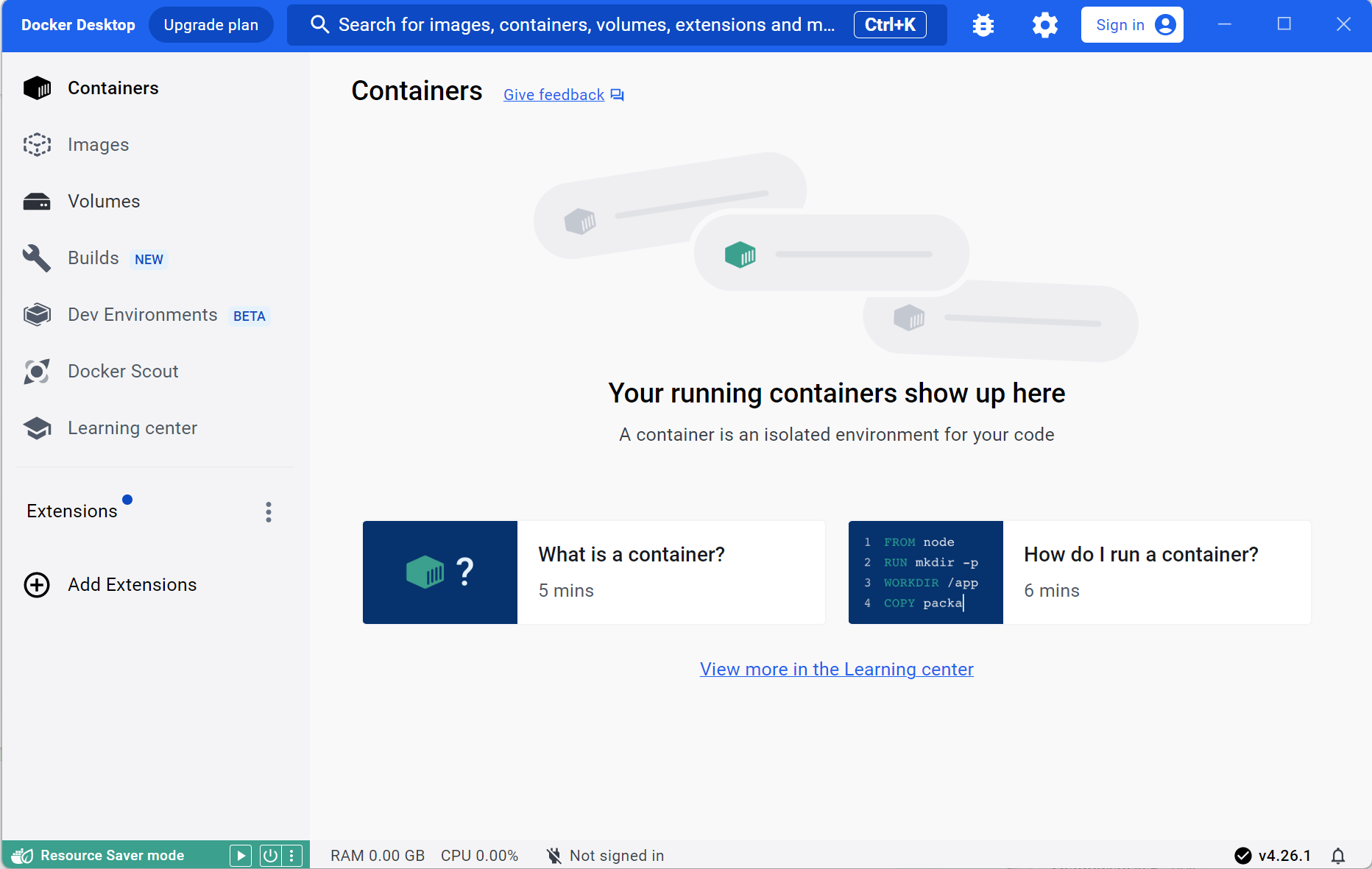
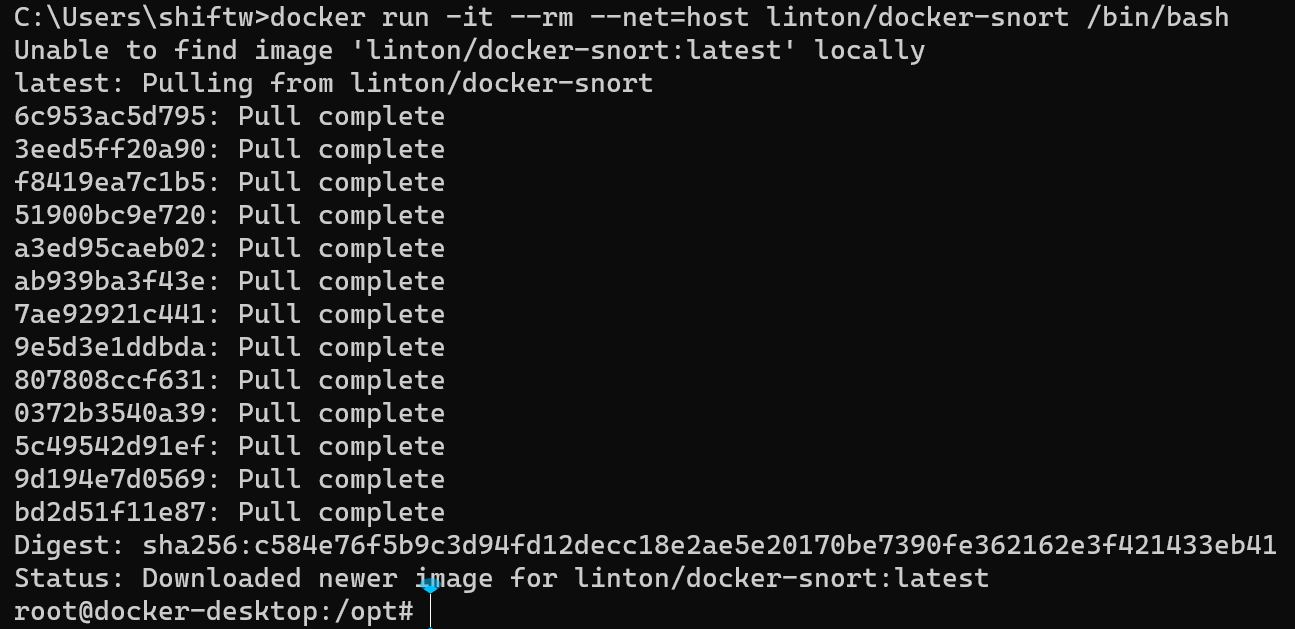
**网络入侵与防御实验**

