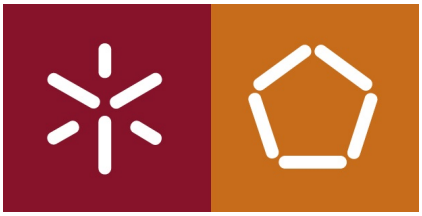

Universidade do Minho
Mestrado Integrado em Engenharia Informática
3ºano - 2º Semestre

Comunicações por Computador

Trabalho Prático Nº2 – Rede Overlay de anonimização do originador

Grupo 1 - PL4

a83732 – Gonçalo Rodrigues Pinto
a84197 – João Pedro Araújo Parente
a84829 – José Nuno Martins da Costa

20 de Maio de 2020

Conteúdo

Lista de Figuras

1 Introdução

O objetivo deste trabalho foi desenhar e implementar uma rede overlay de anonimização do originador. Para atingir esse fim utilizou-se a linguagem Java para implementar esta rede. Os clientes de origem e os servidores de destino não foram implementados pois foi nos indicado que estes poderiam ser genéricos, ou seja, podem ser clientes e servidores HTTP, ou SSH ou outros.

O único componente desenhado e implementado foi o gateway de transporte para a rede overlay de anonimização designado por AnonGW. A rede é formada por múltiplas instâncias desse mesmo servidor (AnonGW). Esses agentes recebem pedidos TCP vindos dos clientes e pedidos UDP vindos dos seus pares.

O objetivo principal foi, pois, desenhar um protocolo que funciona sobre UDP que garanta a entrega ordenada (pela mesma ordem recebida) e confidencial (usando cifragem do conteúdo) dos dados de uma ou mais conexões de transporte TCP (multiplexagem de clientes).

Neste trabalho é de realçar que não se lidou com a congestão da rede. Todos os dados recebidos de uma extremidade TCP são reenviados pelo túnel UDP e vice-versa.

2 Arquitetura da solução

Imaginando o cenário em que um cliente pretende interrogar um servidor. Em vez de efetuar a conexão de transporte TCP diretamente a esse servidor, denunciando assim o seu endereço IP de origem e deixando um rasto auditável nesse servidor, o cliente opta por usar a rede overlay de anonimização desenvolvida.

Esta rede é formada por um conjunto de gateways de transporte, designados *AnonGW*, espalhados pela rede, e tudo o que o cliente tem de fazer é dirigir a sua conexão para qualquer um dos nós *AnonGW* disponíveis escolhendo aleatoriamente um e faz a conexão tal como se estivesse a fazê-la diretamente para o servidor. O *AnonGW* contactado aceita a conexão e serve de primeiro intermediário poderia desde já conectar-se ao servidor, que já camuflaria a origem verdadeira do pedido.

Mas em vez disso escolhe também ele aleatoriamente um outro *AnonGW* presente na rede para acrescentar mais um nível de indireção. Entre eles estabelece-se uma sessão segura sobre UDP para maior versatilidade e eficiência. Todos os dados trocados entre os *AnonGW* são encapsulados num protocolo de Anonimização desenhado especificamente para o efeito apresentado no capítulo 3. Este protocolo basicamente implementa um túnel UDP anonimizador e finalmente cabe ao segundo *AnonGW* a tarefa de fazer realmente o pedido ao servidor.

O percurso inverso é usado de igual modo nas respostas do servidor. Todos os dados enviados pelo servidor são recebidos pelo *AnonGW* de destino através da conexão TCP (exposta), e reenviados pelo túnel UDP anonimizador até ao *AnonGW* de origem, que finalmente os envia ao cliente pela conexão TCP protegida (não exposta).

A privacidade é garantida pois todos os servidores *AnonGW* da rede overlay implementada esquecem os dados de todas as conexões efetuadas ou recebidas, não guardando nenhum registo em ficheiro.

3 Especificação do protocolo UDP

O sistema é composto por um único programa, designado por *AnonGW*, que escuta numa porta TCP (por pedidos de clientes) e numa porta UDP (por pedidos de outros *AnonGW*).

