Trabalho Prático Nº0 – Emulador de Rede e Analisador de Tráfego (*Sniffer*)

Duração: 1h

Este trabalho deve ser realizado com recurso à máquina virtual **XubunCORE_7_5** que está disponibilizada em http://marco.uminho.pt/ferramentas/CORE/xubuncore.html (user: *core* password: *core*)

1. Objetivos

O objetivo deste trabalho é a familiarização com as ferramentas que vão ser usadas nos trabalhos práticos.

2. Introdução

Para além de compreender os conceitos teóricos subjacentes, um engenheiro informático deve ser capaz de conceber topologias de rede, configurar os diversos equipamentos que a integram e verificar o seu correto funcionamento. Uma outra componente importante é o desenvolvimento de programas que integrem os diversos componentes nos vários níveis da pilha protocolar.

Estas competências, para poderem ser adquiridas, dependem da existência de Laboratórios onde existam os diversos tipos de equipamento que podem integrar uma rede tais como *routers*, *switches*, *hubs*, *hosts* (servidores e clientes) através de diferentes formas de interligação.

Para alguns trabalhos, a exemplificação de determinadas técnicas, protocolos ou tecnologias torna indispensável o uso de uma topologia de rede de determinada dimensão. Atendendo à rápida evolução da área de estudo, o uso de simuladores de redes para investigação e para ensino é prática comum. Todavia, num simulador normalmente utiliza-se um modelo simplificado da realidade e os conceitos e a experiência prática acumulada não podem ser transportados para os equipamentos reais com muita facilidade.

Com o desenvolvimento do conceito de virtualização e a implementação de máquinas virtuais com as mesmas potencialidades dos sistemas de computação, dos sistemas operativos mais populares e dos próprios equipamentos de rede deu-se um passo decisivo para a emulação de redes.

Um dos problemas tradicionais com a virtualização é a falta de escalabilidade, particularmente visível quando o número de máquinas virtuais aumenta. O problema complica-se mais quando se pretende ter uma topologia de rede com um conjunto heterogéneo de máquinas virtuais. Este problema pode ser eventualmente minimizado através de uma implementação leve, isto é, mais simplificada, da máquina virtual.

Neste contexto, apresenta-se como plataforma de apoio às aulas a plataforma CORE - *The Common Open Research Emulator*. O CORE proporciona um ambiente de emulação *open-source* que usa técnicas existentes de virtualização de sistemas operativos para construir redes com fios e sem fios onde os protocolos e as aplicações podem correr sem qualquer modificação. Sendo um emulador em tempo real, essas redes podem estar conectadas a redes reais estendendo os ambientes de teste a equipamento real. A arquitetura CORE inclui uma GUI para desenhar facilmente topologias de rede, uma camada de serviço para instanciar máquinas virtuais e uma API para orquestrar tudo isto. As topologias podem ser também criadas através de *scripts* em Python. O CORE permite trabalhar com topologias de múltiplos nós, contudo o número de nós virtuais suportado depende da capacidade computacional do sistema em questão.

Assim, um emulador de redes como o CORE permite emular a topologia de uma rede num computador pessoal. Desta forma, pode-se estudar o funcionamento da rede e correr nos diversos nós da topologia o mesmo software que corre na rede real. Esta situação permite que a experiência acumulada possa ser transportada com facilidade para redes com equipamentos de rede reais.

Como complemento ao emulador de redes CORE, será usada também uma ferramenta que permite a captura e a análise de tráfego que circula nas redes de computadores, vulgarmente designada Analisador de Tráfego (ou *Sniffer*). Um *sniffer* de rede permite analisar em deferido tráfego capturado em tempo real.

O objetivo deste trabalho preliminar é precisamente a familiarização com estas duas ferramentas.

3. O Emulador de Rede: CORE

O CORE (*Common Open Research Emulator*) é uma ferramenta que permite criar vários cenários de rede através da definição de uma topologia e da configuração de diversos equipamentos de interligação (e.g., *routers, switches, hubs*) ou equipamentos finais (*hosts*). Para além disso, é possível ligar a rede emulada à rede real.

