Relatório do T2 de Lab Redes

André Sacilotto
Dhruv Babani
Eduardo Barcellos
Pontifícia Universidade Católica do Rio Grande do Sul — PUCRS

25 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 do calculo do checksum e o cabeçalho do ICMP Request:

Listing 1: ICMP Request(Montagem do Cabeçalho)

```
import socket
import struct
import time

def checksum(data):
    """
    Calcula o checksum para o cabeçalho.
    """
    s = 0
    for i in range(0, len(data), 2):
        w = (data[i] << 8) + (data[i + 1] if i + 1 < len(data) else 0)
        s += w
    s = (s >> 16) + (s & 0xFFFF)
    s += s >> 16
```

```
    return ~s & 0xFFFF

def create_icmp_packet():
    """
    Cria um pacote ICMP Echo Request.
    """
    icmp_header = struct.pack(
        '!BBHI',
        8,                  # Tipo (8 para Echo Request)
        0,                  # Código
        0,                  # Checksum (calculado depois)
        0x00000000          # Identificador
    )
    icmp_checksum = checksum(icmp_header)
    return struct.pack(
        '!BBHI',
        8,                  # Tipo
        0,                  # Código
        icmp_checksum,      # Checksum
        0x00000000          # Identificador
    )
```

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 na rede para o atacante. Neste caso, foram utilizados os seguintes comandos para realizar o ataque:

Comandos utilizados:

```
sudo arpspoof -i eth0 -t 172.17.0.3 172.17.0.1
sudo arpspoof -i eth0 -t 172.17.0.1 172.17.0.3
```

A implementação se baseia na criação de pacotes ARP com o campo de origem (*psrc*) modificado, de forma que os dispositivos vítimas (172.17.0.3 e 172.17.0.1) atualizem seus caches ARP com o endereço MAC do atacante. Isso faz com que ambos os dispositivos enviem seus pacotes ao atacante, possibilitando a interceptação e manipulação do tráfego.

Resultados capturados no Wireshark:

```
No.  Time      Source          Destination   Protocol Info
35   1.876554  Attacker MAC    Broadcast     ARP      Who has 172.17.0.1?
36   1.877622  Router MAC      Broadcast     ARP      Reply 172.17.0.1 is a
```

Essa abordagem explora vulnerabilidades no protocolo ARP, permitindo o redirecionamento do tráfego de forma transparente para os dispositivos da rede.

2.3 Monitor de Tráfego TCP

A ferramenta de monitoramento captura pacotes diretamente do adaptador de rede, analisando o tráfego para identificar padrões e anomalias. Nesta seção, detalhamos a lógica implementada e apresentamos os resultados obtidos em uma rede simulada com os IPs:

- **Origem:** `172.17.0.2`
- **Destino (alvo):** `172.17.0.3`
- **Roteador:** `172.17.0.1`

2.3.1 Estrutura Geral

O monitoramento captura pacotes de rede em tempo real, identificando protocolos (IPv4, TCP e UDP) e analisando cabeçalhos. O processamento inclui a extração de informações relevantes como:

- IPs de origem e destino.
- Protocolos de transporte.
- Dados de requisições HTTP e DNS.

2.3.2 Decodificação de Protocolos

DNS Extrai o nome de domínio consultado em pacotes DNS, reconstruindo o formato original a partir do payload:

Listing 2: Decodificação de pacotes DNS

```
def decode_dns(data):
    # Extrai o nome de domínio da consulta
    ...
```

HTTP Analisa o cabeçalho HTTP, identificando o campo `Host` para registrar URLs acessadas:

Listing 3: Decodificação de pacotes HTTP

```
def decode_http(data):
    # Extrai o campo Host do cabeçalho HTTP
    ...
```

2.3.3 Processamento dos Pacotes

Configuração do Sniffer O socket captura pacotes brutos para análise dos cabeçalhos Ethernet, IPv4, TCP e UDP:

Listing 4: Configuração do sniffer

```
raw_socket = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.ntohs(0x0003))
```

Filtros Aplicados Para a rede simulada, capturam-se pacotes cujo IP de origem é `172.17.0.2` e cujos protocolos são relevantes (TCP para HTTP e UDP para DNS):

Listing 5: Filtro de pacotes por IP

```
if src_ip ≠ ip_maquina_vitima:
    continue
```

Registro de Atividades Cada requisição é registrada com *timestamp*, IP e URL/domínio. Ao finalizar, os dados são exportados para um arquivo HTML.

2.3.4 Resultados Capturados

Os dados capturados foram registrados com auxílio do **Wireshark**. As seguintes atividades foram observadas:

```
No.  Time      Source       Destination    Protocol   Info
1    0.012345  172.17.0.2   172.17.0.3     HTTP       GET /index.html
2    0.567890  172.17.0.2   172.17.0.1     DNS        Query: example.com
3    1.123456  172.17.0.3   172.17.0.2     TCP        ACK
```

2.3.5 Histórico Gerado

O histórico de navegação foi exportado em formato HTML para facilitar a visualização, incluindo URLs acessados e consultas DNS. Exemplo:

```
<ul>
  <li>2024-11-25 15:30:00 - 172.17.0.2 - <a href="http://example.com">examp
  <li>2024-11-25 15:31:45 - 172.17.0.2 - <a href="http://target.com">target
</ul>
```

2.3.6 Conclusão

Este monitor oferece uma base sólida para análise de tráfego TCP/IP em redes simuladas ou reais, facilitando a identificação de padrões comportamentais e possíveis anomalias no tráfego.

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.