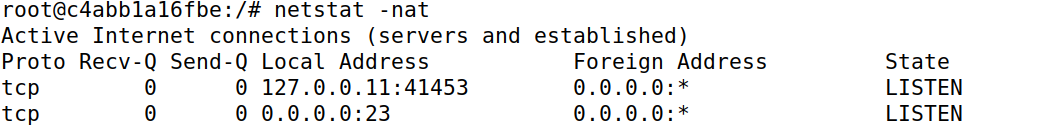
Lab2

57118111王雯

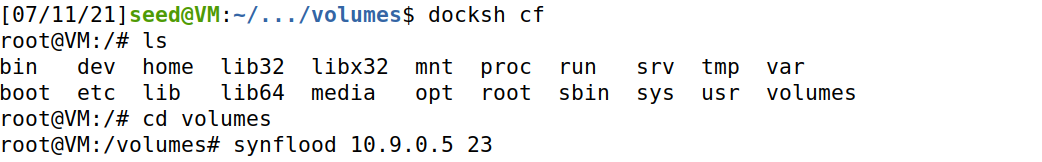
Task1： SYN Flooding Attack

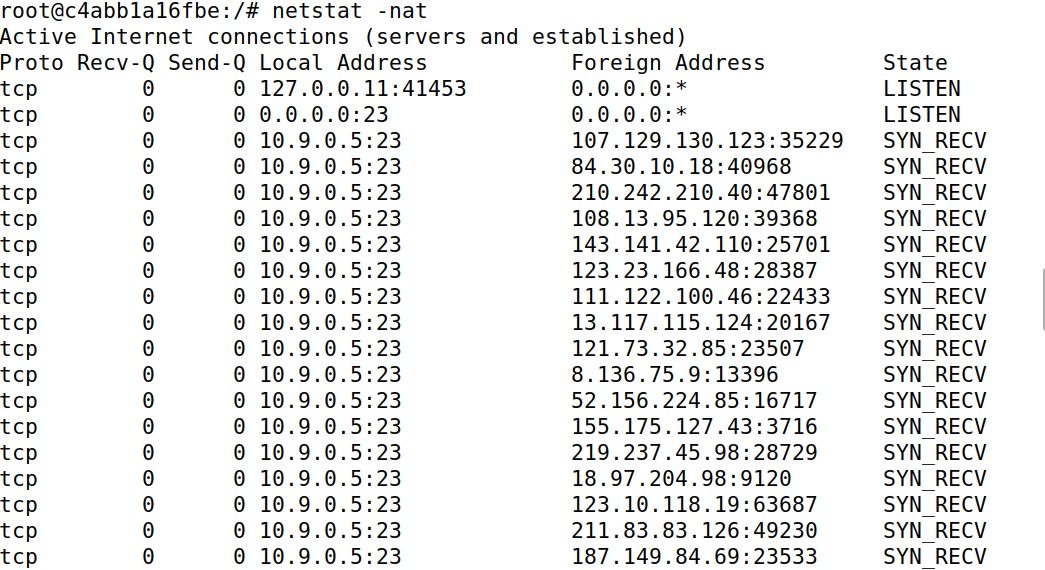
1. 对目标主机进行SYN泛洪攻击

攻击开始前查看受害者主机tcp状态，可以发现目前只有两个listen 状态。

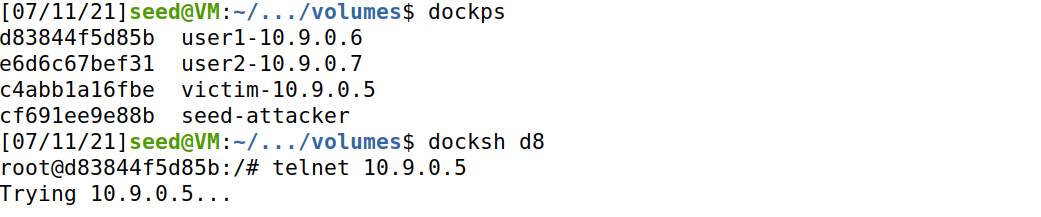


在攻击者主机的volumes下编译并执行synflood.c 程序，对被害者的23 号端口进行SYN 泛洪攻击，之后在受害者的主机里查看tcp连接状态：



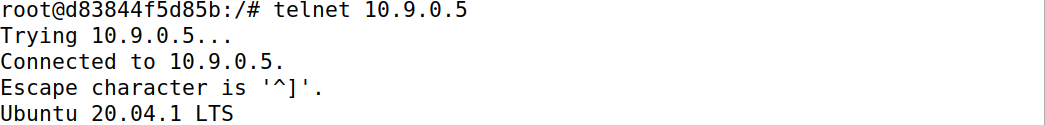


