Grupo E
Participantes: João Pedro Antunes, Giorgia Marques, Tomás Caldas

Trabalho final de Redes

1ª Etapa: Varredura de hosts pela rede

A primeira etapa do trabalho consiste na criação de uma ferramenta para identificar hosts ativos na rede. Essa ferramenta realiza uma varredura semelhante a um ping scan, enviando pacotes ICMP para identificar dispositivos conectados na rede.

A ferramenta usa AF_INET para a criação de um socket RAW, com uso da biblioteca time.h para definir um timeout para os pacotes enviados. Para cada IP dentro do intervalo de endereços, um pacote ICMP é enviado, e um reply ICMP indica que o host está ativo.

Estrutura dos headers

Cabeçalho IP

Campo	Descrição
ihl	Comprimento do cabeçalho IP.
version	Versão do protocolo IP (para IPv4 , o valor é 4).
tos	Tipo de serviço ou prioridade do pacote.
total_len	Tamanho total do pacote IP, incluindo cabeçalho e dados.
id	Identificador do pacote.
frag_off	Deslocamento de fragmento e flags de fragmentação.
ttl	Tempo de vida do pacote, limita o número de saltos.
protocol	Protocolo da camada superior (TCP, UDP, ICMP).
checksum	Verifica a integridade do cabeçalho.
saddr	Endereço IP de origem (emissor).
daddr	Endereço IP de destino (receptor).

Cabeçalho ICMP

Campo	Descrição
type	Tipo de mensagem (8 para Echo Request, 0 para Echo Reply).
code	Código específico (0 para Request e Reply).
checksum	Verifica a integridade do cabeçalho.
id	Identificador do pacote.
seq	Número de sequência do pacote.

Cálculo do checksum

O checksum é calculado somando-se pares de bytes de 16 bits. Caso haja um byte restante, ele é somado ao final. Se houver um carry, ele é somado novamente para obter o complemento de 1, resultando no checksum.

```c
uint16_t checksum_cal(void *buffer, int len){
    uint16_t *buf = buffer;
    uint32_t sum = 0;
    uint16_t result;

    for(sum = 0; len > 1; len -= 2){
        sum += *buf++;
    }
    if(len == 1){
        sum += *(uint8_t *)buf;
    }
    sum = (sum >> 16) + (sum & 0xFFFF);
    sum += (sum >> 16);
    result = ~sum;
    return result;
}
```

Cálculo do CIDR

A função cidr permite calcular o endereço base e a máscara da rede a partir da notação CIDR. Utilizando sscanf, o endereço e o prefixo são separados. A máscara de rede é obtida deslocando 32 menos o prefixo para a esquerda, definindo a parte fixa do endereço. Com isso, a parte do host é zerada.

```c
void cidr(const char *cidr, uint32_t *base_ip, uint32_t *mask){
    char ip[32];
    int network_num;

    sscanf(cidr, "%[^/]/%d", ip, &network_num); //Separa endereço IP e número de bits de rede

    inet_pton(AF_INET, ip, base_ip);
    *mask = htonl((0XFFFFFFFF << (32 - network_num)) & 0xFFFFFFFF); //Exemplo: 32-24 = 8bits

    *base_ip &= *mask;  //Exemplo: 192.168.0.197 & 255.255.255.0 = 192.168.0.0
}
```

Execução:

```
user@debianvm:~/Downloads/TF/PARTE1$ sudo ./sn 10.89.0.1/28 500
Varredura na rede 10.89.0.1/28 com timeout de 500 ms...
Host inativo: 10.89.0.1
Host ativo: 10.89.0.2
Host ativo: 10.89.0.3
Host ativo: 10.89.0.4
Host inativo: 10.89.0.5
Host inativo: 10.89.0.6
Host inativo: 10.89.0.7
Host inativo: 10.89.0.8
Host inativo: 10.89.0.9
Host inativo: 10.89.0.10
Host inativo: 10.89.0.11
Host inativo: 10.89.0.12
Host inativo: 10.89.0.13
Host inativo: 10.89.0.14
Total de hosts ativos: 3
```

Grupo E
Participantes: João Pedro Antunes, Giorgia Marques, Tomás Caldas

2ª Etapa: Intercepção de pacotes

Para interceptar o tráfego entre o host alvo e o roteador, foi-se utilizado a técnica de ARP Spoofing com a ferramenta *arpspoof*.