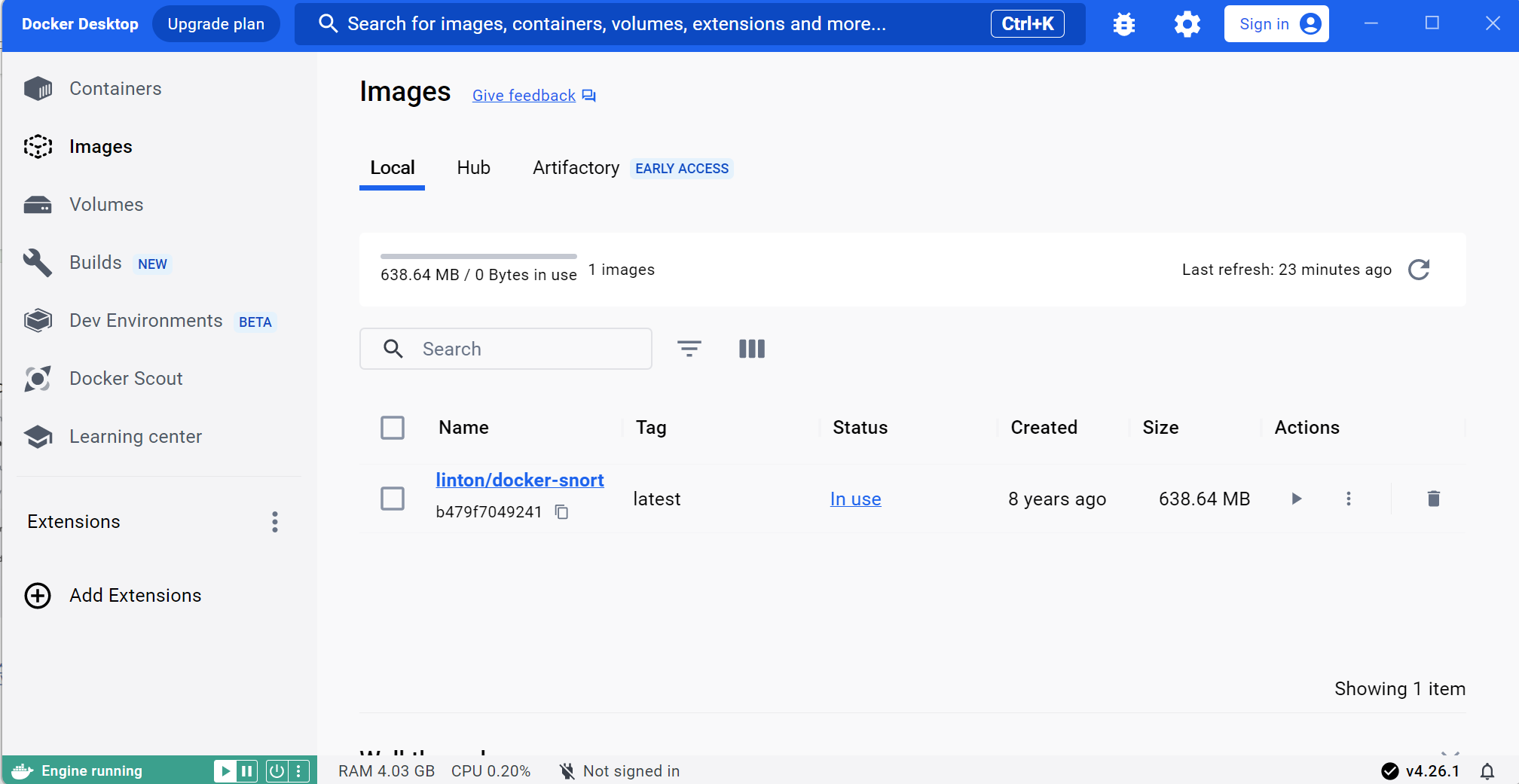
1. **实验1利用入侵检测系统检测 ICMP恶意扫描**
   1. **环境配置**