发现有大量的SYN-RECV 状态，说明受害者主机已经被进行SYN 泛洪攻击。在use1主机中Telnet 受害者主机，发现无法连接：

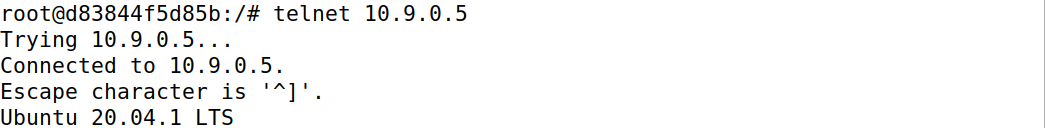


（2）攻击前进行Telnet连接

如果在攻击者主机使用ctrl+c停止攻击，发现再次在user1主机中Telnet 受害者主机，发现Telnet 连接成功。



如果Telnet 连接成功一次后再次进行SYN 泛洪攻击，然后再次Telnet 受害者主机，发现还是能连接成功。



说明被攻击主机已经记住了过去成功的连接，当再次连接时，不会受到攻击的影响。我们可以在受害者主机上执行查看保存的连接信息的命令：



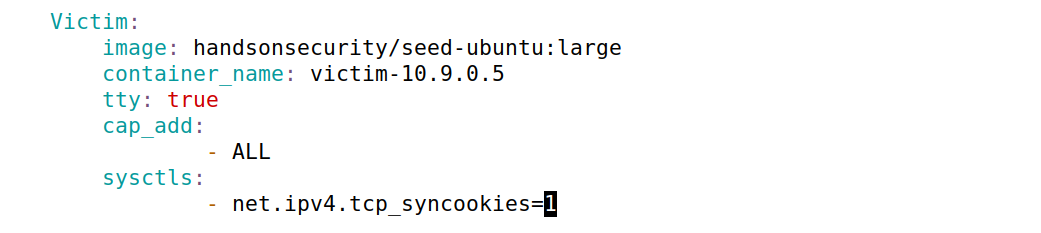
在清除保存的连接信息后，再次在攻击进行时进行Telnet连接时又连接失败了。这是因为以前成功的连接信息被清除了，所以该连接又会受到攻击的影响了。

