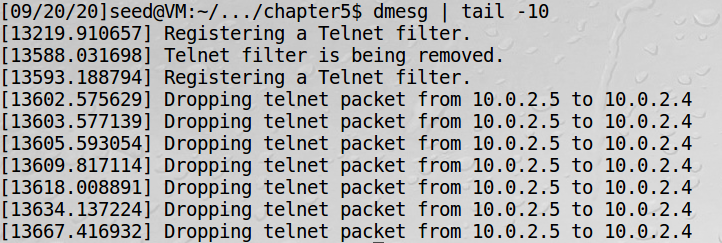




运行：



查看/var/log/syslog文件：



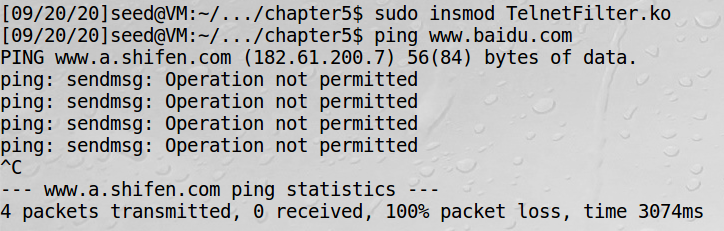
可以看到成功拦截了B发往A的telnet请求。

阻止A向外网www.baidu.com发起通信：





运行：



可以看到A已经无法访问www.baidu.com了。

**Task 3: Evading Egress Filtering**

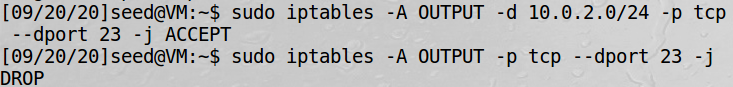
* **实验目的**

通过SSH建立隧道绕过外向防火墙。