3.1 Formato das mensagens protocolares (PDU)

Para esconder acessos a um servidor, o servidor *AnonGW* tem que atender em TCP, tal como o servidor que pretende proteger, e que lhe é passado por configuração. Tem ainda de conhecer a lista restrita de pares com os quais pode estabelecer túneis e dos quais (apenas desses) aceita PDUs do protocolo criado enviados por UDP. O modo de funcionamento é o que foi apresentado no capítulo 2.

3.1.1 Encriptadas

De seguida, apresentamos o formato das mensagens protocolares encriptadas que são trocadas entre os diferentes *AnonGW* por UDP.

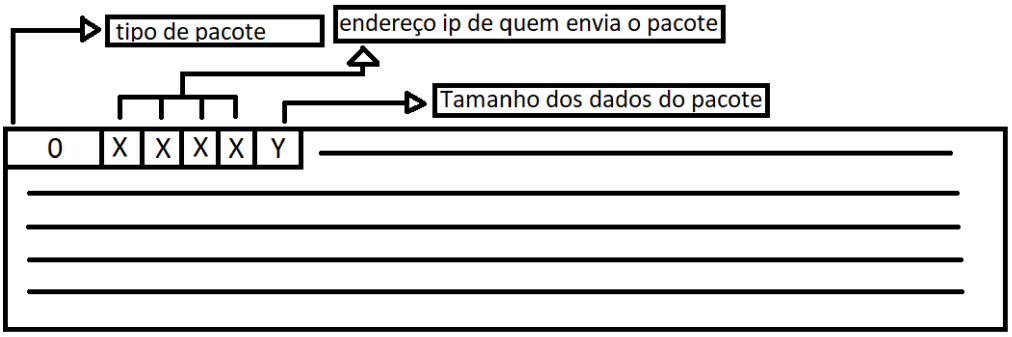

Figura 1: Mensagem protocolar encriptada que é trocada entre os diferentes *AnonGW*.

Nesta mensagem protocolar encriptada acrescentamos aos dados que são enviados pelo cliente ou servidor, as seguintes informações:

- **Tamanho dos dados do pacote** - indicamos o tamanho do pacote de forma a permitir quem receba os dados saber quanto espaço tem que alocar para receber o pacote na totalidade.
- **Endereço IP de quem envia o pacote** - indicamos o endereço IP do originador do pacote de forma a que o destino saiba quem mandou o pacote.

- **Tipo do pacote** - no início de cada pacote indicamos o tipo de pacote pois pode existir diferentes trocas de informação entre as diferentes entidades, de forma a distinguir os pacotes atribui-se diferentes identificadores numéricos, ou seja, um pacote identificado com:
  - número 0 - representa que o pacote trocado possui a chave pública de um par, sendo que este irá ser a resposta a um pacote com o identificador com o número 1;
  - número 1 - representa que o pacote trocado contém a chave pública de um par;
  - número 2 - representa que o pacote possui dados, prontos a desencriptar;
  - número 3 - representa que o pacote em questão está encriptado com a chave do $AnonGW$ que enviou o pacote, e que neste encontra-se a chave a ser usada para encriptar ou desencriptar a mensagens trocada;
  - número 4 - informa o $AnonGW$ no qual o cliente que estava a comunicar, que este já terminou a conexão;
  - número 5 - informa o $AnonGW$ no qual o servidor que estava a comunicar, que este já terminou a conexão;

3.1.2 Desencriptadas

Após desencriptar a mensagem ou antes de encriptar a mensagem acrescenta-se a seguinte informação aos dados a serem trocados entre o cliente e o servidor.