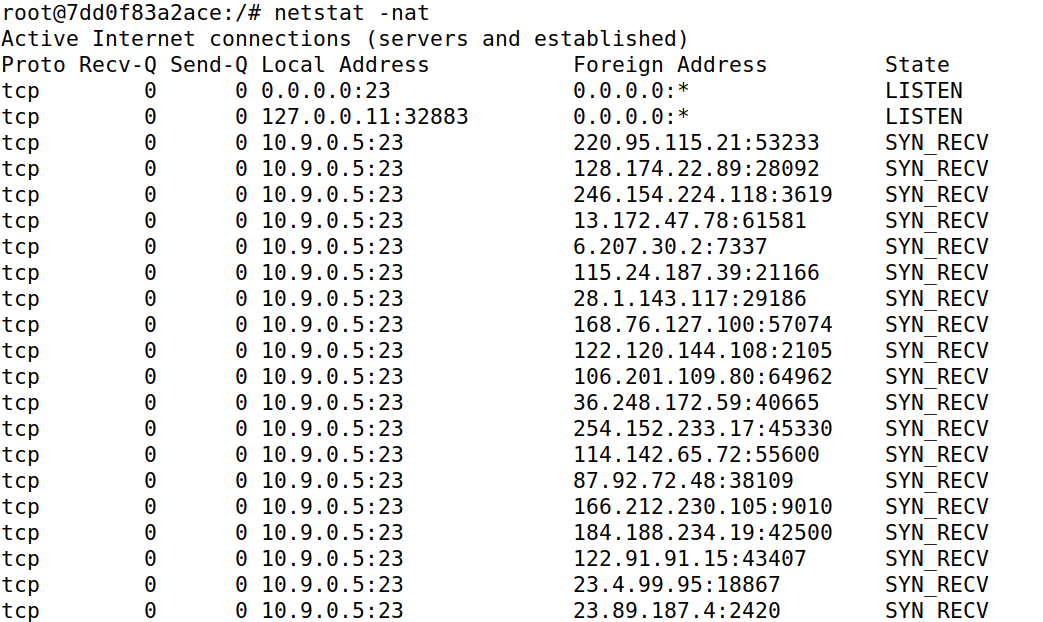


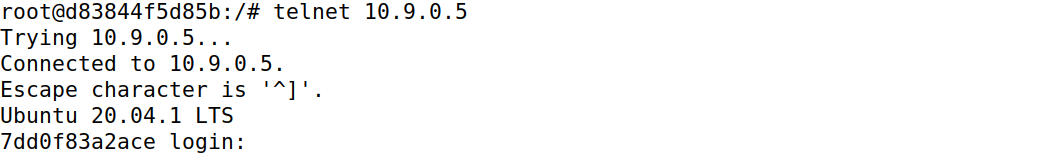


（3）启动SYN cookie 机制

在labsetup的终端中，进入docker-compose.yml 文件中修改为net.ipv4.tcp\_syncookies=1，启用SYN cookie机制。需要注意的是这里需要重启容器。再次进行攻击和Telnet 连接：





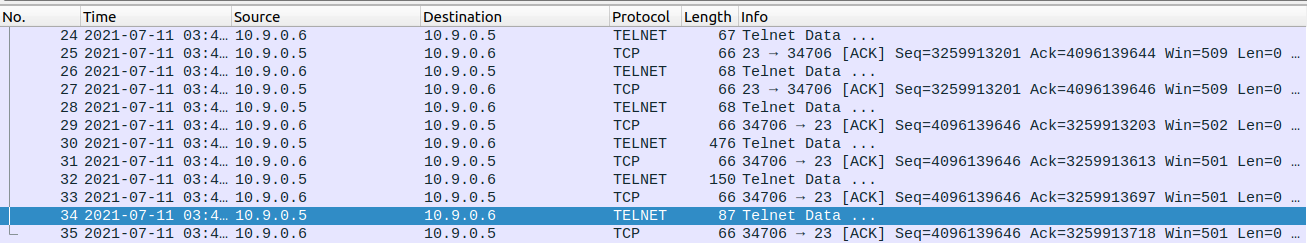


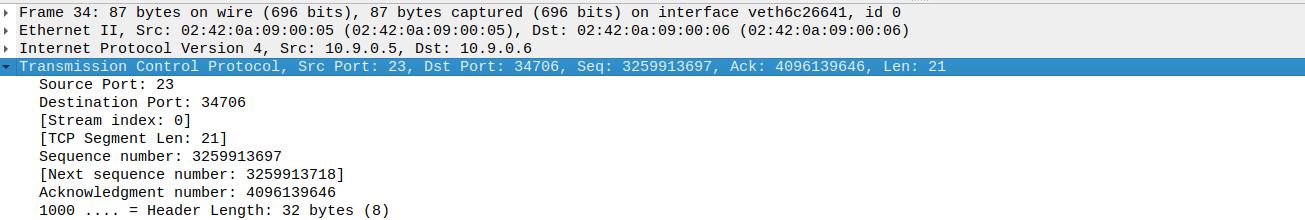
发现此时虽然被攻击主机仍然显示有许多SYN-RECV 状态连接，但是其他主机

依旧可以成功Telnet 连接到被攻击主机。说明SYN cookie 机制成功抵抗了泛洪攻击。

Task2：TCP RST Attacks on telnet Connections

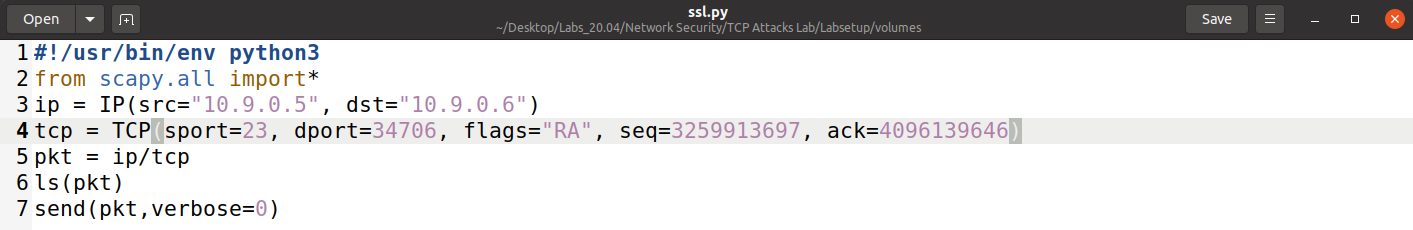
在user1主机中对受害者主机进行Telnet 连接，用wireshark 抓包，观察最后一个Telnet报文