下载Docker并安装



下载snort container，安装运行成功





* 1. **检测 ICMP 数据包**

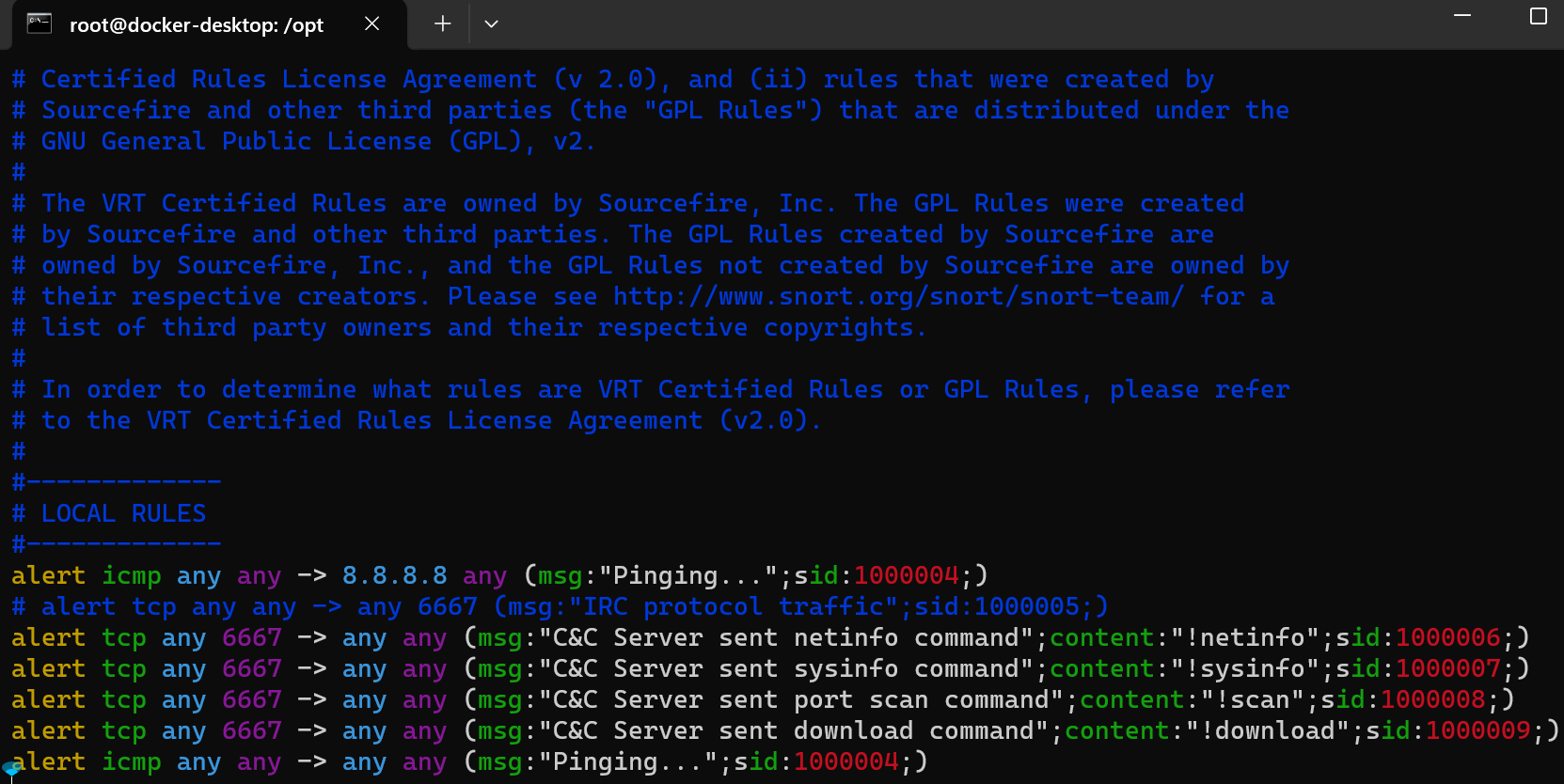
运行snort-docker

docker run -it --rm --net=host linton/docker-snort /bin/bash

在规则文件中添加ping检测规则：

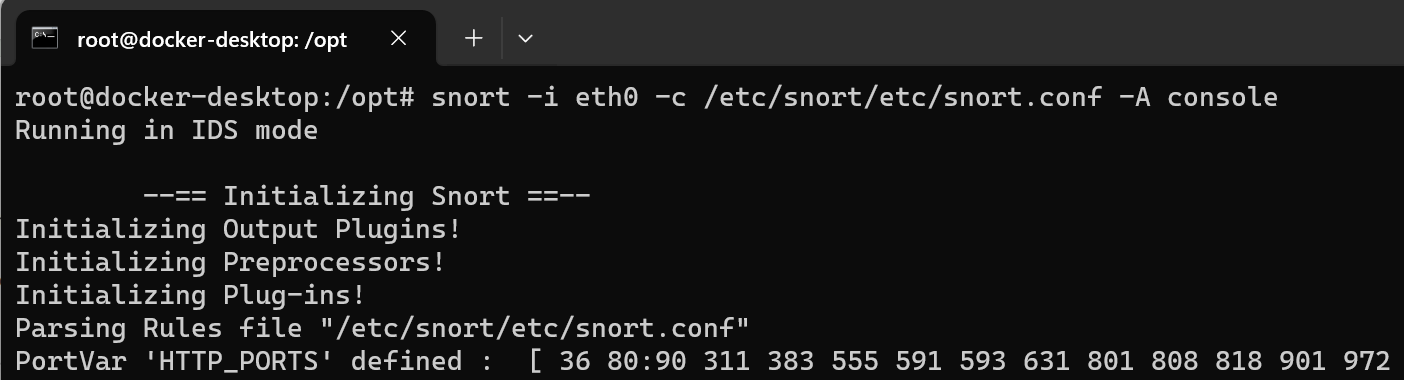
vi /etc/snort/rules/local.rules

alert icmp any any -> any any (msg:"Pinging...";sid:1000004;)

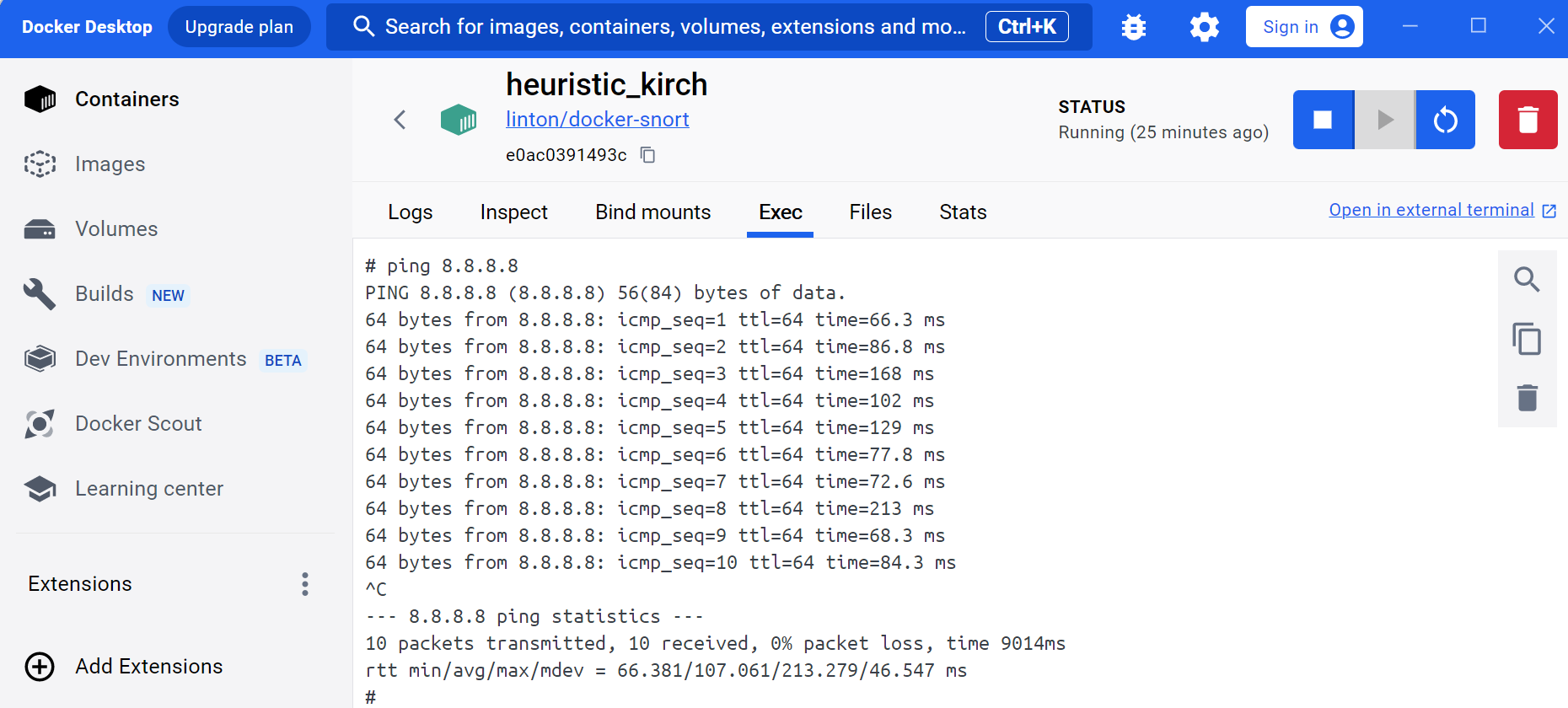


在 Snort-Docker 的 Terminal 运行 Snort

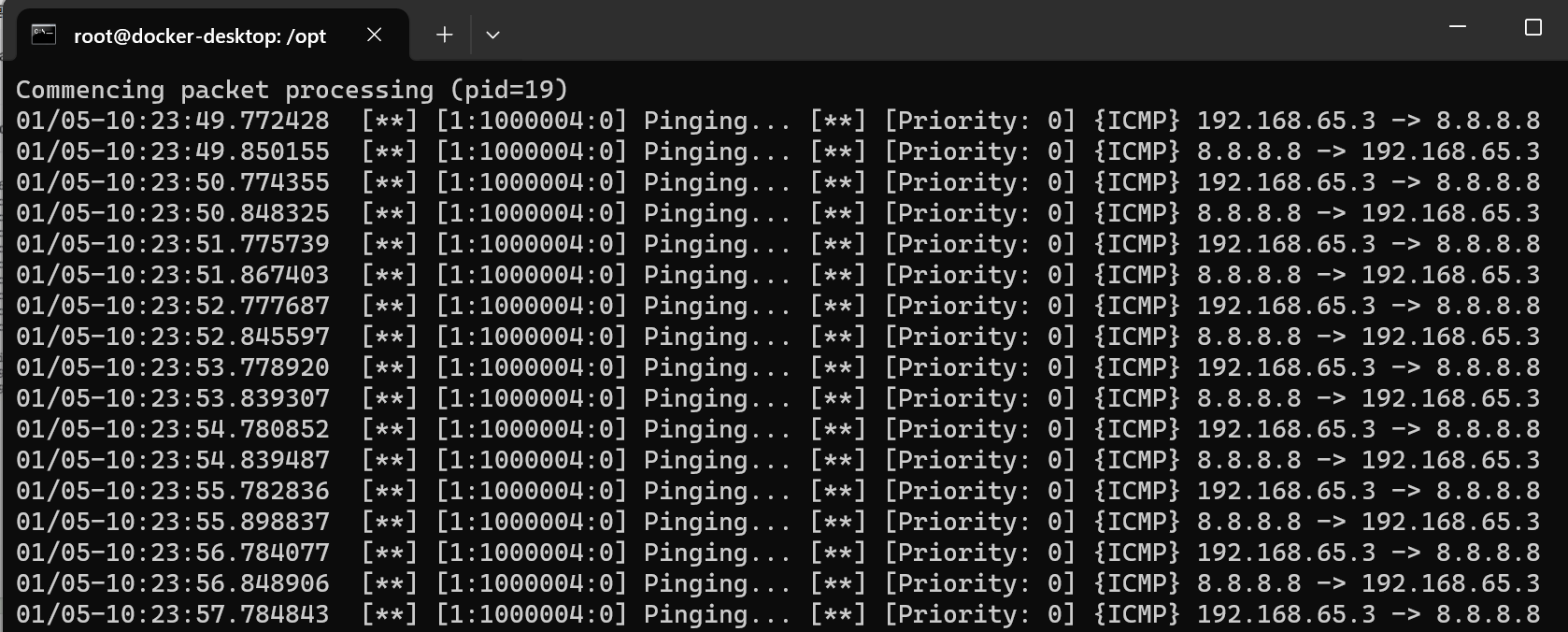
snort -i eth0 -c /etc/snort/etc/snort.conf -A console