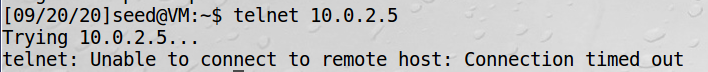
* **实验过程**

1. 使用防火墙关闭指定的对外通信。

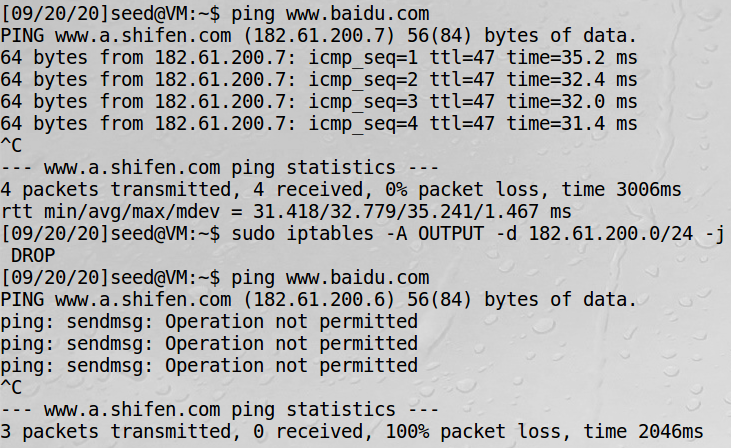
关闭所有对外telnet连接：



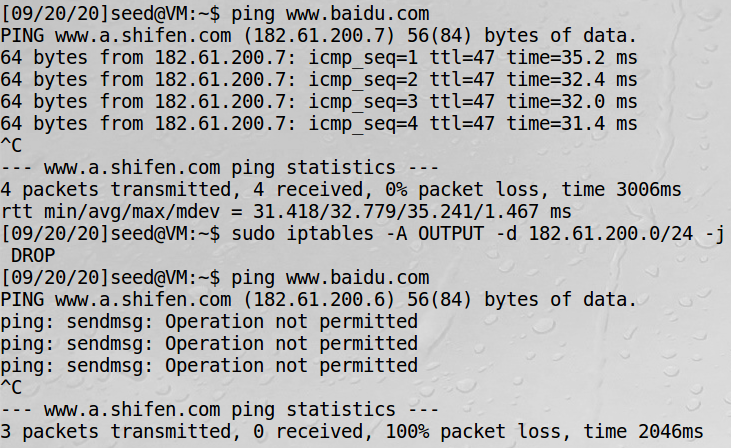
测试：



关闭对www.baidu.com的访问：



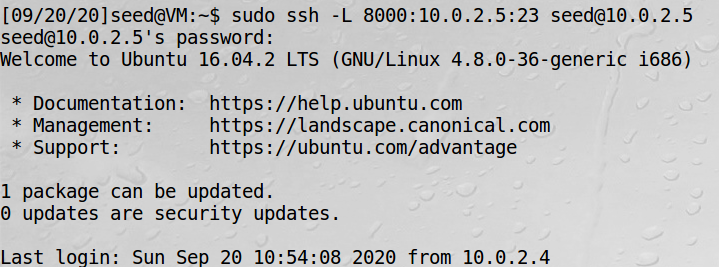
测试：



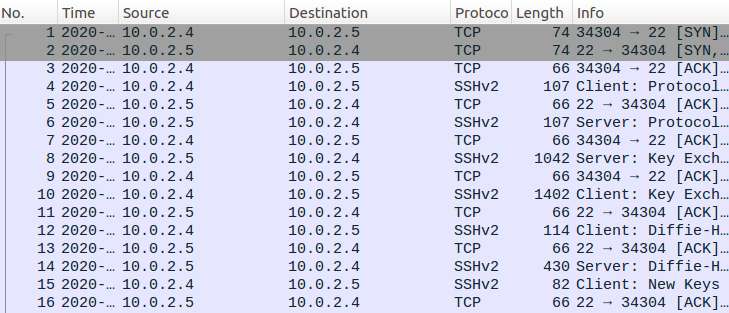
现在虚拟机A（10.0.2.4）已经无法对外发起telnet连接，也无法访问www.baidu.com了。