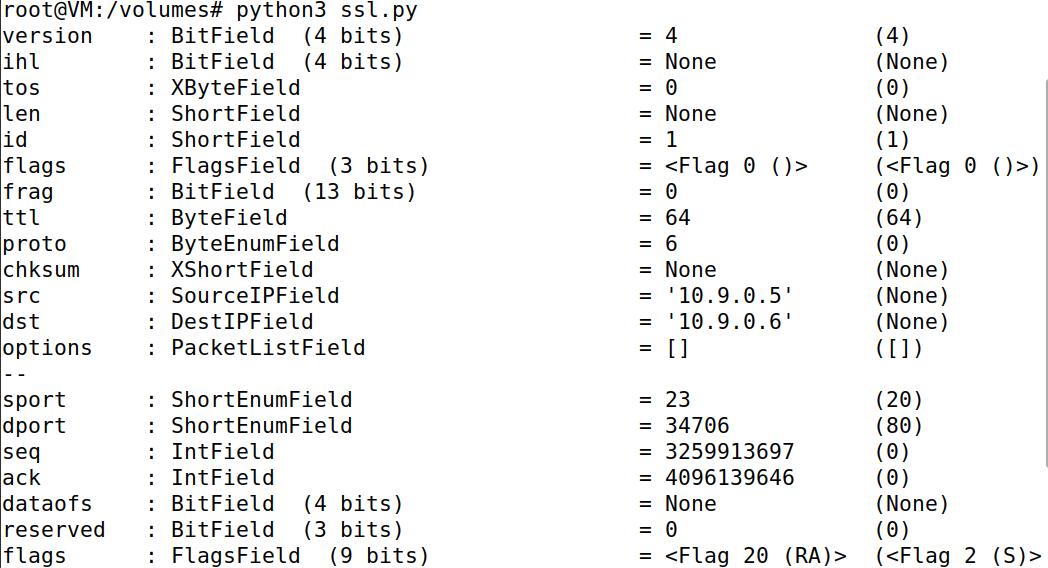


源IP 地址为10.9.0.5，目的地址为10.9.0.6，源端口23，目的端口34706，

seq 值为3259913697，ack 为4096139646，构造攻击数据包，代码如下：



攻击方执行代码后可以发现Telnet 连接被中断了。

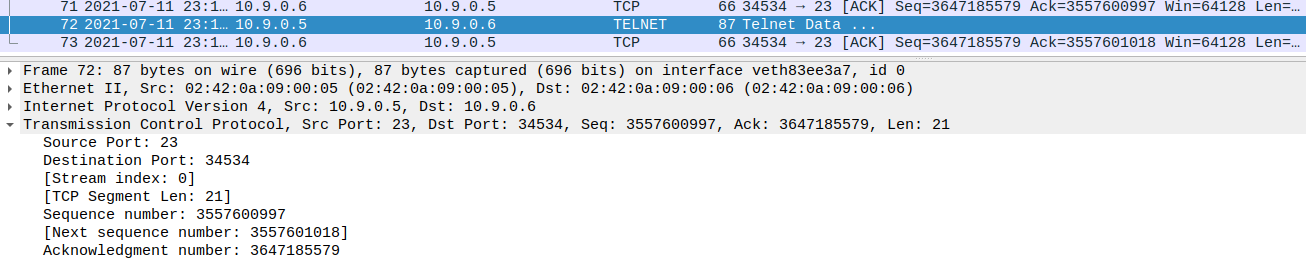


说明攻击成功：