O CORE é bastante flexível, escalável e simples de usar, permitindo também correr "código real" nos equipamentos definidos.

A plataforma CORE inclui uma interface gráfica que permite definir com facilidade a topologia de rede e configurar os equipamentos activos da rede (*routers*, *switches*, *access points* (APs), etc.). Esta plataforma foi desenvolvida por um grupo de técnicos da Boeing, usando como ponto de partida uma ferramenta (*IMUNES*) desenvolvida na Universidade de Zagreb. Utiliza ou FreeBSD ou Ubuntu como sistema base e encaminhamento baseado nos sistemas *Quagga/Zebra* para os routers virtuais.

3.1 Procedimento experimental

Verifique se tem o CORE instalado no seu computador. Caso isso não aconteça pode utilizar a máquina virtual disponibilizada para o efeito no seguinte link:

- http://marco.uminho.pt/ferramentas/CORE/xubuncore.html

Após a instalação, teste o seu funcionamento, abrindo um terminal e executando o comando **core-gui** na linha de comandos.

1- Relativamente aos menus da GUI do CORE, verifique (se necessário, recorra ao manual do CORE):

   a. Que tipo de equipamentos se podem interligar em topologias de rede?

   b. Quais as diferenças básicas entre um *hub* e um *switch*?

   c. Quais as diferenças entre um *switch* e um *router*?

   d. Quais são as possibilidades de configuração nos *links* de ligação física?

2- Defina em modo de edição uma topologia com quatro *routers* como nós. Faça uma ligação do nó 1 para o nó 2, deste para o nó 3, e deste para o nó 4.

   a. Verifique que são atribuídos automaticamente endereços de rede IPv4 e IPv6 aos vários nós.

   b. Inspecione qual o débito nominal das ligações que interligam os nós? Coloque esse débito a 10 Mbps.

   c. Passe para o modo de emulação. Faça um clique duplo no nó 1 para verificar que processos estão em execução. Identifique que processos estão a correr.

   d. O *ping* é um utilitário que permite testar a conectividade IP entre dois nós na topologia de rede. Verifique se há conectividade entre o nó 1 e o nó 4. Que conclui?

3- A partir do modo de edição considere modificar a topologia de rede da seguinte forma: (i) remova a ligação entre o nó 2 e o nó 3 e entre este e o nó 4; (ii) insira um hub; (iii) ligue o nó 2, o nó 3 e o nó 4 a esse hub; (iv) ligue um servidor ao nó 1 e (v) dois clientes ao nó 4.

   a. Verifique as alterações ocorridas a nível de endereçamento quando introduziu o *hub* na topologia da rede.

   b. Teste a conectividade entre o nó 7 e o servidor.

   c. Estabeleça uma ligação *ssh* ao servidor a partir do nó 7.

4. O Analisador de Tráfego: Wireshark

Um analisador de tráfego ou *sniffer*, e.g. Wireshark (www.wireshark.org), é uma ferramenta para observação das mensagens protocolares trocadas entre duas entidades protocolares. O *sniffer* permite capturar os pacotes que circulam na rede enviados e recebidos através da interface de rede do seu computador pessoal. O *sniffer* apenas captura tráfego, não gerando tráfego adicional para a rede.

Como ilustrado na Figura 1, um *sniffer* é software adicional que inclui dois componentes principais, a captura e a análise de pacotes. A biblioteca de captura de pacotes obtém uma cópia de todas as tramas (ou *frames, e.g. Ethernet*) que são enviados e recebidos numa dada interface de rede.

