Rede Overlay de anonimização do originador

Pedro Gomes[A84220], José Santos[A84288], and Henrique Paz[A84372]

Universidade do Minho - Comunicaçoes por computador - Grupo 3 - PL2

Abstract. Trabalho pratico numero dois da UC de comunicaçoes por computador que é centrado em desenvolver um tunel anonimizador com recurso ao protocolo UDP.

Keywords: Protocolo · AnonGateWay · Pacotes TCP/UDP

1 Introdução

Nesta secçao vamos explicar um pouco do trabalho pratico desenvolvido, tal como a estrutura do relatorio. O trabalho que desenvolvemos foi um tunel anonimizador em que o objetivo de desenhar e implementar uma rede Overlay de anonimização do originador. Os clientes de origem e os servidores de destino não precisam de ser implementados pois podem ser genéricos, por ex clientes e servidores HTTP, ou SSH ou outro tipo qualquer. De seguida vamos explicar a arquitetura adotada por nós para por em pratica a rede de anonimizaçao assim como o protocolo que implementamos em conjunto com o UDP para garantir a segurança da anonimizaçao, a ordenaçao dos pacotes e a multiplexagem do tunel UDP. Por fim mostramos a implementaçao e a conclusao.

2 Arquitetura da solução

O nosso projeto de rede anonimizador decidimos que seria feito na linguagem JAVA onde o ponto de inicilizaçao é a classe AnonGW que tem o metodo main e nesse mesmo metodo inicia uma instancia da classe Command.java que le o imput introduzido para a inicializaçao da rede anonimizadora e preenche a classe BaseArgsInfo, que é um singleton, com a informaçao do ip do target server, os overlay-peers e a porta do target server para ter essa informaçao a qualquer momento disponivel.
Depois de termos dado parse ao imput inicializamos 2 threads, uma para ficar a escuta de pacotes TCP, ou seja potenciais clientes, e a outra thread para ficar a escuta de pacotes UDP, ou seja de pacotes dos outros AnonGW.

No caso de receber um pacote TCP cria uma nova thread com a classe AnonGWServer que escolhe um AnonGW a sorte. Para cada cliente novo é gerada uma nova porta para os dois AnonGW comunicar unica e exclusivamente sobre esse cliente, ou seja cada cliente tem a sua porta. A porta 6666 é usada para comunicar a nova porta unica do cliente, isso acontece atravez da classe UDPPortMessage que diz qual a porta do cliente.

No caso de receber um pacote UDP, le esse mesmo pacote e inicia a classe AnonGWClient que trata da logica de ler do UDP socket a query do cliente e establecer a sua conecçao via TCP com o target server e dps enviar a resposta do server denovo pelo tunel UDP.

3 Especificação do protocolo UDP

Nesta secçao vamos explicar como fizemos o nosso protocolo para assegurar a entrega ordenada dos pacotes, a sua segurança e a multiplexagem do canal UDP

3.1 Formato das mensagens protocolares (PDU)

No nosso projeto temos duas maneiras de comunicar atravez do pacote UDP a primeira que é quando um novo cliente se conecta é com a classe UDPPortMessage que é enviada pelo port 6666 e serve para indicar ao segundo AnonGW a nova porta que vao usar para falar em relaçao a aquele cliente.

Depois de a porta especifica do cliente estar definida usamos a class UDPdata para mandar toda a informaçao que é necessaria (messagem do cliente/Servidor mais header), que consiste num array de bytes que é os dados que está a enviar, seja o pedido do cliente ou a resposta do servidor, alem disso tem uma flag em forma de bool a dizer se o pacote em questao é o ultimo ou nao e por fim tem um indice que é o numero do pacote.

4 Implementação

Para começar usamos as bibliotecas java.net.ServerSocket, java.net.Socket para ser possivel iniciar coneçoes via TCP e as bibliotecas java.net.DatagramPacket, java.net.DatagramSocket para iniciar coneçoes via UDP.

