Relatório do projeto de IRC

Bernardo Prior - 2018285108

Luís Silva - 2018278644

Professor Vasco Pereira, IRC 2019/2020

Introdução

O objetivo deste projeto é desenvolver sistema cliente-servidor para transferência de ficheiros entre dois sistemas recorrendo aos protocolos de transporte TCP e UDP. Para tal, começámos por rever o código desenvolvido na ficha 3 acrescentando a esta estrutura um proxy. Atribuímos ao servidor o endereço *127.0.0.2* e ao proxy o endereço *127.0.0.3.* A partir daqui, fomos implementando gradualmente todas as funcionalidades requisitadas no enunciado e alguns extras para um melhor funcionamento do nosso programa.

Conexão Cliente - Servidor

Como atribuímos diferentes endereços ao servidor e ao proxy podemos conceder o mesmo porto a ambos. Numa primeira fase da conexão, o cliente transmite ao proxy via TCP o endereço do servidor fornecido ao iniciar o executável. Com este endereço, o proxy conecta-se ao servidor utilizando o mesmo porto que lhe foi atribuído.

Assim que o servidor recebe uma conexão, verifica se o número de conexões simultâneas já atingiu o limite. Como decidimos recorrer à criação de um processo para cada conexão, tivemos necessidade de criar um mecanismo para que os processos tenham acesso ao número de conexões que existem no momento. Deste modo, recorremos à criação de uma shared memory que contém informação sobre o número de clientes. Sempre que uma conexão ultrapasse o limite, é enviada para o cliente uma mensagem de rejeição, o que leva ao término do cliente e ao processo , do servidor, responsável pela sua conexão.

Autenticação

Todos os clientes que se conectam ao servidor são submetidos a um processo de autenticação mútuo entre cliente e servidor. Para tal recorremos, como sugerido, à biblioteca *libsodium.* Graças a esta, o cliente gera uma chave pública tal como o servidor. De seguida, estas chaves são transmitidas e validadas entre os dois. No caso da validação ser bem sucedida, o cliente irá estar pronto para enviar os comandos desejados para o servidor. Caso contrário, a ligação é rejeitada.

Funcionamento geral do proxy

Partindo do príncipio que o proxy funciona como intermediário entre o cliente e o servidor, é necessário criar um processo ou thread para cada conexão. Neste caso, optámos pela criação de threads visto que evitamos assim a criação de uma shared memory.

Para além de uma thread por conexão, necessitámos de criar outra para que seja possível a inserção de comandos no terminal. Para conseguirmos listar todas as conexões atuais através do comando <SHOW>, decidimos criar uma lista ligada com a informação relativa a todas as conexões ativas no momento. Quando se inicia ou termina uma conexão, esta lista é atualizada. Em relação à funcionalidade de simular perdas de bytes durante um download, calculamos o número de bytes que é pretendido remover de acordo com a percentagem introduzida com o comando <LOSSES N>. Assim eliminamos, de forma aleatória, o número de bytes pretendido. Ou seja, alteramos a informação transmitida pelo servidor para o cliente, removendo N bytes. No que diz respeito ao comando <SAVE>, caso seja pretendido gravar a informação do download, o proxy para além de a transmitir para o cliente, irá guardá-la em memória.

Download de ficheiros

Quando é solicitado ao servidor um download de um ficheiro, o cliente transmite em primeiro lugar o nome do mesmo, pelo que, o servidor enviará uma mensagem de rejeição caso o ficheiro pretendido não se encontre na diretoria desejada. Após o cliente receber uma validação, comunica ao servidor o protocolo e se deseja ou não que a informação seja encriptada. Com esta mensagem, o servidor sabe, ao certo, qual o protocolo que deve utilizar para transmitir o ficheiro e se é desejada a encriptação do mesmo. O processo atribuído para manter a conexão activa irá abrir o ficheiro desejado, lendo-o byte a byte para um *buffer* de tamanho definido. Considerámos que seria interessante verificar sempre o sucesso ou insucesso do download pretendido, pelo que tomámos a decisão de implementar uma função simples de *checksum*, recorrendo ao último byte do mesmo buffer para indicar o valor desta função para cada conjunto de informação. Assim, é possível que o cliente, recorrendo à mesma função, verifique a integridade da informação

recebida. Isto será útil para que o cliente perceba, de forma automática, que houve perda de informação durante o download.

No caso em que é pretendido recorrer ao protocolo UDP para o download do ficheiro, havendo a possibilidade de existirem perdas de informação não é possível prever o fim do download, pelo que recorremos a uma solução que verifica se o canal de comunicação entre proxy e cliente tem informação para ser lida. Em caso afirmativo, o cliente lê sempre informação independentemente do tamanho da mesma. Caso não exista bytes para serem lidos, o cliente aguarda uma unidade de tempo para abortar leitura do canal. Caso após um segundo este canal continue vazio, o cliente dá por terminado o download. Evitamos, assim, o bloqueio do cliente na instrução *recvfrom*.

Encriptação

Como é pedido no enunciado, é necessário que a transferência de ficheiros possa ser encriptada pelo servidor. Utilizando o sistema de autenticação inicial, recorrendo à biblioteca *libsodium*, é possível criar uma chave que servirá para a encriptação por parte do servidor e de uma de desencriptação por parte do cliente. Estas chaves são criadas através das chaves publicas de cada um, o que faz com que o proxy não consiga ter acesso a nenhuma destas, impossibilitando, assim, a sua desencriptação.

Como encriptar um conjunto de bytes aumenta, geralmente, o seu tamanho, pensámos que seria mais adequado encriptar primeiro em causa e só depois proceder com a sua transmissão, recorrendo, novamente, à função de *checksum*. Após o download, removemos este ficheiro temporário. O cliente receberá o ficheiro encriptado e após o término da transferência, irá proceder à sua desencriptação, recorrendo à chave privada construída no início da conexão.

Listagem de ficheiros na diretoria

Para a listagem dos ficheiros que se encontram na diretoria do servidor, basta inserir no cliente o comando <LIST>. Esta instrução é enviada para o servidor, sendo que este responde com uma listagem do nome de todos os ficheiros possíveis para download juntamente com o tamanho dos mesmos. Decidimos ignorar todos os ficheiros que não têm qualquer conteúdo, já que caso fosse possível transferir ficheiros vazios haveria a probabilidade de existir um bloqueio do programa.

Terminação de conexões

É possível terminar a conexão de cliente - proxy - servidor através do comando QUIT. Este comando enviará uma mensagem para o proxy, que por sua vez reencaminhará a mesma para o servidor dando ordem para que a thread e processo responsáveis pela ligação terminem de forma correta.

Para controlar a limpeza de recursos, decidimos implementar ignorar o sinal SIGINT no cliente e reformular a resposta a este sinal por parte do servidor. Assim, asseguramos que só é possível terminar o servidor mal todas as conexões terminem, evitando terminações indesejadas. Para além disso, asseguramos que a *shared memory* inicializada é, devidamente, terminada.