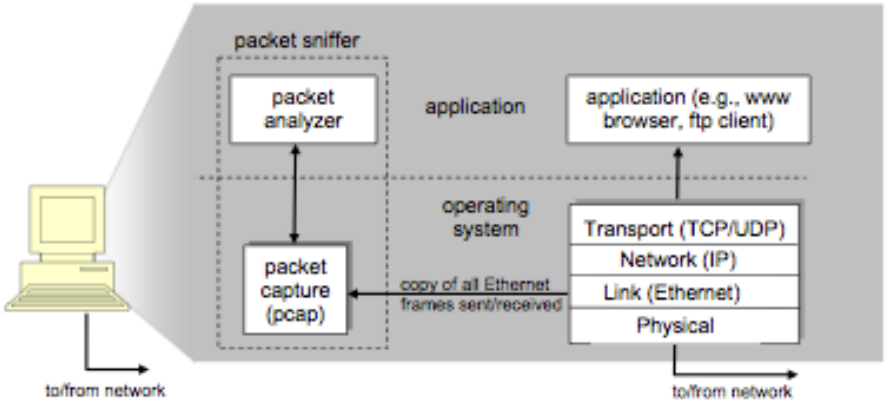

Figura 1: Estrutura típica de um *sniffer*

A comunicação entre sistemas remotos traduz-se na troca de mensagens geradas por protocolos de alto nível (normalmente associados a aplicações ou serviços dos utilizadores), que sofrem um processo de encapsulamento protocolar até ao nível da tecnologia de rede disponível. O conceito de encapsulamento protocolar, explicado nas aulas teóricas, é ilustrado na Figura 2.

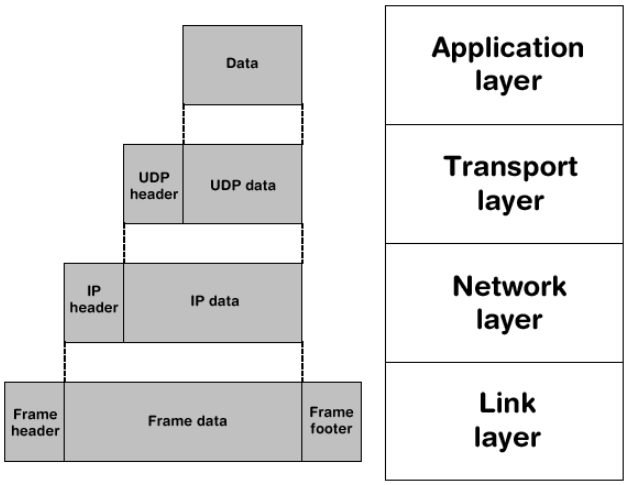

Figura 2: Encapsulamento protocolar

O segundo componente do *sniffer* é um analisador de pacotes que mostra o conteúdo de todos os campos do pacote numa mensagem protocolar. Para esse efeito o analisador de protocolos tem que compreender a estrutura das mensagens trocadas pelos protocolos.

Por exemplo, na troca de informação entre um cliente e um servidor Web, se os pacotes trocados forem capturados, o analisador de tráfego tem de perceber o formato da trama (*frame)* Ethernet (camada 2), do datagrama IP (*Internet Protocol)* (camada 3) dentro da trama, do segmento TCP (*Transmission Control Protocol*) (camada 4) dentro do datagrama e, por fim, da unidade protocolar HTTP dentro do segmento TCP (camada de aplicação).

Quando se corre o *Wireshark*, após uma captura de tráfego, é mostrada uma interface equivalente à apresentada na Figura 3.

