Partie 0

Faire demo sous ubuntu VMware

Monter la page web dans un navigateur

Partie 1

Nous sommes sur un **protocole applicatif basé sur TCP** et, dans ce cas là, il est généralement **possible d'utiliser Telnet** (si le serveur le permet) pour étudier un peu les commandes…

La commande GET

Nous ouvrons une session telnet sur le serveur HTTP de 192.168.102.76 (Un serveur web écoute par convention sur le port 80).

```
Telnet 192.168.102.76 80

Trying 192.168.0.253...
Connected to 192.168.102.76.
Escape character is '^]'.
```

La session est ouverte. Utilisons maintenant la commande GET…

```
GET / HTTP/1.0
```

Nous la faisons en **protocole 1.0**, c'est un peu plus simple. Deux « return » pour valider… Et voici qu'arrive la réponse du serveur. Cette partie constitue l'**en-tête**, systématiquement **envoyée** par le **serveur** :

```
HTTP/1.1 200 OK
Date: Thu, 12 Feb 2009 22:06:56 GMT
Server: Apache/2.2.9 (Ubuntu)
Last-Modified: Thu, 12 Feb 2009 21:43:36 GMT
ETag: "2048-ec-462bf9cd01200"
Accept-Ranges: bytes
Content-Length: 236
Vary: Accept-Encoding
Connection: close
Content-Type: text/html
```

- Le serveur sait faire du HTTP 1.1. C'est un Apache version 2.2.9.
- Il indique également La date GMT de dernière modification du document demandé,
- qu'il mettra fin à la connexion TCP à la fin de l'envoi,
- et que le document fourni est au format MIME : text/html

Et voici le document proprement dit :

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv= "Content-Type" content= "text/html; charset=iso-
8859-1">
</head>

<body>
<p>Hello world.</p>
<p><img src="images/tux.jpeg" width= "97" height= "115"></p>
</body>
</html>
```

Partie 2

Bien ! Mais l'image ? Essayons de l'avoir…

```
Telnet 192.168.102.76 80
GET /images/tux.jpeg  HTTP/1.0

HTTP/1.1 200 OK
Date: Thu, 12 Feb 2009 22:15:52 GMT
Server: Apache/2.2.9 (Ubuntu)
Last-Modified: Thu, 12 Feb 2009 21:38:11 GMT
ETag: "2045-bdf-462bf8970f6c0"
Accept-Ranges: bytes
Content-Length: 3039
Connection: close
Content-Type: image/jpeg

▶JFIF☺☺☺☺C        ♠


▬

§▶▬ ↔"" ↔▼▼$(4,$&1'▼▼-=-157:::#+?D?8C49:7C☺
```

Là, il ne fallait tout de même pas s'attendre à des miracles, avec une console en mode texte. On a bien reçu l'image, **le type MIME (image/jpeg) est bien signalé**, mais il n'est pas possible de la visualiser avec telnet.

Partie 3

On espionne avec le sniffeur...

Comment travaille le navigateur ?

Il a fait exactement la même chose que nous :

- **Il appelle la page d'accueil GET / HTTP/1.0** (éventuellement HTTP/1.1, mais alors, suivant la définition de cette version HTTP, il devra au moins envoyer aussi le nom d'hôte du serveur interrogé).
- Une fois la page reçue, **il va chercher dedans tous les URI**, **ici** celui de **l'image**, et va les **appeler** avec un **GET**. Dans le cas de **l'image**, il devra la **recomposer**, ce qui lui est possible parce qu'il **connaît** le **format d'encodage** jpeg.
- Il **affiche** alors la **page**, dans son intégralité.

Première observation de la conversation

Charger le fichier capture_http.pcap

Nous appelons la page avec Internet Explorer et le sniffeur va enregistrer ce qu'il se passe. Les trames surlignées sont celles qui sont propres à HTTP. Mais n'oublions pas que HTTP s'appuie sur TCP, raison pour laquelle les autres trames existent :

- La trame 7 représente la première requête du client
- La trame 9 renvoie le document demandé, c'est à dire la page d'accueil du site.
- La trame 13 indique une requête supplémentaire pour l'image
- Les trames 18 représentent l'envoi par le serveur de l'image demandée.

Partie 4

La première requête HTTP (trame 7)

Développer la trame 7

Pour cette première analyse, je laisse volontairement la totalité de la trame, afin de bien montrer que HTTP est un protocole « application », qui s'appuie sur TCP/IP, lui-même s'appuyant sur Ethernet dans cet exemple. Avec une connexion PPPoE, on aurait une couche supplémentaire introduite par PPP

A adapter