****

在 Docker UI 界面运行 Ping 命令

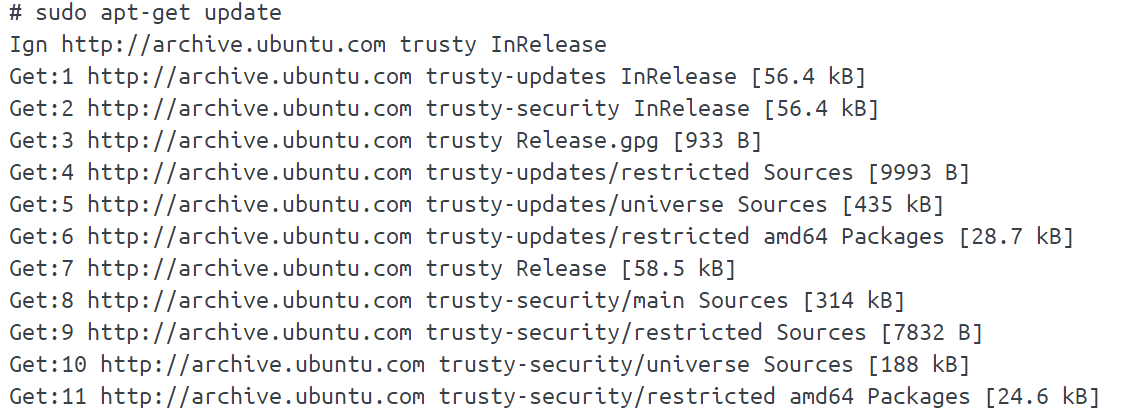


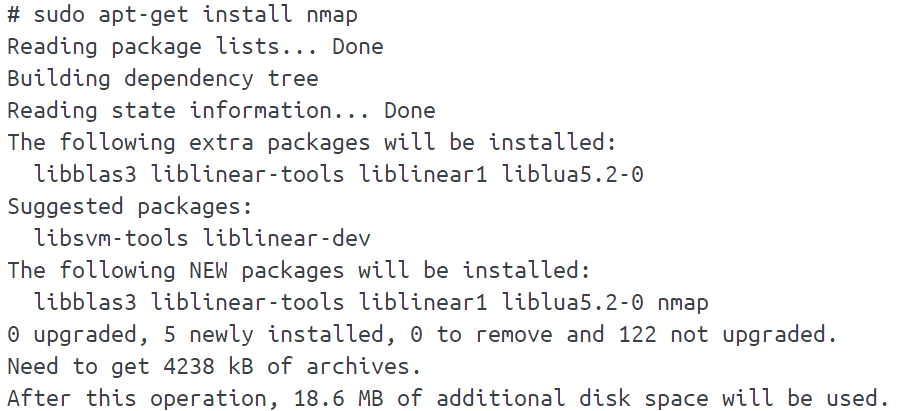
在终端上可以看到snort的检测结果



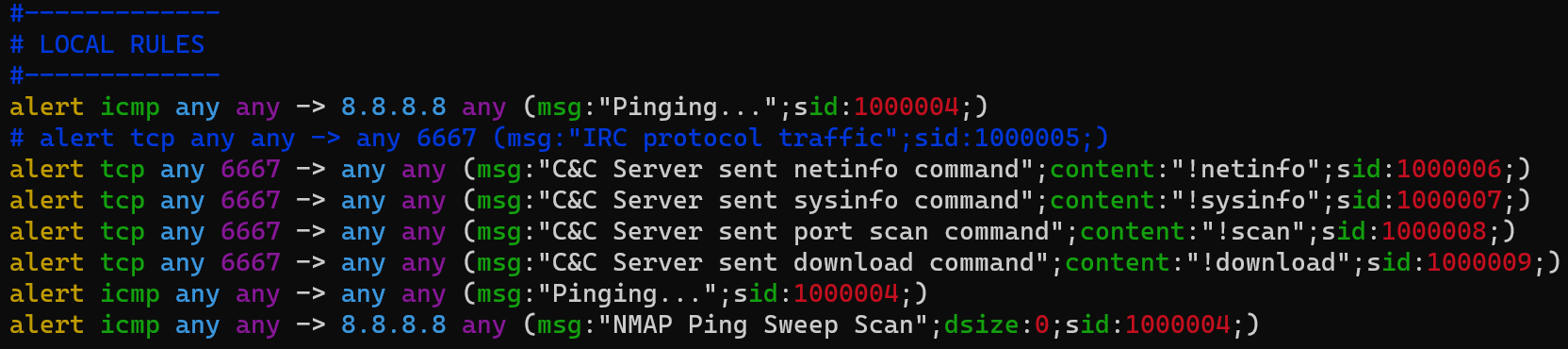
* 1. **检测 NMAP 扫描**

首先需要安装nmap





终端中修改snort检测规则，加入新规则alert icmp any any -> 8.8.8.8 any (msg:"NMAP Ping Sweep Scan";dsize:0;sid:1000004;)