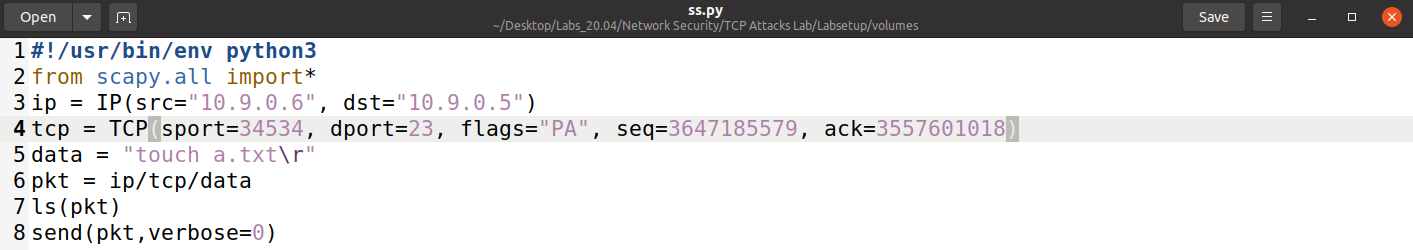


Task3:TCP Session Hijacking

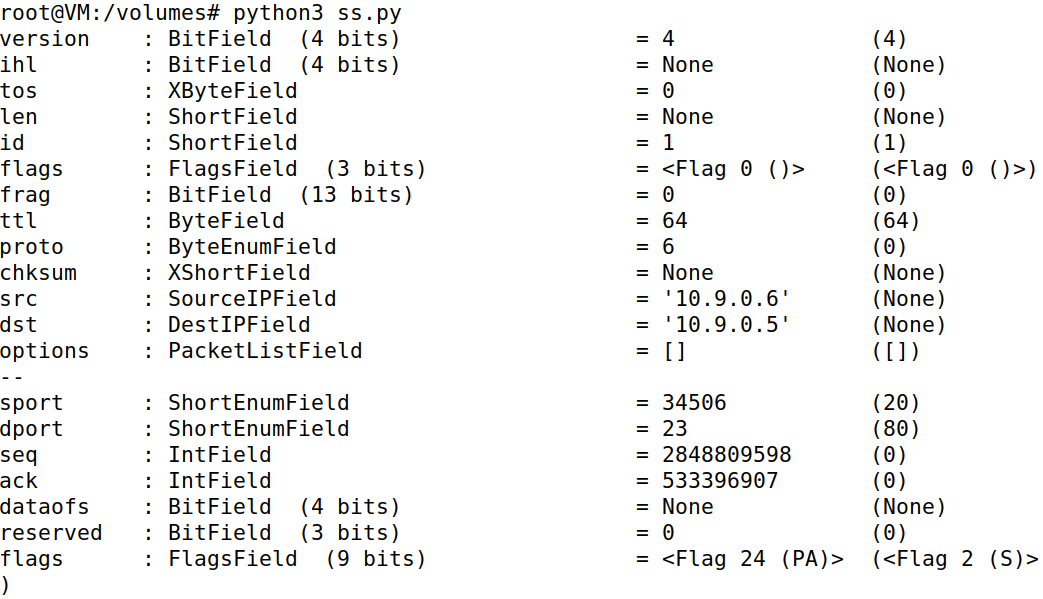
在user1主机中对受害者主机进行Telnet 连接，用wireshark抓包，观察最后一个Telnet报文



代码如下，代码中我们执行touch a.txt 命令来，最后可以通过观察被攻击主机中是否生成了a.txt 文件来判断命令是否正确执行，即判断TCP 会话劫持是否成功。