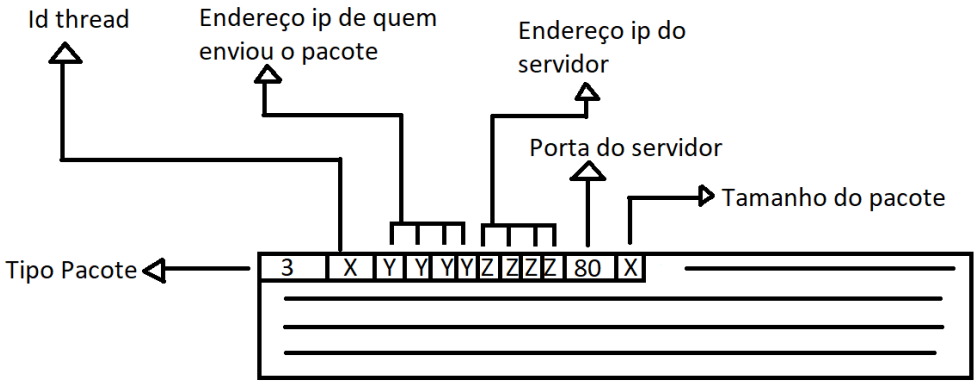

Figura 2: Mensagem protocolar desencriptada utilizada para enviar dados de um cliente para um servidor.

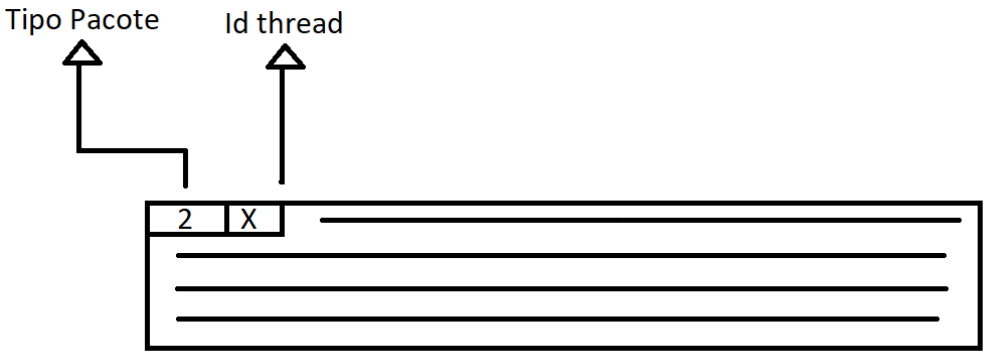

Figura 3: Mensagem protocolar desencriptada utilizada para enviar dados de um servidor para um cliente.

No caso da figura 3 não é necessário enviar tanta informação adicional no pacote pois a Thread que recebe este pacote, só necessita de observar qual é o seu endereço IP e o identificador da thread para onde envia o pacote.

3.2 Interações

Nas figuras abaixo apresentamos as iterações implementadas entre um cliente e um servidor:

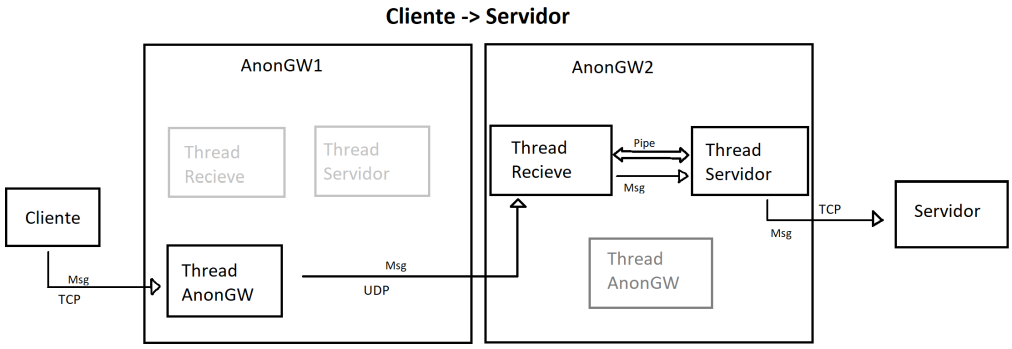

Figura 4: Esquema da comunicação entre o Cliente e Servidor.

A mensagem protocolar encriptada é trocada entre a Thread AnonGW do *AnonGW1* e a Thread Recieve do *AnonGW2*. A mensagem desencriptada é trocada entre a Thread Recieve e a Thread Servidor do *AnonGW2*.