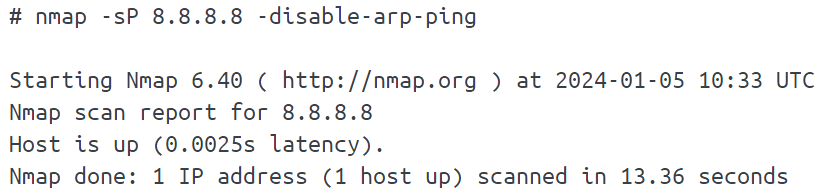


运行snort

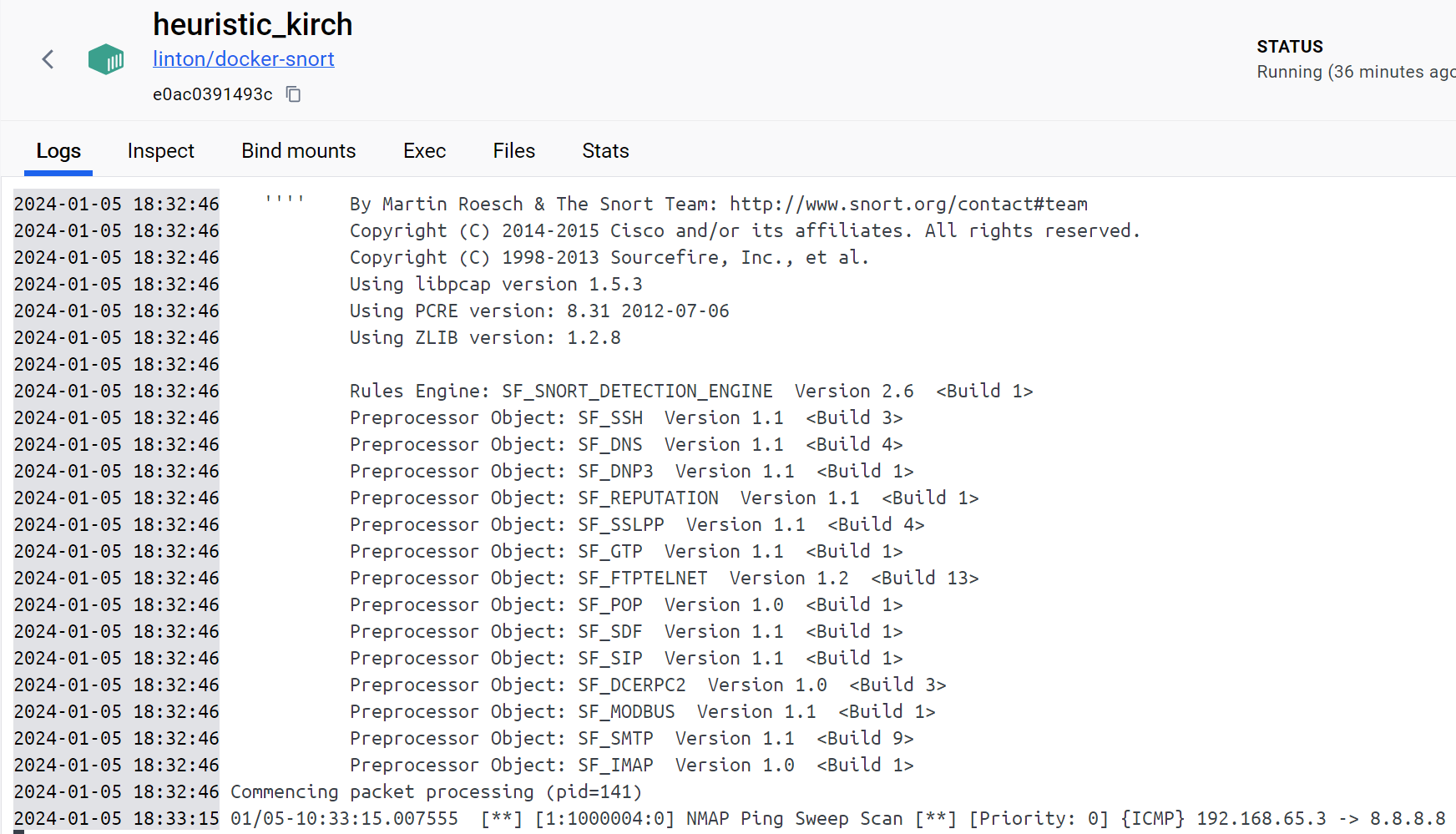
snort -i eth0 -c /etc/snort/etc/snort.conf -A console

然后在 Docker UI 界面进行 Nmap Scan：

nmap -sP 8.8.8.8 -disable-arp-ping

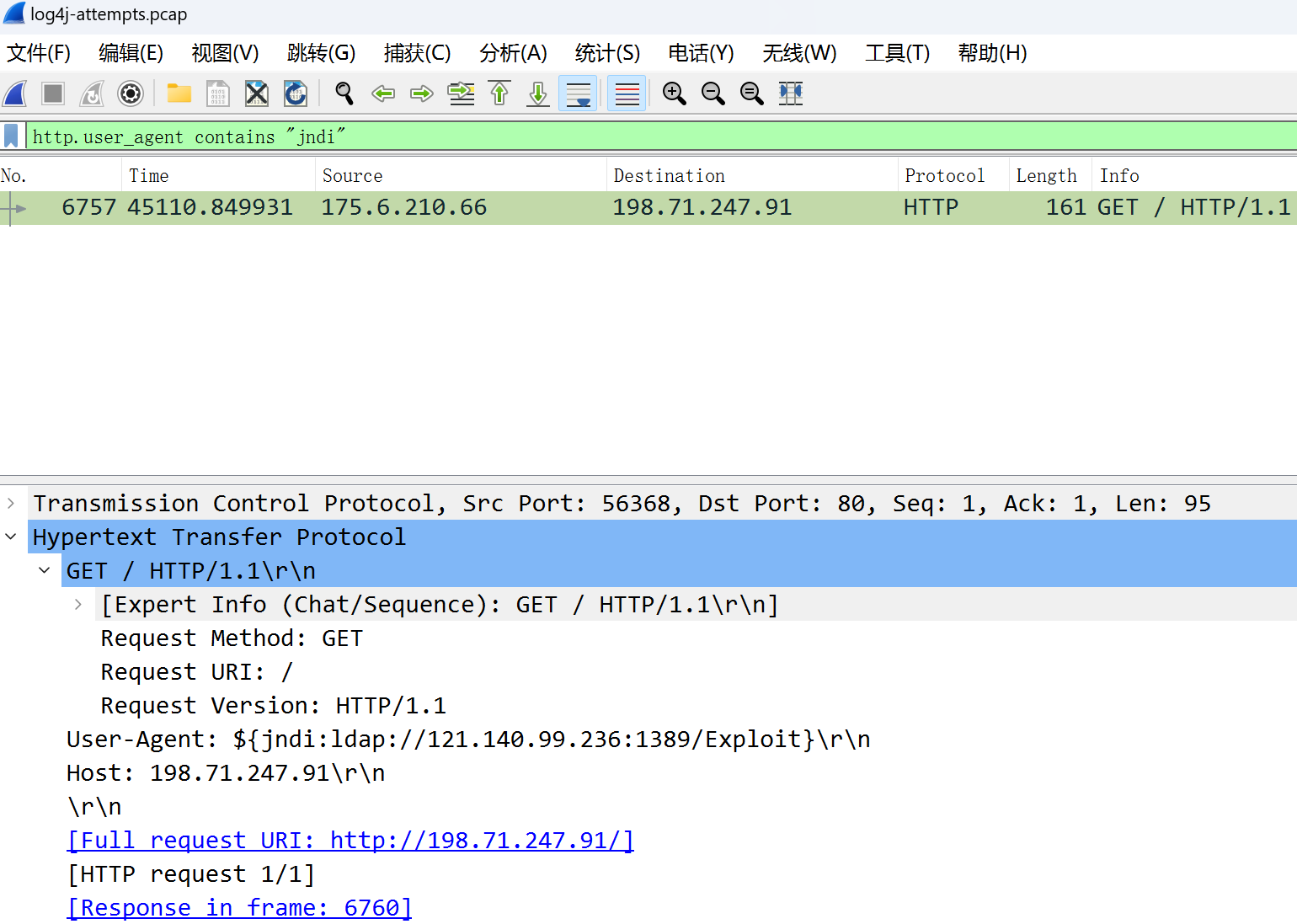


在 Docker UI 的 Log 界面显示成功检测结果



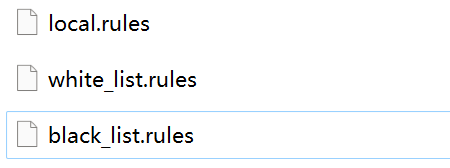
1. **实验2 利用网络入侵检测系统检测注入式攻击**
   1. **Snort 检测 Log4j 攻击**

