Trabalho Prático Nº2 – Rede Overlay de anonimização do originador

Duração: 5 aulas (não consecutivas, ver calendarização das aulas práticas)

Contexto e Motivação

No capítulo da segurança o parente mais pobre é seguramente a privacidade. As ameaças à privacidade são bastas vezes encaradas com alguma ligeireza. Talvez porque a privacidade pode sempre ser vista por dois prismas. Ou realmente como um direito de cada cidadão de não ser expiado em todas as suas atividades online como utilizador de serviços sejam eles quais forem. Ou apenas como uma ameaça à segurança das comunidades, que permite esconder ações ilegais e criminosas que afetam a segurança de todos. A privacidade per si não existe, pois qualquer atividade na rede pode ser potencialmente auditada, pelos registos de conexão dos servidores *("logs")*. Uma das formas de tentar proteger a privacidade passa por camuflar a verdadeira origem das conexões. Esta abordagem está ilustrada na *figura 1*, embora com algumas nuances próprias.

O cenário ilustrado na *figura 1* é simples. Um cliente *(Origin)* pretende interrogar um servidor *(TargetServer)*. 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 uma Rede Overlay de Anonimização *(Anon).* A rede Overlay *Anon* é formada por um conjunto de *Gateways de Transporte (AnonGW)*, espalhados pela rede, e tudo o que o cliente *Origin* tem de fazer é dirigir a sua conexão para qualquer um dos nós *AnonGW* disponíveis. Escolhe aleatoriamente um e faz a conexão tal como se estivesse a fazê-la diretamente para o *TargetServer*. O *AnonGW* contactado aceita a conexão e serve de primeiro intermediário. Podia desde já conectar-se ao TargetServer, que já camuflaria a origem verdadeira do pedido, mas em vez disso escolhe também ele aleatoriamente um outro AnonGW 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 *(Anon Protocol)* desenhado especificamente para o efeito. Este protocolo na prática implementa um túnel UDP anonimizador. Finalmente cabe ao segundo AnonGW a tarefa de fazer realmente o pedido ao TargetServer.

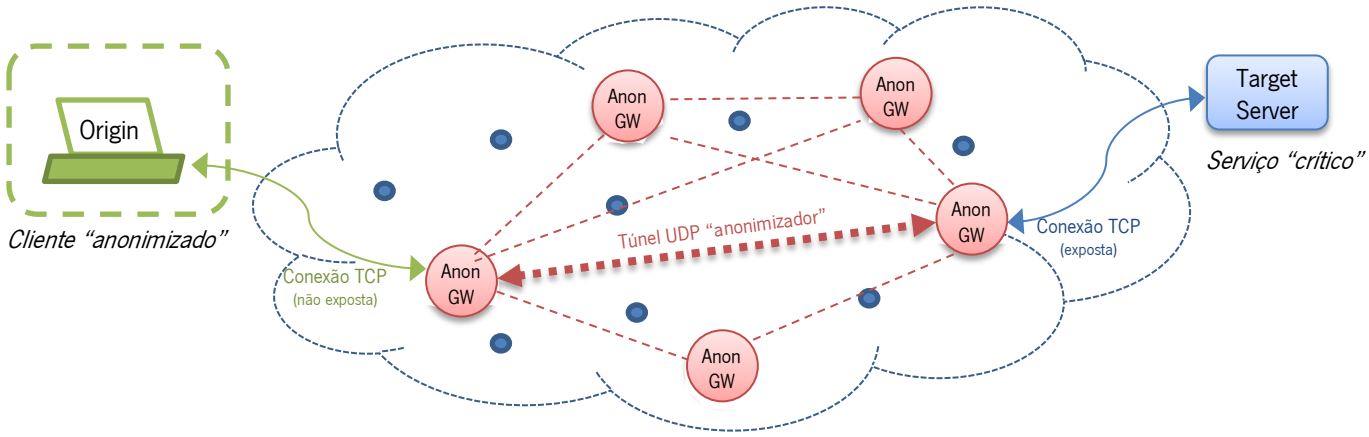

Figura 1

Todos os dados enviados pelo cliente *Origin* na conexão TCP (não exposta) são recebidos pelo primeiro *AnonGW* e imediatamente enviados pelo túnel UDP anonimizador para um segundo *AnonGW*, que por sua vez os faz chegar numa nova conexão TCP (exposta) ao *TargetServer*. O percurso inverso é usado de igual modo nas respostas do servidor. Todos os dados enviados pelo TargetServer 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 *Origin* pela conexão TCP protegida (não exposta).
A privacidade é garantida se e só se todos os servidores *AnonGW* da rede *Overlay* esquecerem os dados de todas as conexões efetuadas ou recebidas, não guardando nenhum registo em ficheiro.