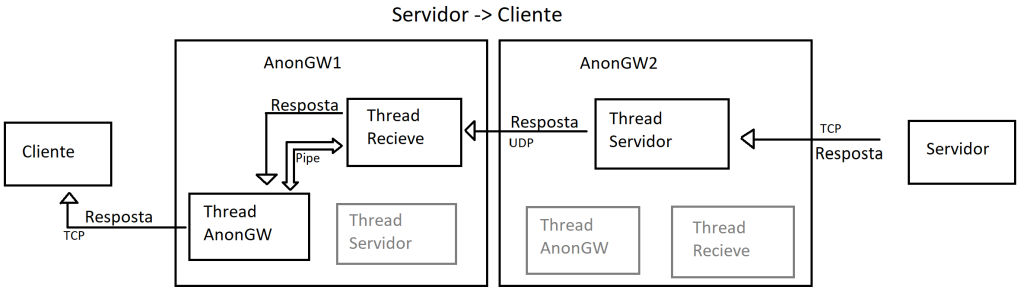

Figura 5: Esquema da comunicação entre o Servidor e Cliente.

A mensagem protocolar encriptada é trocada entre a Thread Recieve do *AnonGW1* e a Thread Servidor do *AnonGW2*. A mensagem desencriptada é trocada entre a Thread Recieve a Thread AnonGW do *AnonGW1*.

4 Implementação

4.1 Entrega ordenada e Multiplexagem de clientes

Quando um determinado cliente efectua uma conexão com um *AnonGW* é criada uma thread etiquetada com um identificador, este identificador percorre todos os pacotes que este *AnonGW* recebe ou envia, quando a Thread Recieve de um outro *AnonGW* recebe um pacote de dados (desencriptado) do tipo 3, igual ao da figura 2, é criada uma Thread Servidor (sendo que esta vai tratar de ler de um pipe os dados e comunicar estes ao servidor) e colocado os dados deste num pipe indexado pelo identificador da thread e o endereço IP do *AnonGW* origem, assim como o endereço IP e porta do servidor. Com isto, podemos garantir que se houver muitos clientes ao mesmo tempo a efetuar uma conexão a um *AnonGW* este vai poder processá-los corretamente assim como assegurar que todos os seus dados chegam por ordem ao servidor.

4.2 Confidencialidade

De forma a tornar as comunicações entre os diferentes *AnonGW* segura, decidiu-se que cada um possui uma chave pública e uma chave privada para encriptar e desencriptar os pacotes de dados, no entanto, no processo de implementação desta estratégia de segurança revelou-nos que a quantidade de dados que é possível encriptar e desencriptar é relativamente pequena, assim sendo, decidiu-se que cada *AnonGW* ia na mesma ter uma chave pública e privada, contudo acrescentou-se uma chave partilhada pelos *AnonGW* para trocar informação para isso utilizou-se as chaves públicas e privadas para combinar com os seus pares uma chave para encriptar e desencriptar única, sendo que esta chave é trocada com os pares de forma encriptada, com a sua chave privada. Na figura abaixo apresentamos um esquema como a comunicação de chaves é efectuada.

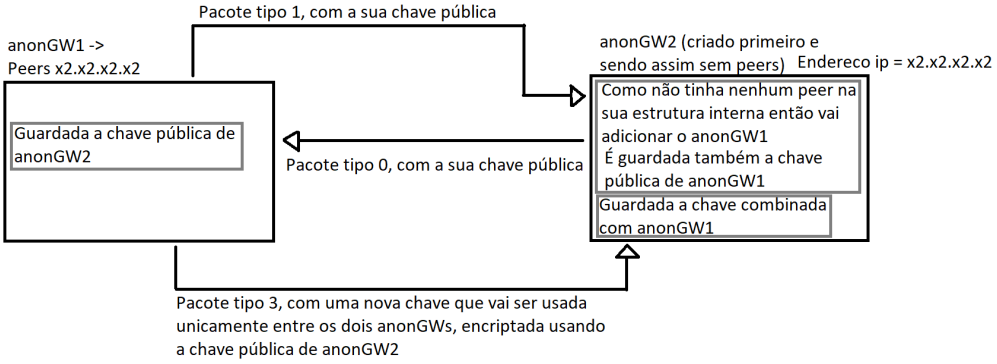

Figura 6: Esquema da comunicação entre *AnonGW*, na partilha das suas chaves.