用 wireshark 打开pcap 文件，输入http.user\_agent contains "jndi"过滤项过滤Log4j攻击包

****

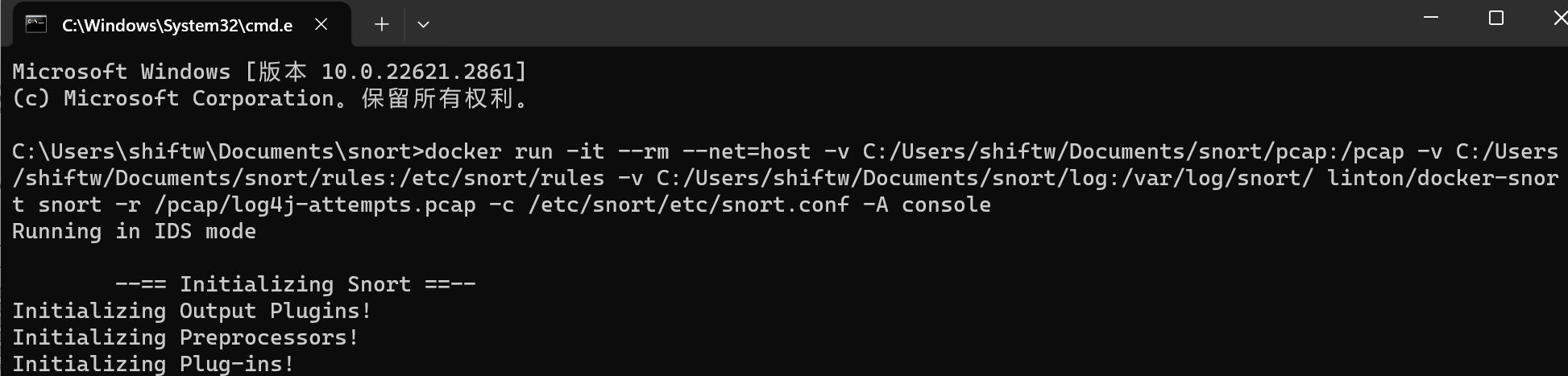
制定检测 Log4j 攻击的 snort 规则。在主机中建立子文件夹pcap，log，rules，并将log4j-attempts.pcap放入pcap，在rules中创建local.rules并加入检测规则alert tcp any any -> any 80 (content:"jndi:ldap://"; content:"GET"; http\_method; sid:1000000;)

实际操作中发现还需要创建两个文件，white\_list.rules black\_list.rules不然在之后的静态pcap分析中会报错

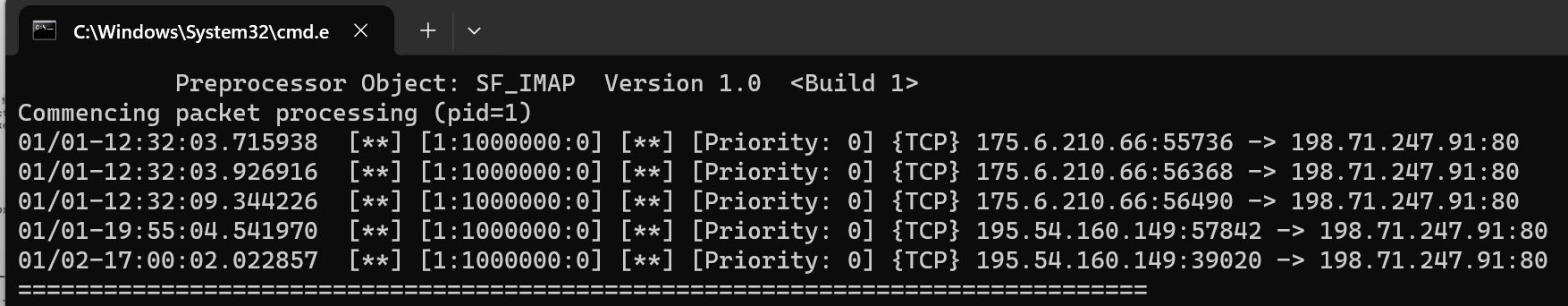


运行 Snort 检测 Log4j 攻击：

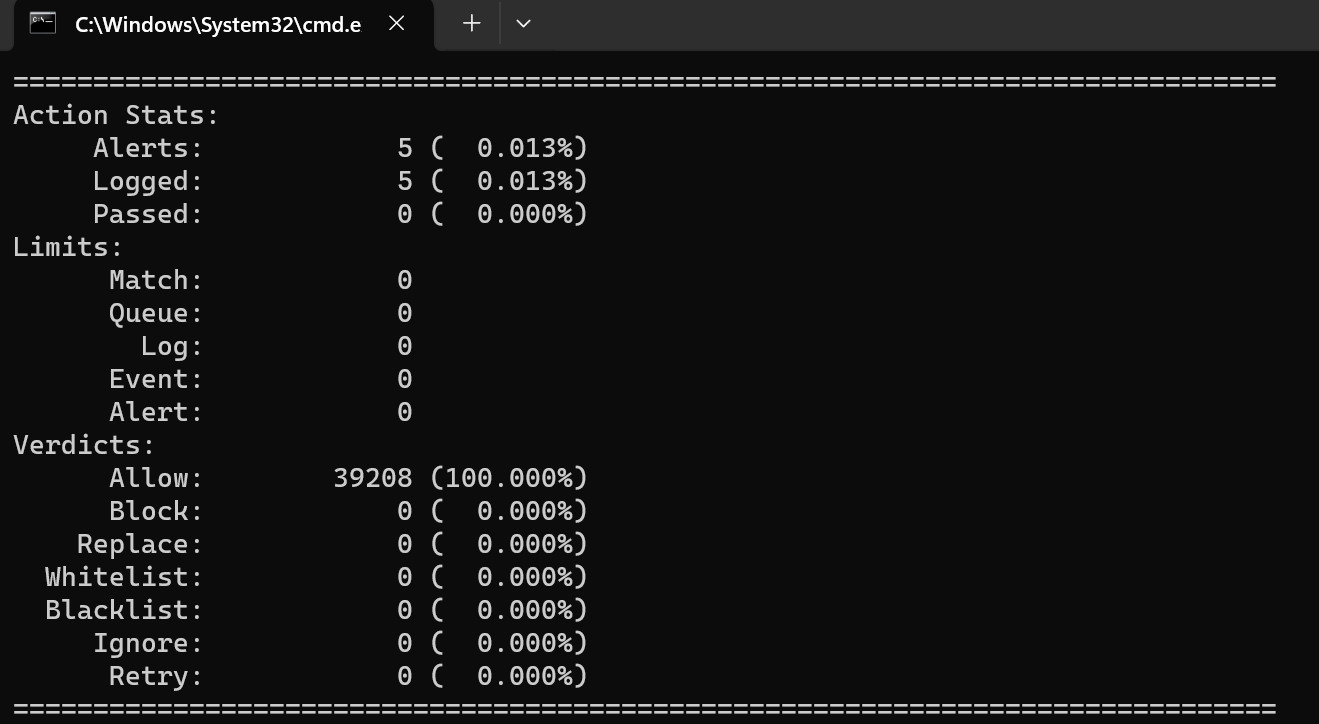
docker run -it --rm --net=host -v C:/Users/shiftw/Documents/snort/pcap:/pcap -v C:/Users/shiftw/Documents/snort/rules:/etc/snort/rules -v C:/Users/shiftw/Documents/snort/log:/var/log/snort/ linton/docker-snort snort -r /pcap/log4j-attempts.pcap -c /etc/snort/etc/snort.conf -A console