Passos:

- Instalação do pacote dsniff: **sudo apt install dsniff**
- Envio de pacotes ARP para o host alvo indicando que o atacante é o roteador.
  Exemplo: **sudo arpspoof -i enp0s3 -t 192.168.0.197 192.168.0.1**
- Em paralelo, são enviados pacotes para o roteador indicando que o atacante é o host alvo.
  Exemplo: **sudo arpspoof -i enp0s3 -t 192.168.0.1 192.168.0.197**
- Para que o processo continue, é necessário ativar o IP Forwarding, fazendo com que o atacante esteja apto a receber pacotes do roteador.
  Exemplo: **sudo echo 1 > /proc/sys/net/ipv4/ip_forward**

Esse procedimento força a tabela ARP dos dispositivos envolvidos a serem alteradas, de forma que o atacante passe a mediar o tráfego entre o alvo e o roteador.

Verificação da tabela ARP na máquina atacada:
Note que 192.168.0.234 é o endereço de IP do atacante.

Antes:

```
Interface: 192.168.0.197 --- 0xe
  Endereço IP           Endereço físico       Tipo
  192.168.0.1           02-00-00-00-00-04     dinâmico
  192.168.0.205         a4-63-a1-70-d3-c3     dinâmico
  192.168.0.234         08-00-27-cc-69-1c     dinâmico
  192.168.0.255         ff-ff-ff-ff-ff-ff     estático
  224.0.0.22            01-00-5e-00-00-16     estático
  224.0.0.251           01-00-5e-00-00-fb     estático
  224.0.0.252           01-00-5e-00-00-fc     estático
  239.255.255.250       01-00-5e-7f-ff-fa     estático
  255.255.255.255       ff-ff-ff-ff-ff-ff     estático
```

Depois:

```
Interface: 192.168.0.197 --- 0xe
  Endereço IP           Endereço físico       Tipo
  192.168.0.1           08-00-27-cc-69-1c     dinâmico
  192.168.0.205         a4-63-a1-70-d3-c3     dinâmico
  192.168.0.234         08-00-27-cc-69-1c     dinâmico
  192.168.0.255         ff-ff-ff-ff-ff-ff     estático
  224.0.0.22            01-00-5e-00-00-16     estático
  224.0.0.251           01-00-5e-00-00-fb     estático
  224.0.0.252           01-00-5e-00-00-fc     estático
  239.255.255.250       01-00-5e-7f-ff-fa     estático
  255.255.255.255       ff-ff-ff-ff-ff-ff     estático
```

Agora o endereço MAC do roteador é o mesmo do interceptador.

Grupo E
Participantes: João Pedro Antunes, Giorgia Marques, Tomás Caldas

3ª Etapa: Leitura de pacotes interceptados

Foi desenvolvida uma aplicação sniffer que utiliza sockets raw para capturar pacotes de rede e reconstruir o histórico de navegação, identificando e armazenando os sites acessados a partir de pacotes DNS e HTTP.

Com as estruturas dos cabeçalhos de IP, TCP, UDP e DNS definidas, o sniffer consegue identificar pacotes DNS e HTTP. Para pacotes DNS, o sniffer identifica aqueles que utilizam a porta 53 no protocolo UDP e extrai os nomes de domínio a partir dos dados contidos nos pacotes. Já para pacotes HTTP, o sniffer foca na análise dos pacotes que utilizam a porta 80 no protocolo TCP, verificando a carga útil em busca de requisições do tipo GET para capturar URLs completas.

Implementação do sniffer

- Configuração dos Sockets: Os sockets são criados com a função socket() para os protocolos IPPROTO_TCP e IPPROTO_UDP, permitindo a captura de pacotes TCP e UDP. Cada socket utiliza um buffer de recepção para armazenar temporariamente os pacotes.
- Identificação de Pacotes DNS: Pacotes DNS são identificados pela porta 53 no protocolo UDP. O cabeçalho DNS é acessado no deslocamento correto do buffer, e o nome de domínio é extraído e formatado como uma URL com o prefixo http:// e registrados em um arquivo HTML.
- Identificação de Pacotes HTTP: Pacotes HTTP são filtrados pela porta 80 no protocolo TCP. A carga útil é analisada para identificar requisições GET, que contêm a URL completa do recurso acessado.
- Organização e Formatação do URL para Pacotes DNS:
    - No início, o HTML registrava todos os URLs detectados, mesmo os que não foram acessados. Então ajustamos o código para salvar apenas os URLs acessados enquanto o programa está ativo.
    - Pacotes DNS armazenam os nomes de domínio em uma forma compactada, com labels que indicam o comprimento de cada parte do nome, separadas por pontos ao serem reconstruídas. Por exemplo, “www.google.com” é armazenado como "3www6google3com". O código há uma parte que transforma essa sequência no formato legível, adicionando pontos entre os labels.
- Registro do histórico de Navegação: Data, hora, endereço IP e URL são registrados em um arquivo HTML. Cada entrada é formatada como um item de lista clicável, usando a função write_html().

Grupo E

Participantes: João Pedro Antunes, Giorgia Marques, Tomás Caldas