1. 通过SSH连接虚拟机B上的telnet服务。

使用SSH搭建加密通信信道：

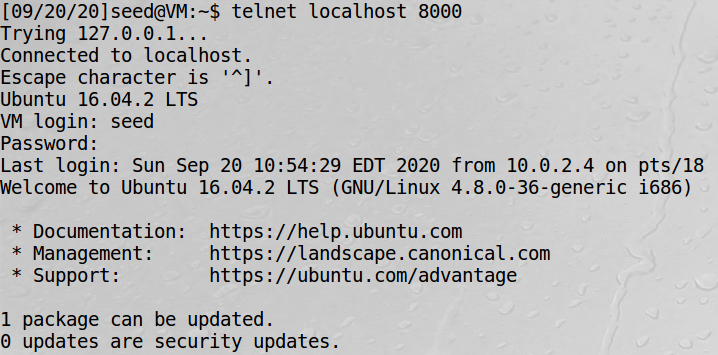


观察wireshark：

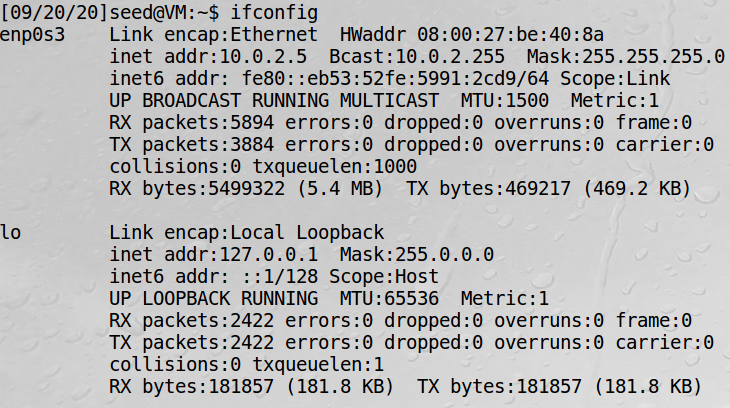


可以看到有大量的TCP与SSHv2报文。

测试telnet服务：

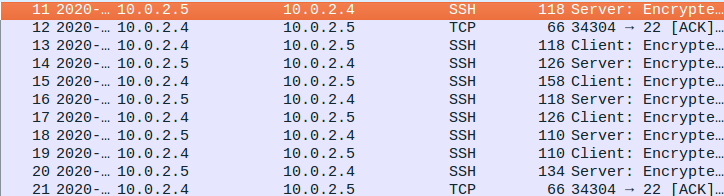


观察该telnet属性：



可以看到确实登录了虚拟机B（10.0.2.5）的telnet服务端。

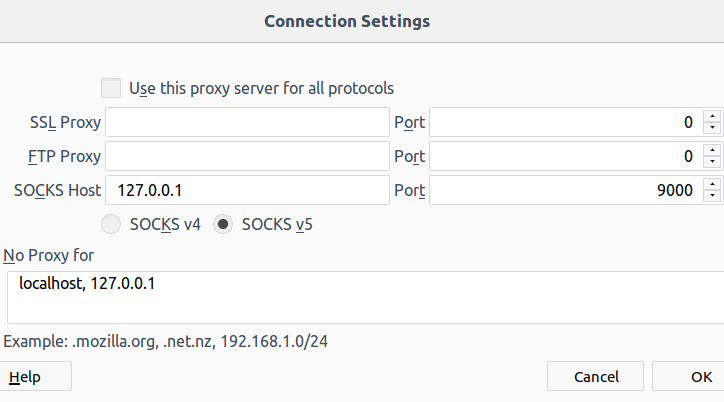
观察wireshark：



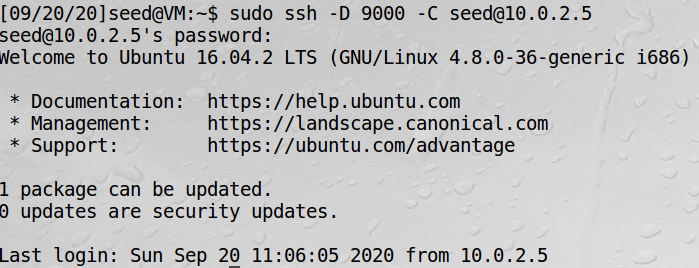
可以看到确实是10.0.2.4与10.0.2.5之间通过SSH进行加密通信。

1. 通过SSH访问www.baidu.com。

手动设置网络访问代理：



运行SSH建立隧道：

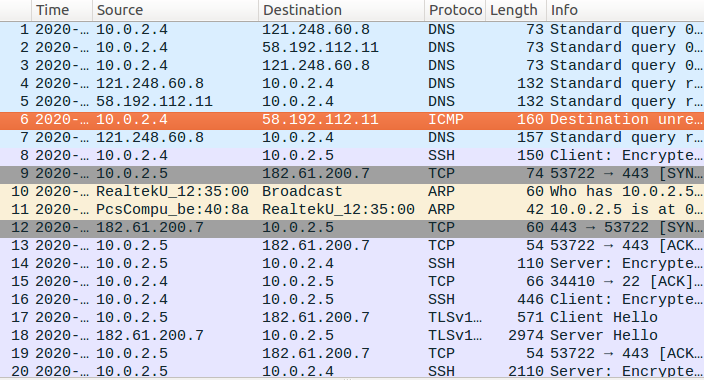