在snort终端中观察到结果，捕获的网络攻击包如下

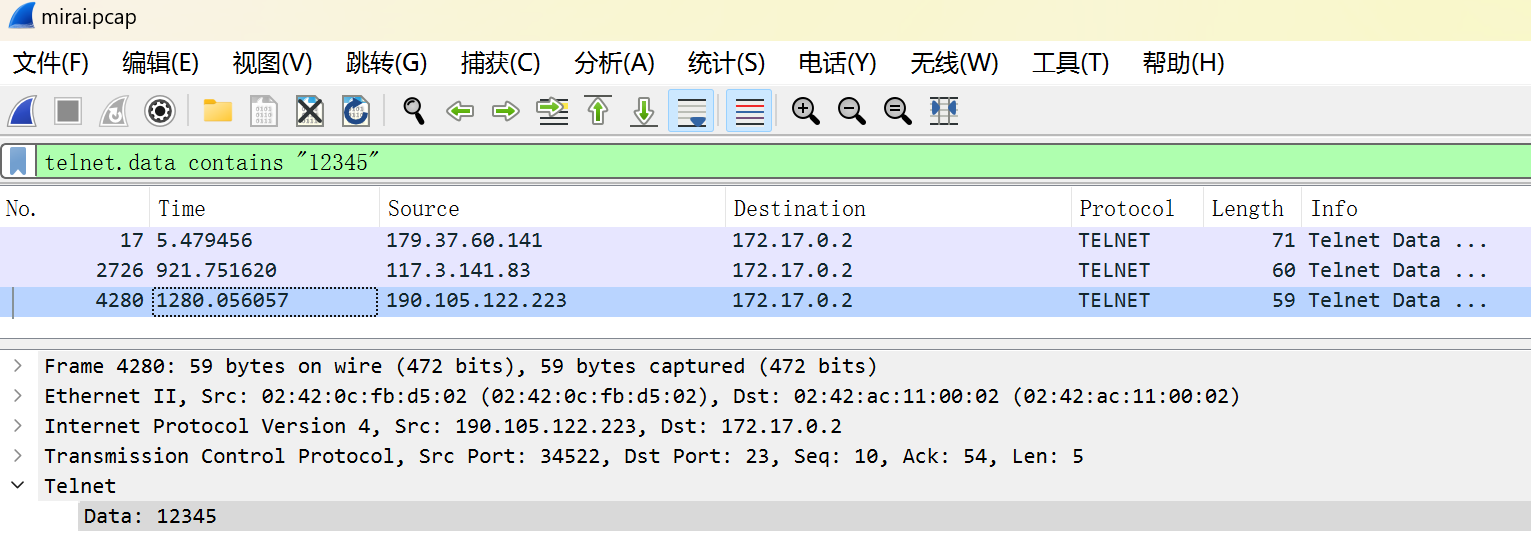


产生安全警告如下

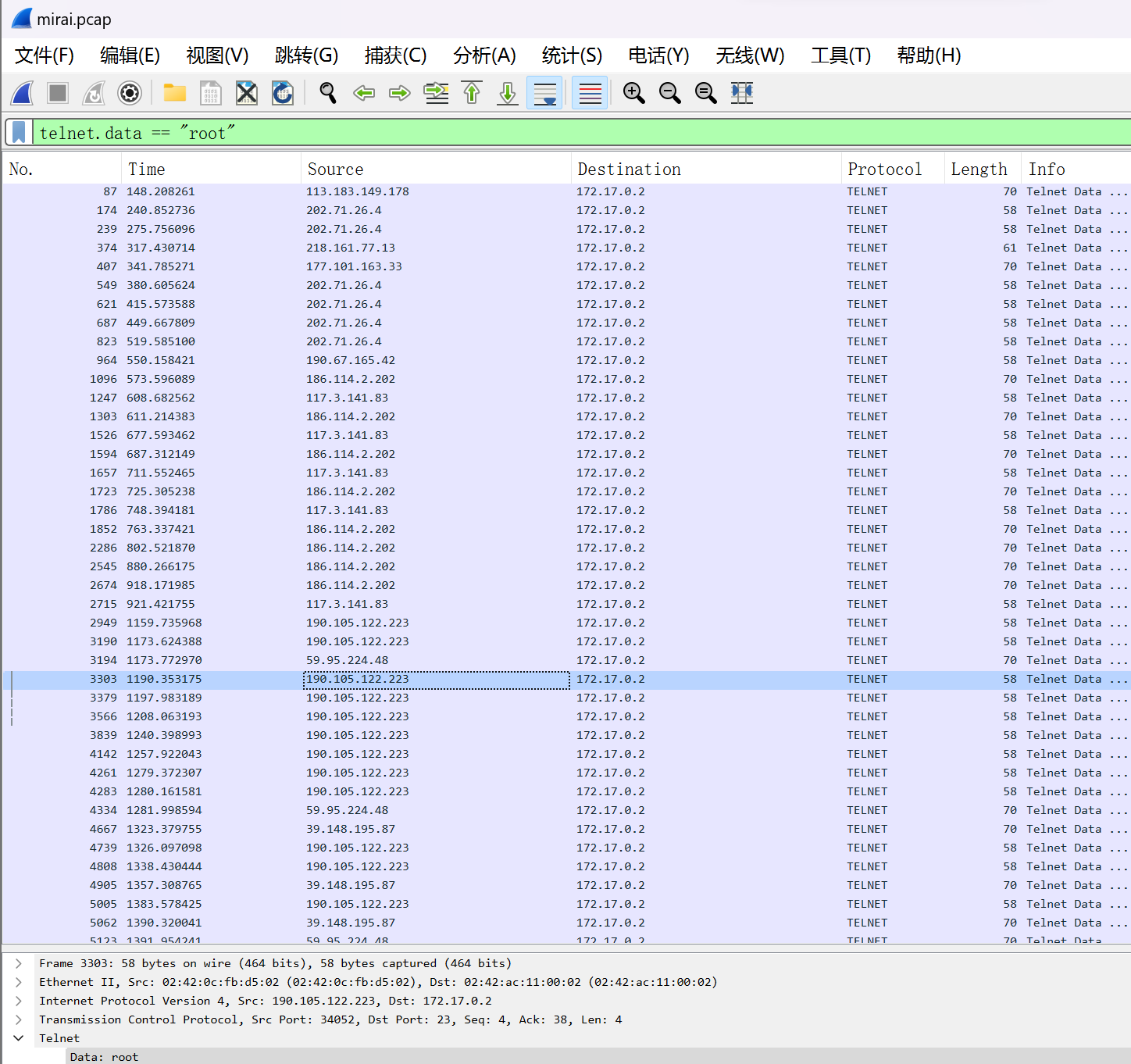


* 1. **练习 snort检测Mirai攻击**

先用wireshark分析网络流量，打开mirai.pcap文件，过滤出包含简单字符串12345的telnet包，结果如下，逐一检查发现都是向目标机发送简单字符串，很可能是mirai攻击



