UC Redes de Computadores
ICT/UNIFESP
19/01/2020

Trabalho 2: Capturando mensagens HTPP com Wireshark

Observações:

- O trabalho poderá ser realizado em grupo de até 3 alunos.
- Verifique em anexo os comandos de interesse.
- A submissão no classroom estará aberta até **02/02/21**.
  - Entregável:
    - Video-relatório. Suba o vídeo em sua nuvem/youtube e envie somente o link do video (NÃO envie o arquivo de video pelo classroom).
- Requisitos do video-relatório:
  - Tempo esperado para o video: ~20 min.
  - Descrever as contribuições de cada integrante do grupo no desenvolvimento do trabalho.
  - Apresentar uma auto-avaliação do grupo sobre o trabalho realizado.

Neste laboratório, iremos explorar vários aspectos do protocolo HTTP. Esta é uma atividade prática sugerida no livro: James F. Kurose, Keith W. Ross; "Redes de Computadores e a Internet: uma abordagem Top-down"; Edição 5; Editora Pearson; 2010.

1. Interação básica requisição/resposta HTTP

Vamos começar baixando um arquivo HTML muito simples, o qual não contém objetos web incorporados. Então, faça o seguinte:

a) Abra o browser.
b) Limpe o cache.
c) Abra o *sniffer* Wireshark (caso não o tenha instalado em sua máquina, acesse `https://www.wireshark.org/`)
   - Pelo terminal: digite "`sudo wireshark`"
d) Configure a filtragem e inicie a captura: vá em *Capture* → *Options*, no campo *Interface* selecione a interface de rede ethernet ativa na máquina, e depois clique no botão *Start*.
e) Na barra do filtro, insira "`http`" (sem aspas) para que apenas mensagens deste protocolo sejam apresentadas na janela de listagem de pacotes.
f) No browser, digite na barra de endereços o nome (FQDN) de uma página estática. Um exemplo é o site:
   `www.pudim.com.br`
g) Pare a captura de pacotes. Vá em Capture → Stop.

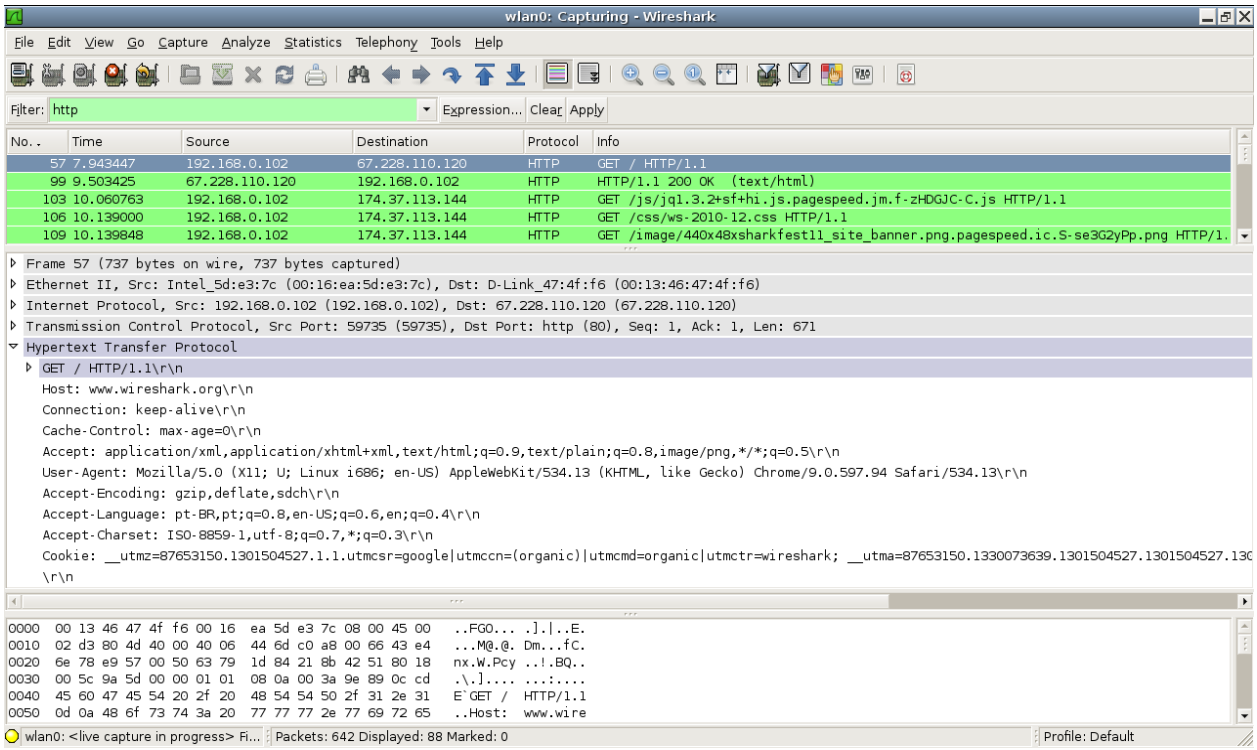

Figura 1: Screenshot da execução do sniffer wireshark.

A Figura 1 mostra a janela de listagem de pacotes de duas mensagens HTTP capturadas: a mensagem de GET (a partir de seu browser para o servidor web x.x.x.x) e a mensagem de resposta do servidor para seu browser. A janela de conteúdo mostra detalhes da mensagem selecionada (neste caso, a mensagem HTTP GET, que está realçada). O Wireshark exibe todo o encapsulamento:

> *Lembre-se que a mensagem HTTP é encapsulada dentro de um segmento TCP, que é encapsulado dentro de um datagrama IP, que é encapsulado dentro de um quadro (frame) Ethernet.*

Para omitir dados não-HTTP, na janela de exibição de detalhes dos protocolos encapsulados em um pacote, retraia a exibição dos protocolos Ethernet, IP e TCP. Clique no canto esquerdo da linha do HTTP para expandi-lo, exibindo os dados do seu cabeçalho.