打开火狐浏览器，访问www.baidu.com：



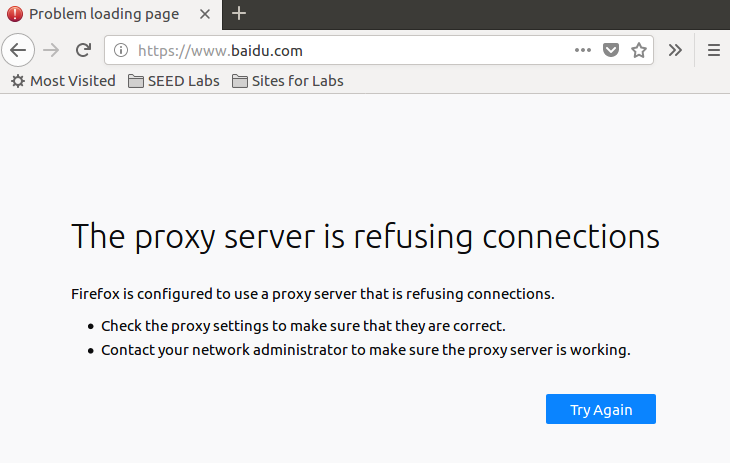
成功登录。

观察wireshark：



可以看到虚拟机A首先发出DNS查询，找到百度的IP地址，随后与虚拟机B通过SSH加密信道传输信息，而虚拟机B则与百度服务器使用TLS协议通信，虚拟机B就相当于虚拟机A与百度服务器通信的中转站。

退出SSH服务：



不再能够访问百度，代理服务器拒绝连接。

重新打开SSH服务：



再次成功访问。

* **实验结论**

可以看到，使用ssh可以绕过防火墙。这是因为防火墙只屏蔽了内部主机A发往外部特定端口或网址的数据包，导致主机A无法直接访问，但是主机A仍可以访问主机B的ssh端口。那么通过A与B之间建立的ssh隧道，将A要发送的数据包传给B，再由B与A的目的地址进行通信。也就是说数据包从A的应用软件发出，交给A自身的ssh客户端口，传递给B的ssh服务端口，再交付B与目的地址通信的客户端，作为该客户端的数据发送给目的服务器，接收数据则是相反流程。这样，就可以利用ssh加密隧道绕开防火墙对外部特定端口与网址的阻碍了。