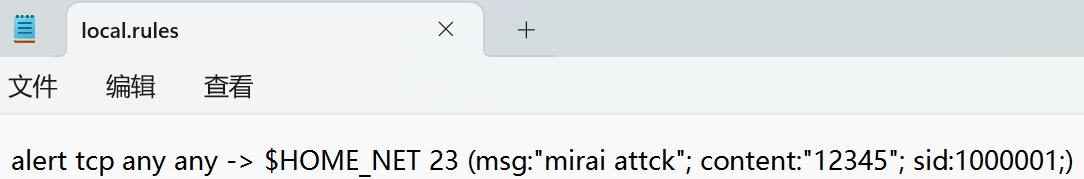
考虑到mirai攻击的目的是使用root权限控制设备，尝试过滤root用户登录的数据包，发现有很多。其中有短时间内多次发送申请root登录数据包，非常可疑，正常用户一般不会短时间内多次连续发包



用wireshark的追踪流功能，先选择第一个数据包的相关TCP流分析（telnet依托TCP协议），可以看到源IP成功使用简单密码登录root用户后，执行一系列指令。认为是攻击者发出的mirai攻击

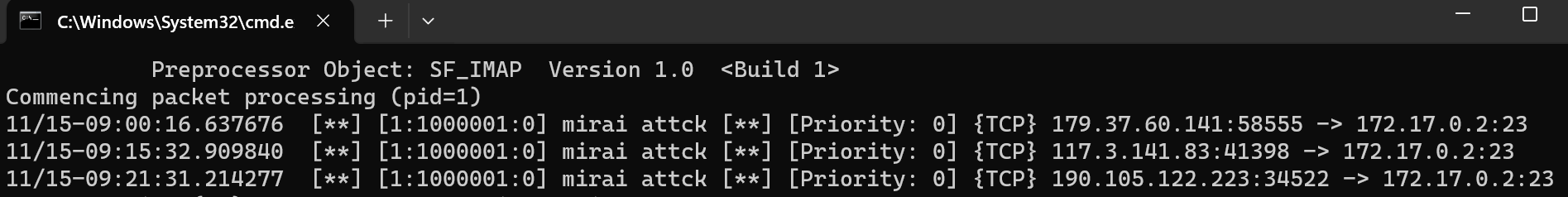


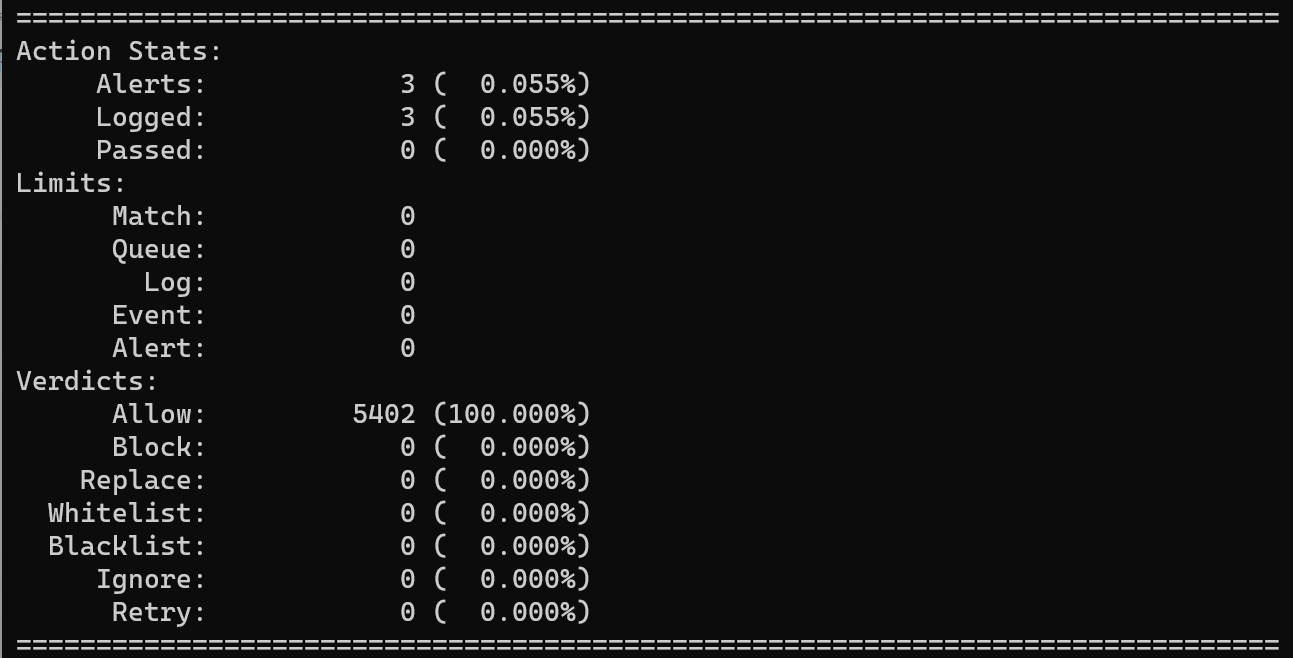
下一步进行snort静态分析pcap，由于telnet登录是依靠tcp协议的，所以设置tcp数据包检测的snort规则如下，设置目标地址是主机IP的23端口（mirai攻击常用的弱指令端口），设置包含简单密码。sid是识别规则的唯一序列，如果只有一条规则可以不设置，多条规则的话必须设置不同sid



然后在主机中开启snort分析

docker run -it --rm --net=host -v C:/Users/shiftw/Documents/snort/pcap:/pcap -v C:/Users/shiftw/Documents/snort/rules:/etc/snort/rules -v C:/Users/shiftw/Documents/snort/log:/var/log/snort/ linton/docker-snort snort -r /pcap/mirai.pcap -c /etc/snort/etc/snort.conf -A console





可以看到snort捕获的网络攻击包和wireshark分析的结果一致，也做出了对应的安全警告。实际上mirai攻击数据包并不是只是发送简单密码的数据包，应该还要包括整个攻击过程的所有数据包，比如输入root用户名，以及之后使用root权限执行非法命令等等。而且发送简单密码的也不一定是mirai攻击，实际中应该结合前面所分析的，短时间内多次申请root权限的才是较为可疑的攻击行为。

1. **思考**

不难发现，在snort的本地规则库中配置的所有规则，其访问者的IP和端口都设置为了any，在实际应用时，这将引起较大数量的警告误报。因此在实际应用中或许需要细化规则，从而避免一刀切地进行防御，以免过高的误报率。

且作为一个入侵防御系统（IPS），从防御检测技术来看，snort目前的防御方式似乎还停留在初代的规则匹配的包过滤防火墙的级别。未来可以考虑在snort上增加状态检测等功能提高其性能或安全性。