```
Hypertext Transfer Protocol
    GET / HTTP/1.1\r\n                          La version du protocole HTTP
utilisé.
Suivent les informations supplémentaires qu'envoie le client au serveur...
    Accept: image/gif,                          Les images gif...
            image/x-xbitmap,                    Les images bitmap (bmp par exemple
)
            image/jpeg,                         Les images jpeg...
            image/pjpeg,
            application/vnd.ms-powerpoint,      Les trois lignes qui suivent
            application/vnd.ms-excel,           Représentent des informations dont
l'intérêt
            application/msword,                 peut paraitre contestable...
            */*\r\n
    Accept-Language: fr\r\n                     Nous parlons français...
    Accept-Encoding: gzip, deflate\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)\r\n
    Host: linux.maison.mrs\r\n                  Celle-ci est indispensable au
protocole v1.1
    Connection: Keep-Alive\r\n                  Notez qu'IE demande à garder la
connexion
    \r\n
```

Note : Internet Explorer informe qu'il accepte ce type de documents. Dans la pratique, tous les navigateurs les acceptent, mais vous proposeront seulement de les enregistrer en tant que fichiers. Ici, IE indique qu'il est capable de les afficher lui-même.

Partie 5

Développer la trame 9

La page d'accueil arrive

```
0000  3c 68 74 6d 6c 3e 0a 3c  68 65 61 64 3e 0a 3c 74   <html>.< head>.<t
0010  69 74 6c 65 3e 55 6e 74  69 74 6c 65 64 20 44 6f   itle>Unt itled Do
0020  63 75 6d 65 6e 74 3c 2f  74 69 74 6c 65 3e 0a 3c   cument</ title>.<
0030  6d 65 74 61 20 68 74 74  70 2d 65 71 75 69 76 3d   meta htt p-equiv=
0040  20 22 43 6f 6e 74 65 6e  74 2d 54 79 70 65 22 20    "Conten t-Type"
0050  63 6f 6e 74 65 6e 74 3d  20 22 74 65 78 74 2f 68   content=  "text/h
0060  74 6d 6c 3b 20 63 68 61  72 73 65 74 3d 69 73 6f   tml; cha rset=iso
0070  2d 38 38 35 39 2d 31 22  3e 0a 3c 2f 68 65 61 64   -8859-1" >.</head
0080  3e 0a 0a 3c 62 6f 64 79  3e 0a 3c 70 3e 48 65 6c   >..<body >.<p>Hel
0090  6c 6f 20 77 6f 72 6c 64  2e 3c 2f 70 3e 0a 3c 70   lo world .</p>.<p
00a0  3e 3c 69 6d 67 20 73 72  63 3d 22 69 6d 61 67 65   ><img sr c="image
00b0  73 2f 74 75 78 2e 6a 70  65 67 22 20 77 69 64 74   s/tux.jp eg" widt
00c0  68 3d 20 22 39 37 22 20  68 65 69 67 68 74 3d 20   h= "97"  height=
00d0  22 31 31 35 22 3e 3c 2f  70 3e 0a 3c 2f 62 6f 64   "115"></ p>.</bod
00e0  79 3e 0a 3c 2f 68 74 6d  6c 3e 0a 0a               y>.</htm l>..
```

C'est au **navigateur** de se débrouiller pour aller **chercher** les **données de cette image**, à partir des références fournies. Tous ceux qui pratiquent le HTML le savent bien…

Ceci **justifie** la présence de la requête de **la trame 13** :

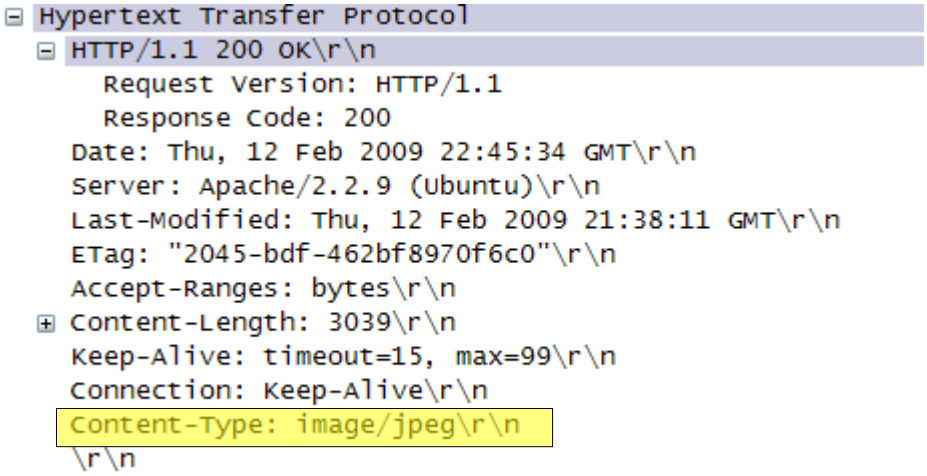
```
Hypertext Transfer Protocol
  GET /images/tux.jpeg HTTP/1.1\r\n
    Request Method: GET
    Request URI: /images/tux.jpeg
    Request Version: HTTP/1.1
  Host: 192.168.102.76\r\n
  User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.9.0.6) Gecko/2009011913 Firefox/3.0.6\r\n
  Accept: image/png,image/*;q=0.8,*/*;q=0.5\r\n
  Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3\r\n
  Accept-Encoding: gzip,deflate\r\n
  Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
  Keep-Alive: 300\r\n
  Connection: keep-alive\r\n
  Referer: http://192.168.102.76/\r\n
  \r\n
```

Appel d'une référence relative : /images/tux.jpeg

Depuis la page indiquée, ce qui aboutit
à la référence absolue: http://192.168.102.76/images/tux.jpeg

Partie 6

Développer la trame 18



Le serveur envoie les données à partir de la trame 18 :

Partie 7

Le coup du cache...

Charger le fichier capture_http_cache.pcap

Puisque nous y sommes, profitons en pour observer un comportement intéressant du navigateur : La mise en cache des pages consultées.

L'internaute ferme son navigateur. Quelques instants plus tard, il l'ouvre à nouveau et réclame la même page. Que va-t-il se passer ?

La réponse n'est pas la même que dans le cas précédent. Voyons ceci de plus près…
Développer la trame 5