**Task 4: Evading Ingress Filtering**

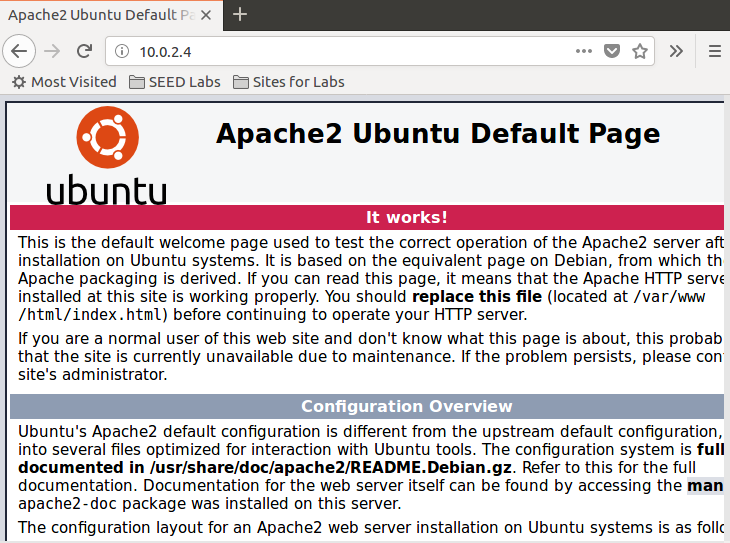
* **实验目的**

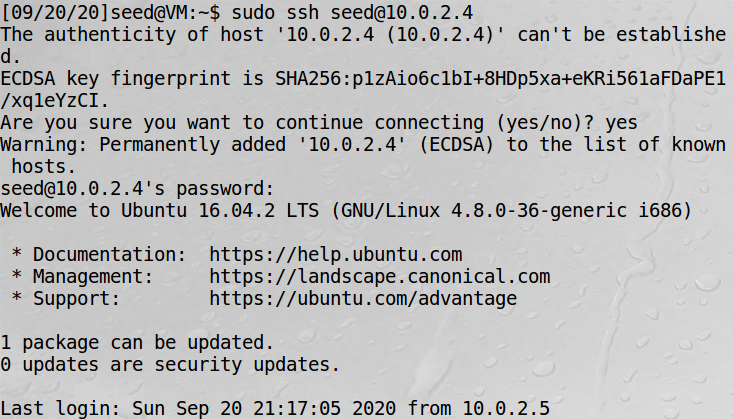
绕过内向防火墙。

* **实验过程**

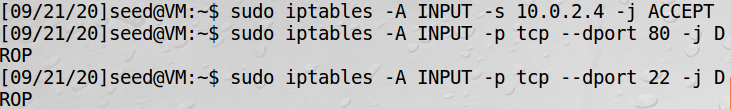
1. 设置虚拟机A的防火墙。

从虚拟机B上正常访问虚拟机A的80端口和22端口：



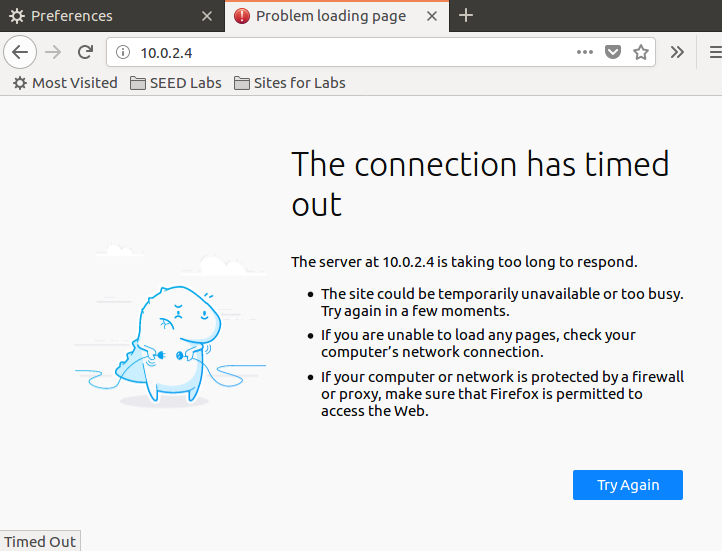


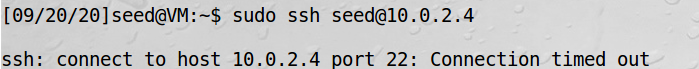
使用iptables防火墙禁止外部访问虚拟机A的80端口和22端口：



注意这里首先需要设置让虚拟机A自己能够访问本机。

在虚拟机B上进行测试：

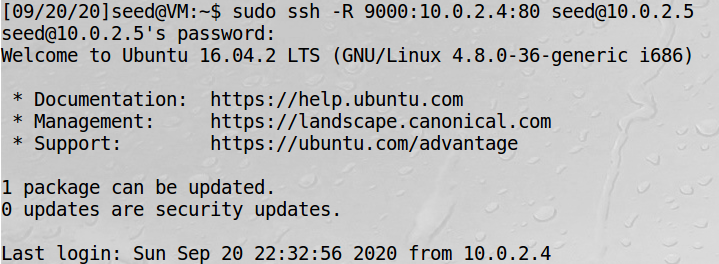




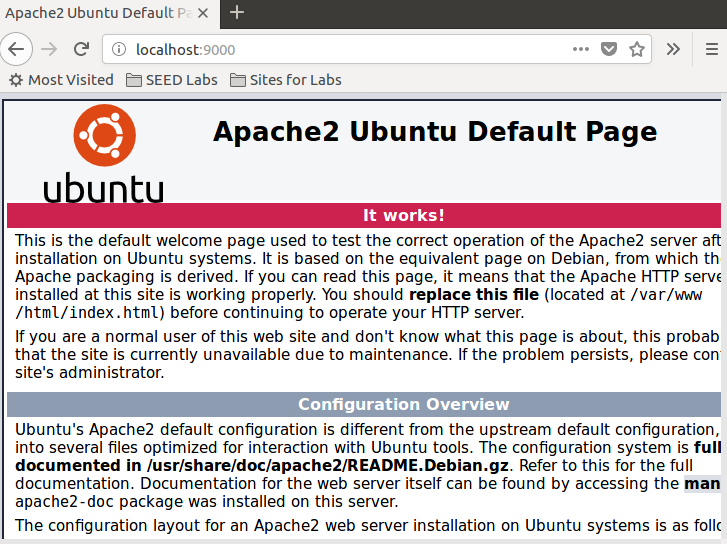
可以看到此时已无法连接虚拟机A的80和22端口。

1. 在虚拟机A上建立反向ssh。

开启80端口的反向ssh：