5 Testes e resultados

Para o cenário de teste tal como nos foi recomendado tentou-se utilizar a máquina virtual xubuncore fornecida com o emulador CORE já instalado com a topologia CC-Topo-2020.imn que nos foi fornecida para criar um cenário de teste adequado contudo devido ao facto de a versão do Java, linguagem essa que foi utilizada para implementar a rede overlay de anonimização do originador, da máquina virtual ser antiga e não ser possível atualizar a mesma. Utilizou-se então um cenário teste nosso.

Decidimos então utilizar dois computadores, cada um com um *AnonGW*, junto de um servidor num e um cliente noutro, a figura abaixo serve para ilustrar o cenário de teste utilizado.

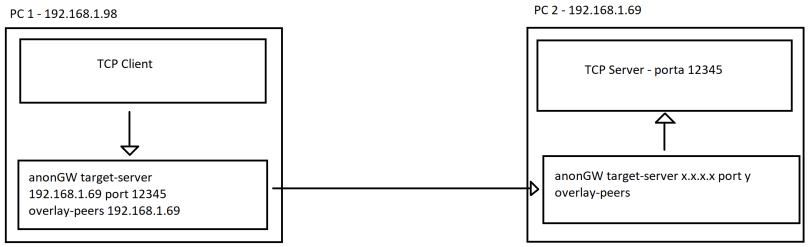

Figura 7: Cenário de teste utilizado.

Depois do cliente conectar-se envia uma mensagem ao *AnonGW* que se conectou, neste caso a mensagem é "login a a".

```
PS D:\Material Curso\3º ano\1º semestre\SD\Trabalho Pratico\SD_31-12-19\src>  & 'c:\Users\jnuno\.vscode\extensions\vscjava.vscode-java-debug
-0.26.0\scripts\launcher.bat' 'C:\Program Files\Java\jdk-13.0.1\bin\java.exe' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\jnuno\AppData\Roaming\
Code\User\workspaceStorage\f1c2d582627673e9c94c9e6be4fe5530\redhat.java\jdt_ws\src_e4e9397b\bin' 'Clients.Client'
login a a
```

Figura 8: Mensagem enviado pelo cliente ao *AnonGW*.

```
Pacote encriptado:
ØI\h╣¤3╞□D)"3^□Ø^□#□ℵ<,ºØ-,\*E¤C0z*╣Ø*4Ø□Ø□1╬═w*↓ □g,M□b⅋Rii[□
Dados desencriptados:
ØØ□¤Øb□¤ØE90;login a a
```

Figura 9: Dados encriptados e desencriptados recebidos pelo *AnonGW* que envia a mensagem ao servidor.

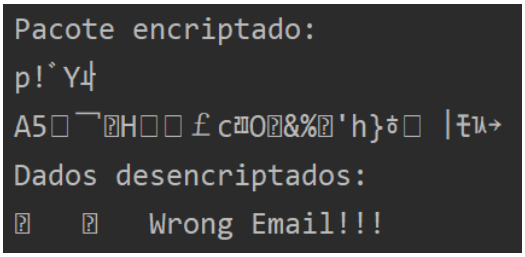

Figura 10: Dados encriptados e desencriptados recebidos pelo *AnonGW* que envia a resposta do servidor ao cliente.

6 Conclusões e trabalho futuro

O presente relatório descreveu, de forma sucinta, a resolução do trabalho proposto de desenho e implementação de uma rede Overlay de anonimização do originador.

Consideramos que os principais objectivos foram cumpridos.

Sentimos que a realização deste projecto consolidou os nossos conhecimentos de programação em Java como também expandi-los. Contudo realçamos que este trabalho alertou para a importância da segurança e das ameaças que existem à privacidade porque esta por si não existe pois qualquer atividade na rede pode ser potencialmente auditada, pelos registos de conexão dos servidores. Uma das formas de tentar proteger a privacidade passa por camuflar a verdadeira origem das conexões daí a importância deste trabalho em fornecer uma entrega ordenada, confidencial de uma ou mais conexões de transporte.

A nível do trabalho futuro poderíamos aprimorar o trabalho realizado com a introdução de controlo de perdas, permitindo a retransmissão de pacotes pedidos como também poderemos autenticar a origem, através da assinatura digital de cada PDU.

Em suma, esperamos que o trabalho realizado seja essencial, enriquecedor e fulcral para as nossas futuras carreiras.