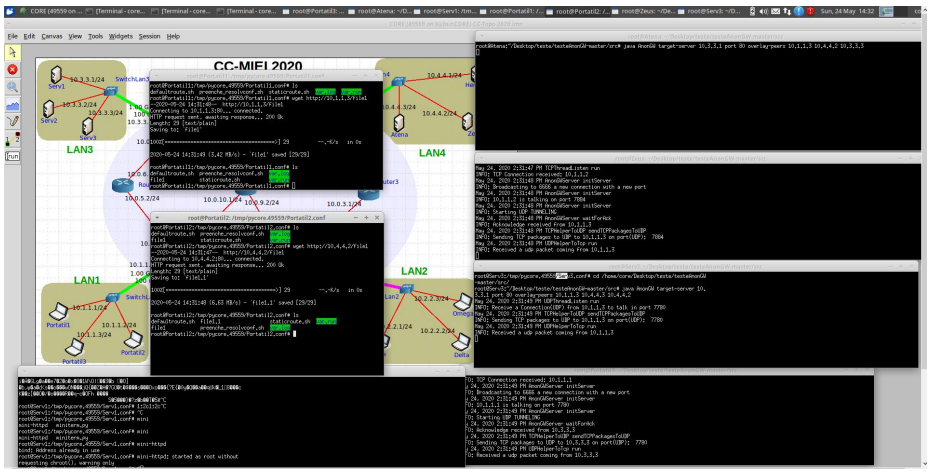
Na classe UDPdata que é usada para mandar os pacotes UDP fizemos a implementar a interface Serializable para ser possivel ser transformada de objeto em bytes e vice versa, e tambem na classe UDPPortMessage que serve para dizer em que port vao os dois AnonGW falar tambem usamos o Serializable.

Usamos a classe Thread do java para termos varias threads a fazer tarefas diferentes concorrentemente, que é o que acontece no AnonGW para podermos estar simultaneamente a escuta por pacotes UDP e TCP, isto é preciso visto que por exemplo o accept do TCP é bloqueante.

Para a incriptaçao dos dados para manter a segurança dos dados que o servidor ou o cliente envia usamos as bibliotecas javax.crypto.Cipher, javax.crypto.spec.SecretKeySpec, java.security.Key que servem para gerar chaves de encriptaçao, algoritmos de encriptaçao baseadas nessas chaves e por fim os dados encriptados com a ajuda do algortimo de encriptaçao.

5 Testes e resultados

Os testes que fizemos foi na topologia CORE em que iniciavamos 4 AnonGW em diferentes nodos e um servidor mini-http.De seguida criavamos um ficheiro no servidor com algum conteudo dentro e a partir de um portatil que nao fosse nenhum AnonGW fazimos a query do ficheiro um AnonGW a nossa escolha. Verificamos que a transferencia do ficheiro era feita com sucesso visto que o ficheiro estava no portatil do cliente com a informaçao certa.Nos tambem experimentarmos termos 2 portateis diferentes a pedir o mesmo ficheiro e funciona tambem corretamente como se pode ver na seguinte imagem.Como se pode ver na imagem o tempo de demora foi menos de que 1 segundo mas esse tempo está relacionado com o tamanho do ficheiro que queremos transferir, que neste caso era reduzido.



6 - Conclusões e trabalho futuro

Para concluir achamos que em primeiro lugar fizemos um trabalho positivo visto que o nosso software transfere com sucesso o ficheiro do servidor target expecificado atravez de nodos de anonimizaçao iniciados previamente.

Achamos com este trabalho que ganhamos muito conhecimento principalmente em relaçao aos protocolos de transport UDP e TCP o que para nós é bastante valioso, visto serem 2 protocolos muito usados na cada de transport na pilha protocolar.

Visto isto sabemos que este trabalho foi bastante importante para adequirir conhecimento para o futuro em relaçao ao funcinamento das comunicaçoes entre computadores e enventualmente para a nossa vida profissional.