Lucas Lima

Trabalho 1
CI301

Brasil

23 de Abril de 2018

1 Decisões

1.1 Linguagem

A linguagem C foi escolhida com o objetivo de desenvolver e entender o algoritmo de varredura de portas em baixo-nivel, bem como ganhar experiência em programação de redes e utilização do protocolo TCP.

1.2 Parsing

Para realizar o parsing dos alcances de endereços IP e das portas passadas como parâmetro para o programa, foi utilizada a função strtok(), da biblioteca string.h. A função recebe um ponteiro para char e um delimitador como parâmetros, com o objetivo de separar os conteúdos entre o limitadores em tokens. A função retorna um ponteiro para o começo do primeiro token, e retorna NULL caso nenhum token tenha sido encontrado. Chamadas subsequentes à função retornam ponteiros para os próximos tokens.

1.3 Representação dos Ranges

Para representar o alcance dos endereços IP de inínio e fim, foi escolhido utilizar um vetor de inteiros, onde cada índice guarda o valor de um octeto. Os dois endereços são representados no mesmo vetor, e o acesso ao endereço de início ou fim, ocorre com a utilização (ou não) de um offset. Dessa forma é possível iterar em todos os octetos.
A transformação dos octetos representados com inteiros para uma string com o endereço completo ocorre com a utilização da função sprintf().
Algo parecido é feito para representar o range de portas, onde é utilizado um vetor de inteiros de duas posições para representar a porta de início e fim, respectivamente.

1.4 Socket não bloqueante e hosts inativos

Alguns problemas foram encontrados ao tentar realizar a conexão em hosts inativos. Para mitigá-los, antes de realizar a conexão TCP nas portas do alvo, confirmamos se o host está ativo com o envio de pings. Futuramente se faz necessária a implementação de uma opção que pula este passo para que o programa possa ser utilizado em hosts que não aceitem pacotes ICMP.
Além disso, o tempo de resposta estava demasiadamente alto para os casos em que é realizada uma tentativa de conexão em uma porta filtrada ou fechada. Este problema foi

contornado utilizando-se de sockets não bloqueantes através da função fcntl() presente na biblioteca fcntl.h, para manipulação de opções em descritores de arquivos. Dessa forma pude definir um timeout, que atualmente é de 250ms, para a resposta da porta.

2 Varreduras

Varredura realizada na máquina listada em sala de aula.

```
$ ./portscan 200.238.144.29 1-1024
Scanning host 200.238.144.29
Port open: 22
Banner: SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1.9


Ports closed or filtered: 1023
```

Varredura realizada num alcance de endereços da minha rede doméstica.
O último endereço (192.168.0.24) é de um Raspberry PI com ssh habilitado.

```
$ ./portscan 192.168.0.22-24 1-1024
Scanning range 192.168.0.22 to 192.168.0.24
Scanning host 192.168.0.22
Host 192.168.0.22 unreacheable. Check if it responds to ping commands.
Scanning host 192.168.0.23
Host 192.168.0.23 unreacheable. Check if it responds to ping commands.
Scanning host 192.168.0.24
Port open: 22
Banner: SSH-2.0-OpenSSH_7.4p1 Raspbian-10+deb9u3


Ports closed or filtered: 1023
```