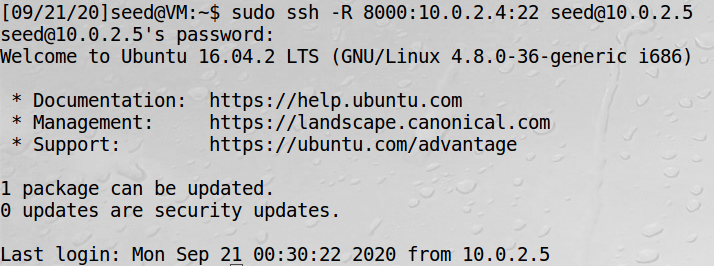


访问本机的9000端口：



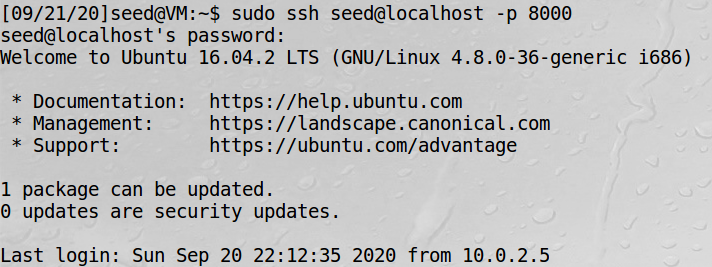
可以看到成功登录了虚拟机A的80端口。

开启22端口的反向ssh：

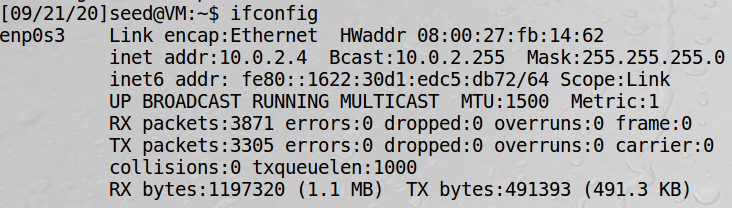


将虚拟机B上8000端口映射到虚拟机A上22端口。

在虚拟机B上使用8000端口登录自身ssh服务：



查看登录信息：



可以看到登录的是虚拟机A的ssh服务。

* **实验结论**

可以看到通过反向ssh可以绕过内向防火墙。这是因为虚拟机A以客户端的身份向虚拟机B发起了ssh通信，建立了一条ssh隧道，并使用反向ssh服务，定义虚拟机B上的本地连接端口，向该端口发起的通信都将传递给虚拟机B上的ssh服务器端口，ssh服务器将通信流量传递给虚拟机A上的ssh客户端，同时定义虚拟机A上的转发端口，ssh客户端会将通信传递给该端口。即通信从虚拟机B上的本地连接端口传递给ssh服务器端口，通过反向ssh连接传给虚拟机A上的ssh客户端口，再由ssh客户端传递给转发端口，将数据转发给内网的任意地址。