De acordo com os dados da mensagem de requisição "`GET / HTTP`" e da respectiva resposta, **responda as seguintes perguntas** (*indique onde na mensagem você encontrou as informações que respondem as perguntas*):

1. A versão do HTTP são as compatíveis entre cliente e servidor?
2. Quais os idiomas (se houver) o seu browser suporta?
3. Qual é o endereço IP do cliente? E do servidor?
4. Qual é o código de status retornado pelo servidor?
5. Quando foi a última modificação no arquivo HTML que você obteve do servidor?
6. Quantos bytes de conteúdo estão foram enviados do servidor para o seu browser?
7. Verificando os dados brutos na janela que mostra os bytes do pacote, você consegue identificar dados que não estão exibidos na janela de detalhes do pacote (janela no meio)? Justifique.

2. Interação GET/response do HTTP Condicional

Vimos em aula a existência de uma entidade no lado do cliente chamada "Cache Web", o qual mantém uma cópia em cache dos objetos mais requisitados. A maioria dos browsers realiza cache de objetos localmente. Dessa forma, quando existe uma cópia em cache local do objeto requisitado, o browser envia uma requisição de GET condicional ao servidor, informando o tempo da última modificação do objeto que possui. Vamos verificar isso na prática.

Antes de executar os passos abaixo, certifique-se que o cache do seu browser está vazio (no Firefox, selecione *Preferences → Privacy & Security → Cookies and Site Data → Clear Data*).

Responda:

8. Acesse a página estática. Verifique o conteúdo da requisição "GET / HTTP" a partir de seu browser para o servidor. Há um campo "IF-MODIFIED SINCE" na linha do HTTP GET?
9. Inspecione o conteúdo da resposta do servidor. Ele retorna explicitamente o conteúdo do arquivo? Por que?
10. Agora, acesse novamente a página estática. Inspecione o conteúdo da segunda solicitação "GET / HTTP". Há um campo "IF-MODIFIED-SINCE" linha no HTTP GET? Se sim, quais as informações contidas nele?
11. Qual é o código de status retornado pelo servidor em resposta ao segundo GET? Será que o servidor retorna explicitamente o conteúdo do arquivo? Explique.
12. Qual a vantagem do uso de cache web e qual a sua limitação?

3. Recuperando grandes objetos web

Nos exemplos até agora, os documentos HTML foram simples e recuperados a partir de arquivos pequenos. Vamos ver o que acontece quando baixamos objetos grandes com HTTP.

Faça o seguinte:

h) Abra o browser, e certifique se o cache está limpo.
i) Abra o Wireshark.
j) Acesse uma página estática que possui ao menos um objeto web grande, de algumas centenas de KBytes ou alguns MBytes. Por exemplo,
`http://www.planalto.gov.br/ccivil_03/constituicao/constituicao.htm`
k) Após a página ser exibida por completo no seu browser, pare a capturar pacotes do Wireshark, e insira "http" no campo de filtro de pacotes.

Perceba que nesta resposta HTTP, o *payload* é todo o arquivo HTML solicitado. No entanto, o arquivo HTML é grande (> 1 MBytes) para ser acomodado em um único pacote (lembre-se que o MTU típico geralmente é 1500 Bytes).

Responda:

13. Quantas mensagens de solicitação GET para o objeto `constituicao.htm` foram enviadas pelo seu browser? Quantas respostas HTTP foram recebidas?
14. Busque por uma linha com a entrada "Reassembled TCP" no wireshark. Explique o que isso significa e mostre quantos segmentos TCP foram necessários para transportar uma única

resposta HTTP.

15. Qual é o código de status e a frase associada com a resposta para a requisição GET?

4. Documentos HTML com outros objetos web embutidos

Até agora vimos como o Wireshark exibe o tráfego de pacotes capturados para arquivos HTML grandes, agora veremos o que acontece quando seu browser baixa um html com com objetos web embutidos, ou seja, um arquivo que inclui objetos que podem estar armazenados no servidor e/ou em outro(s) servidor(es). No exemplo abaixo, os arquivos são imagens.

Faça o seguinte: repita os passos *h* ao *k* do exercício anterior.

Responda:

16. Quantas mensagens HTTP GET foram enviadas pelo browser? Para quais endereços IP elas foram enviadas?

17. Mostre (através de campos dos protocolos da pilha) se as requisições e respostas ocorreram com HTTP persistente ou não-persistente. Explique.

5. Campos do Cabeçalho HTTP

Considere a requisição:

```
GET /protected/index5.html HTTP/1.1
Host: 172.21.211.46
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; pt-BR; rv:1.9.2.14)
Gecko/20110218 Firefox/3.6.14 ( .NET CLR 3.5.30729)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pt-br,pt;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
If-Modified-Since: Thu, 31 Mar 2011 13:23:16 GMT
If-None-Match: "248c-53-49fc7322da022"
Authorization: Basic cmVkZXMtdW5pZmVpLTIwMTE6MTIzNDU=
```

Responda:

18. Do ponto de vista de ambos cliente e servidor, qual a utilidade do campo User-Agent?

19. Para que serve o parâmetro “q” entre as informações do cabeçalho?

20. O que significa uma requisição com campo “Connection: keep-alive”? E “Connection: close”?

Este link irá auxiliá-lo nas respostas:
http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html