在攻击者主机运行代码：



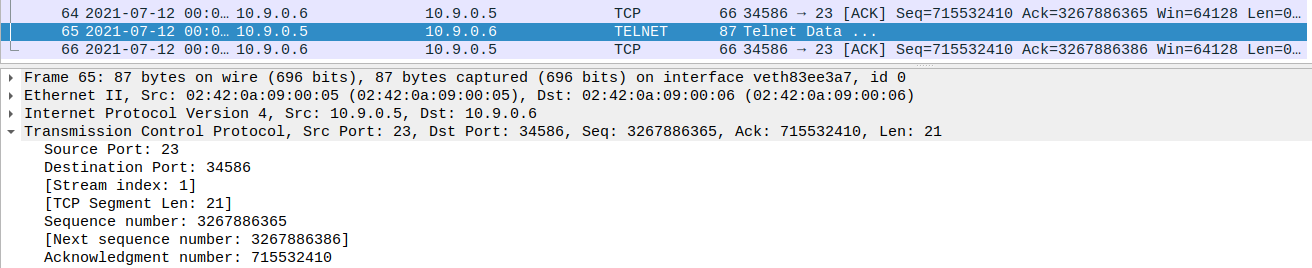
然后使用ls查看受害者主机根目录文件：发现成功生成a.txt，说明劫持成功。

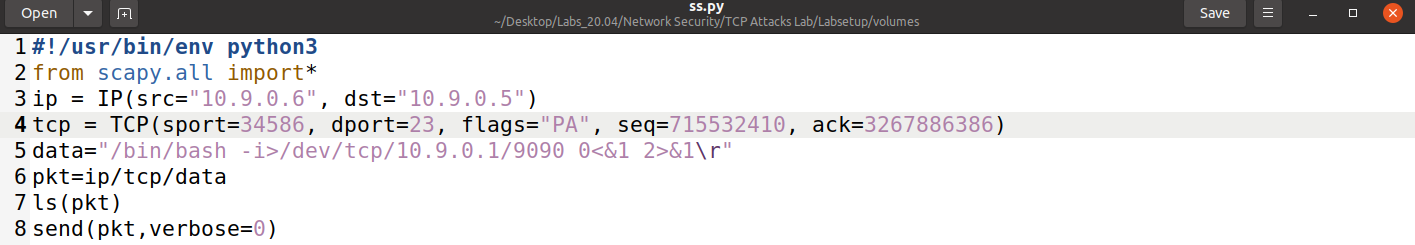


Task4： Creating Reverse Shell using TCP Session Hijacking

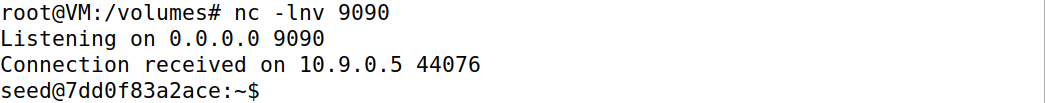
与task3 基本相同，除了代码命令其他步骤与上面相同。抓取的数据包和运行

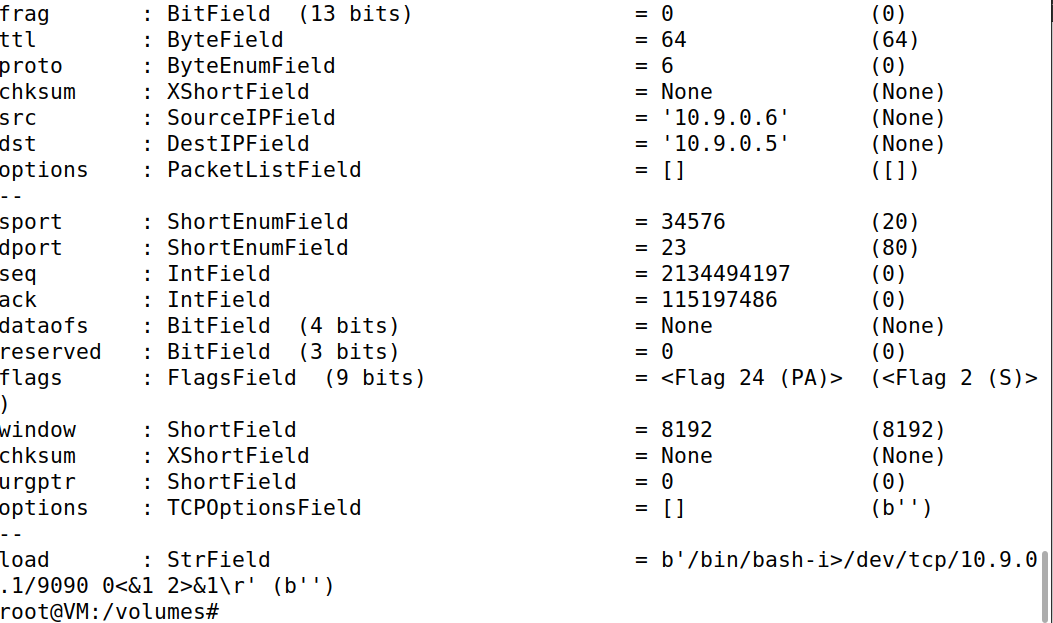
代码如下：





由于需要攻击方一边监听端口，一边发送反弹shell 的程序，所以这里开启两个攻击方的终端。其中一个终端先执行nc -lnv 9090 语句监听端口，然后再另一个终端执行我们的攻击程序。监听成功后两个终端结果如下：





然后我们就可以在攻击方主机上执行shell 控制被攻击主机了，例如查看当前路径：



以及查看被攻击主机网路信息，说明反shell 创建成功