Objetivos

O objetivo deste trabalho é desenhar e implementar uma rede *Overlay* de anonimização do originador, tal como ilustrado na figura 1. Os clientes de origem e os servidores de destino não precisam de ser implementados pois podem ser genéricos (ex: clientes e servidores HTTP, ou SSH ou outros). O único componente que é necessário desenhar e implementar é o gateway de transporte para a rede overlay de anonimização designado por AnonGW. A rede será formada por múltiplas instâncias desse mesmo servidor (AnonGW). Esses agentes devem receber pedidos TCP vindos dos clientes e pedidos UDP vindos dos seus pares. O objetivo principal é, pois, desenhar um protocolo para funcionar 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). Não se pretende lidar com a congestão da rede. Quanto às perdas que possam ocorrer, a recuperação de um pacote por retransmissão é apenas um requisito opcional considerado menos prioritário. Todos os dados recebidos de uma extremidade TCP devem ser reenviados pelo túnel UDP e vice-versa.

Descrição

A *figura 2* mostra uma visão mais detalhada do sistema. O sistema á composto por um único programa, designado na ficura por AnonGW, que escuta numa porta TCP (por pedidos de clientes) e numa porta UDP (por pedidos de outros Anon GW).

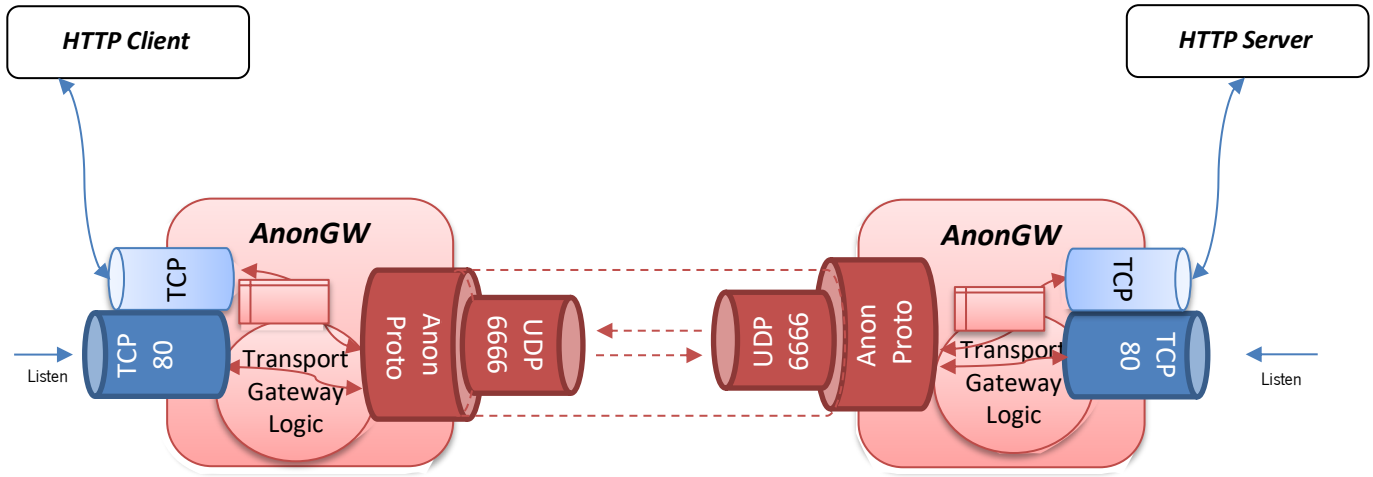

Figura 2

Para esconder acessos a um servidor HTTP, como descrito na *figura 2*, o servidor *AnonGW* tem de atender em TCP na porta 80, tal como o *TargetServer* que pretende proteger, e que lhe será dado por configuração. Tem ainda de conhecer a lista restrita de pares com os quais pode estabelecer túneis e dos quais (e apenas desses) aceita PDUs do protocolo *Anon Proto* enviados por UDP para a porta 6666. O modo de funcionamento é o que já foi descrito. Sempre que receber uma nova conexão TCP na porta 80 deve selecionar aleatoriamente um dos seus pares com o qual vai trocar dados em UDP e manter com ele uma sessão confidencial na qual troca todos os dados da conexão TCP do cliente (e só esses). Sempre que receber um pedido por UDP na porta 6666 de um dos seus pares, deve procurar estabelecer uma conexão TCP com o *TargetSever* para o qual foi configurado.