```
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
    Request Method: GET
    Request URI: /
    Request Version: HTTP/1.1
  Host: 192.168.102.76\r\n
  User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.9.0.6) Gecko/2009011913 Firefox/3.0.6\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
  Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3\r\n
  Accept-Encoding: gzip,deflate\r\n
  Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
  Keep-Alive: 300\r\n
  Connection: keep-alive\r\n
  If-Modified-Since: Thu, 12 Feb 2009 21:43:36 GMT\r\n
  If-None-Match: "2048-ec-462bf9cd01200"-gzip\r\n
  Cache-Control: max-age=0\r\n
  \r\n
```

Le navigateur demande au serveur la page, si elle a été modifiée depuis la date de son précédent chargement, tout simplement parce qu'il a conservé en cache cette page que nous avons déjà demandée il n'y a pas si longtemps.

La réponse

Développer la trame 10

```
Hypertext Transfer Protocol
  HTTP/1.1 304 Not Modified\r\n
    Request Version: HTTP/1.1
    Response Code: 304
  Date: Fri, 13 Feb 2009 01:35:37 GMT\r\n
  Server: Apache/2.2.9 (Ubuntu)\r\n
  Connection: Keep-Alive\r\n
  Keep-Alive: timeout=15, max=99\r\n
  ETag: "2045-bdf-462bf8970f6c0"\r\n
  \r\n
```

Le serveur s'est contenté de répondre que la page n'a pas été modifiée…

Le navigateur va donc réafficher la page qu'il a conservée en cache. Cette méthode de travail présente deux particularités:

- Le temps d'affichage est considérablement raccourci lorsque l'on navigue dans un site, puisque les pages déjà chargées ne le sont généralement plus si l'on revient dessus.
- L'espace requis pour le cache gonfle considérablement et peut occuper jusqu'à plusieurs dizaines de Mo sur votre disque…