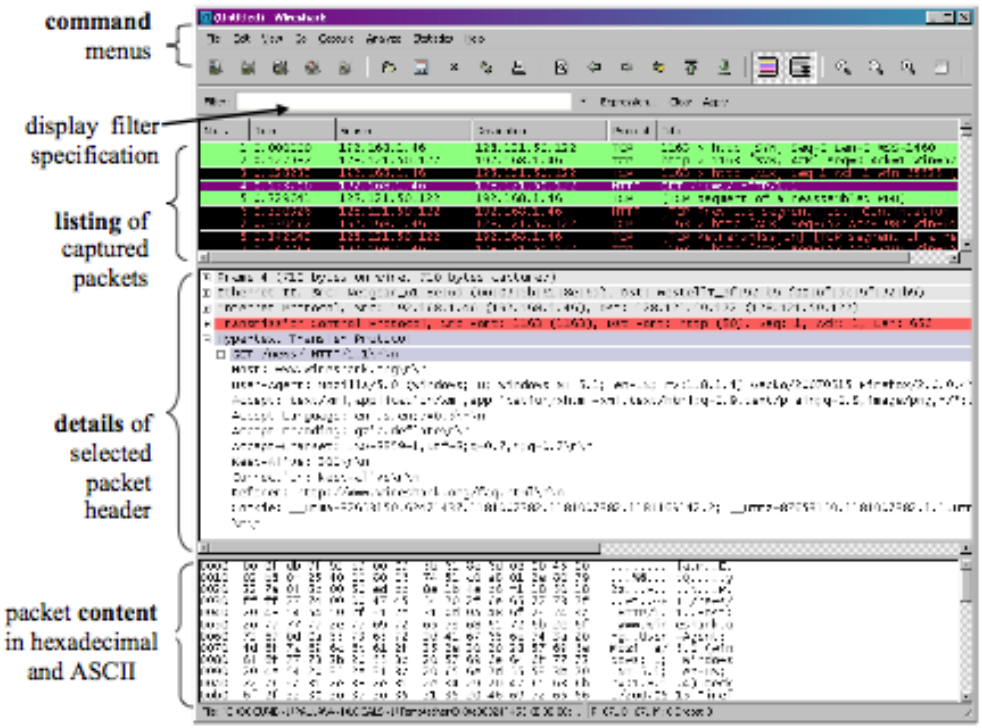

Figura 3: Interface gráfica do Wireshark

A interface do Wireshark tem cinco componentes principais:

- O menu de comando onde se destacam o *File* e o *Capture*. O *File* permite guardar uma determinada captura de tráfego ou abrir um ficheiro com informação previamente armazenada. O menu *Capture* permite iniciar e terminar a captura de tráfego.
- A janela de listagem de pacotes mostra uma linha com um sumário de cada pacote capturado, incluindo um número de pacote (atribuído pelo *Wireshark*), tempo de captura, origem, destino, protocolo, tamanho e tipo de informação.
- A janela de detalhe de cabeçalhos de pacote disponibiliza detalhes sobre o pacote selecionado na janela anterior. Podem-se obter detalhes sobre a trama Ethernet, o datagrama IP e os protocolos de nível superior.
- A janela do conteúdo do pacote mostra o conteúdo inteiro do pacote capturado.
- Junto do topo da Wireshark GUI existe um campo de filtragem dos pacotes a visualizar, onde podem ser colocadas expressões que permite filtrar os dados capturados. Podem-se definir filtros variados tais como definir o nome do protocolo, o endereço de origem e/ou destino, etc..

Assumindo que o seu computador está com acesso à Internet:

1. Ative um *web browser*.
2. Arranque o *wireshark* na sua máquina nativa. Deve obter uma janela inicial, similar à da Figura 3.
3. Escolha *Capture* *Options*. Em *Display Options* desative a opção *Hide Capture Info Dialog*. Se o seu computador tiver mais que uma interface deve selecionar a pretendida (assuma a selecionada por defeito pelo Wireshark).
4. Inicie a captura de tráfego. Deverá aparecer uma janela com o sumário da captura de pacotes realizada. Pode também explorar a opção *Statistics* para verificar os tipos de protocolo envolvidos no tráfego capturado.
5. Com o *wireshark* a correr, aceda à página www.fccn.pt a partir o seu browser. Pare a captura.
6. As mensagens HTTP trocadas com o servidor (e muito outro tráfego) vão aparecer na captura realizada. Filtre o tráfego *http* para facilitar a análise.

7. Selecione a primeira mensagem HTTP GET. Inspecione o processo de encapsulamento protocolar, observando a informação envolvida em cada um dos níveis (trama Ethernet, datagrama IP, transporte, etc.).

8. Identifique quais os endereços IP em uso na sua máquina e na máquina destino (servidor www.fccn.pt).

Referências

- J. Ahrenholz, Comparison of CORE Network Emulation Platforms, Proceedings of IEEE MILCOM Conference, 2010, pp.864-869.
- Manual do CORE, https://coreemu.github.io/core/,
- Manual do *Wireshark*, https://www.wireshark.org/#learnWS.