O parâmetros de configuração podem ser passados na linha de comando, como no exemplo que se segue (mera sugestão):

```
$ anonGW  target-server 10.3.3.1 port 80 overlay-peers 10.1.1.2 10.4.4.2 10.4.4.3
```

Cenário de Teste

Recomenda-se o uso do emulador **CORE** e a topologia **`CC-Topo-2020.imn`** para criar um cenário de teste adequado. Como cliente pode-se usar o **`wget`** ou um browser qualquer. Como servidor pode-se usar por exemplo o servidor HTTP **`mini-http`** (usado no primeiro trabalho prático) ou outro servidor HTTP qualquer. Na verdade, é suposto que o serviço seja transparente e funcione com qualquer aplicação que use TCP, por isso é indiferente qual o cliente e qual o servidor para testes. Executar pelo menos 4 instâncias do *AnonGW* em 4 nós distintos.

Especificação de um teste base minimalista (Teste #0):

- Iniciar a topologia `CC-Topo-2020.imn` no emulador CORE
- Usar o sistema `Serv1` (10.3.3.1) como servidor HTTP "critico" (*TargetServer*), com o `mini-http` a funcionar na porta 80
- Usar o sistema `Portatil1` para cliente "anonimizado (*Origin*) abrindo uma bash para execução do `wget` e testando inicialmente com conexão direta ao *TargetServer*: `wget http://10.3.3.1/file1`
- Executar 4 instâncias do *AnonGW* nas máquinas `Portátil3`, `Antenas`, `Zeus` e `Serv3` com os parâmetros certos para se conhecerem uns aos outros e saberem que todos protegem o servidor `Serv1` na porta 80;
- Testar no sistema `Portatil1` com `wget http://10.1.1.2/file1`

Será que também funciona com o ssh? Experimente.

FASE 1: Protótipo só com componentes TCP (2 aulas)

Sugestões para esta fase:

- Programa principal com os parâmetros adequados
- Implementação das componentes TCP (cliente/servidor)
- Definição das estruturas de dados internas
- Implementação da lógica de gateway de transporte genérico
- Teste com apenas um AnonGW (recebe conexões em TCP e liga-se de imediato ao servidor de destino)

FASE 2: Desenho e implementação do protocolo *Anon Protocol* sobre UDP (3 aulas)

Sugestões para esta fase:

- Especificar o protocolo (formato PDU: sintaxe e semântica):
  - requisitos mínimos: i) entrega ordenada (os dados recebidos de um extremo são entregues no outro extremo pela mesma ordem com que foram recebidos); ii) confidencialidade (cifragem do conteúdo para evitar espionagem); iii) multiplexagem de clientes (capacidade de lidar ao mesmo tempo com mais que uma conexão TCP, separando os fluxos convenientemente);
  - requisitos opcionais: controlo de perdas (retransmissão de pacotes perdidos), autenticação de origem (assinatura digital de cada PDU);
  - não requisitos: controlo de congestão (funcionamento independente da carga da rede)
- Desenhar e implementar os PDU Anon Protocol
- Implementação das componentes UDP e Anon Protocol
- Teste com dois AnonGW

Planeamento

O desenho e implementação deste serviço devem ser feitos ao longo das aulas práticas de CC, quer nas explicitamente dedicadas ao trabalho, como em todas as outras onde existir tempo livre. As fases de desenvolvimento são as seguintes:

FASE 1: As primeiras 2 aulas reservadas para o TP2
FASE 2: As últimas 3 aulas reservadas ao TP2

Relatório

O relatório deve ser escrito em formato de artigo com um máximo de 10 páginas (recomenda-se o uso do formato LNCS - *Lecture Notes in Computer Science*, instruções para autores em http://www.springer.com/computer/lncs?SGWID=0-164-6-793341-0). Deve descrever o essencial do desenho e implementação com a seguinte estrutura recomendada:

- Introdução
- Arquitetura da solução
- Especificação do protocolo UDP
  * formato das mensagens protocolares (PDU)
  * interações
- Implementação
  * detalhes, parâmetros, bibliotecas de funções, etc.
- Testes e resultados
- Conclusões e trabalho futuro

Regras de submissão

Cada grupo deve fazer a submissão (*upload*) do trabalho, usando a opção de "troca de ficheiros" associada ao seu grupo nos "Grupos" do Blackboard (*elearning.uminho.pt*), da seguinte forma:

- **CC-TP2-PL***<Turno>***G***<Grupo>***.pdf** para o relatório
- **CC-TP2-P***<Turno>***G***<Grupo>***Codigo.zip** ou **CC-TP2-P***<Turno>***G***<Grupo>***Codigo.rar** para a directoria com o código desenvolvido (devidamente documentado)

sendo *<Turno>* é o nº do turno e *<Grupo>* o nº do grupo (e.g. CC-TP2-PL1-G1.pdf, CC-TP2-PL1-G1-Codigo.zip para o grupo 1 do PL1)