PHP運作原理(1)

- PHP：
  - **Server** 端的描述語言，主要的目的是用來製作動態網頁。
  - 開放原始碼（**Open Source**）而且是跨平台的伺服端描述語言。
  - 建立成一個**Apache** 模組時，**PHP** 能夠快速執行地，不需像**CGI** 需要產生新的**process** 執行。
  - 可以用來傳送**HTTP** 表頭，也可以設定**Cookies**，做授權管理，以及將使用者重新導向至新的頁面。

PHP運作原理(1)

例：Server 端的 index.php 檔案內容如下：

```
<HTML>
第一個由PHP 輸出的程式 <BR>
<?php
   echo “Hello PHP”;
?>
</HTML>
```

PHP運作原理(2)

Client 端Browser 連線到Server 端的 Apache 動作如下：

1. open index.php

2.輸出 HTTP 回應

3. for(; 檔案未結束;){

    if(檢查到 <?php){

        for(;是否檢查到 ?> 表示php程式段落結束離開迴圈;){

            呼叫 php parser, 將程式輸出結果, 輸出到用戶端 }

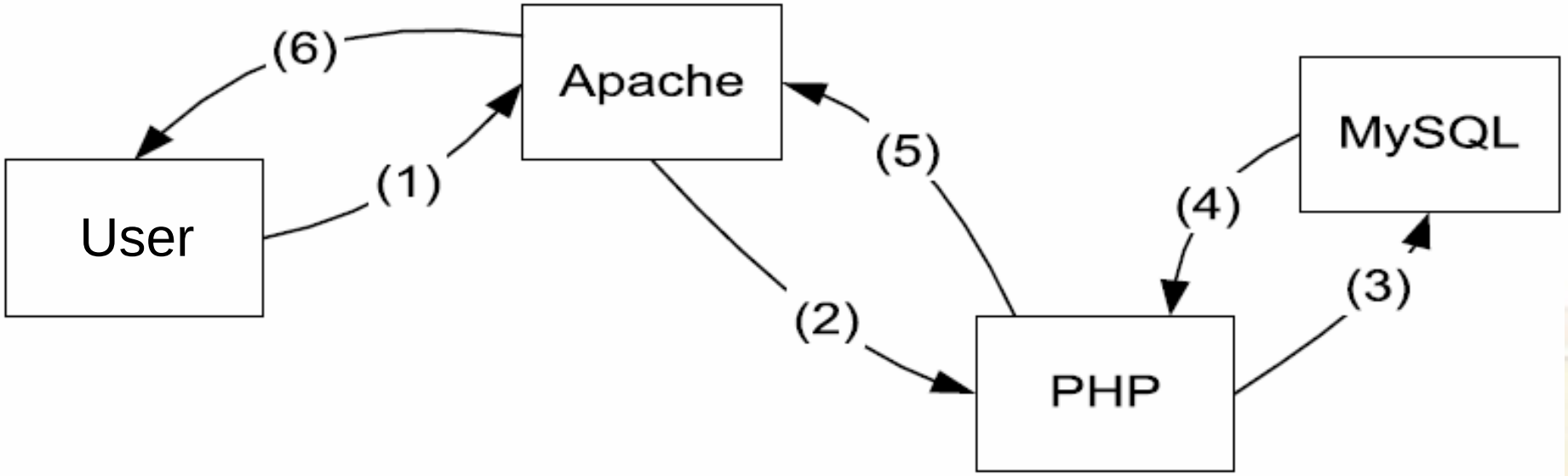
    }

    將檔案內容HTML等輸出到用戶端

}

4.關閉檔案

PHP、Apache、MySQL合作原理(1)



PHP、Apache、MySQL合作原理(1)

- 使用者透過**PHP** 存取**MySQL**資料庫內容的流程:
  - 1.使用者向**Apache**要求**PHP**相關網頁**(.php)**。
  - **2.**當**Apache**收到這個要求，經判斷，把此要求轉交給**PHP**直譯器負責。
  - **3. PHP**直譯器解譯此網頁的**PHP**語法，當遇到**My SQL**相關的函式，則把**Request**間接傳遞給**My SQL**。
  - **4. My SQL**把所要求的結果反傳回**PHP** 。
  - **5. PHP**直譯器把此結果包裝成**Apache**看得懂的**Html**語法，後傳回給**Apache**。
  - **6. Apache**把此結果**(**已被**PHP**處理成**Html)**傳回給使用者的**browser**。

PHP、Apache、MySQL合作原理(2)

- **PHP主要透過四個主函式來達成和MySQL溝通:**
  - mysql_connect – 開啟一和MySQL Server的連結點。

```
$link = mysql_connect("localhost", "mysql_user",
   "mysql_password")
```

  - mysql_query – 送給MySQL的要求。

```
$result = mysql_query("SELECT * WHERE 1=1")
```

  - mysql_fetch_row – 得到透過mysql_query的結果。

```
while ($row = mysql_fetch_row ($result))
   { …
     ..
   }
```

  - mysql_close – 整個作業完成，關閉連線。

```
mysql_close($link);
```

PHP語法(1)

簡單的 PHP 程式

```
1   <?
2       $a = 4;
3       $b = “abcd”;
4       echo ($b[0].$a.$b[1]);
5   // this is comment.
6       function Loop($b) {
7               global $a;
8               $I = 1;
9
10              while ( $I <= $a) {
11                      if ( $I == 4) {
12                              echo (“end\n”);
13                      }else {
14                              echo (“not end\n”);
15                      }
16                      $I++;
17              }
18      }
19
20      ?>
```

PHP語法(2)

- 第1-3行：變數宣告:$NAME = VALUE;
- NAME爲任何非數字開頭的字串，再加上$字號而成變數，而VALUE則可爲任意的數值或是字串等，並不受限制，習慣上字串會用引號括起來。
- 第4行：echo是PHP內定的函式，用來把函式後括號內的式子輸出。
- 字串其實是Array的變形，所以在第3行中，b[1]直接對應b字串的第二個單字，因爲PHP的array和C是一樣的，索引都從0開始。除了數字可作加法外，PHP的字串也可用來相加(Concatenation)，只需在變數中間用. 標示出來即可。
- 第5行：php裡的注解可用C(/*...*/)和C++(//)兩種格式。

PHP語法(2)

- 第6行：使用者可自行定義自已的函式，形式如:
- function ([*variables*]) {.....
- }
- Function裡定義的變數皆爲「區域變數」，Function外的變數則爲「全域變數」若要使區域變數爲引用全域變數，只需在變數定義前加上「Global」。
- 第10-17行：迴圈控制: for(expr1, expr2, expr3)
- statement
- while(expression)
- statement
- 所以可以把此While改寫成:for ($I = 1; $I <= $a; $I++) { ..... }
- do ...while也是迴圈控制的形式，不過和While最大的分別是，do...while一定會先執行一次，不論條件成不成立。

PHP語法(3)

- 第11-15行:條件判斷:if (expression)
- statement
- elseif(expression)
- statement
- else
- statement
- 也可以用switch ...case也改寫，如下:
- switch($I) { case “4”:....
- Default: ....
- 其中，Case是假設有多種條件式，當沒有吻和條件時，就執行Default這個預設式。

PHP程式設計主要概念

- **HTML** 主要是用**GET** 和**POST** 兩種方法傳遞**Browser** 到HTTP Server 的變數，PHP 就是利用這個來收取 Client 傳過來的參數。
- GET變數:當**Web server**收到一組利用「**GET**」的方法來傳遞變數的請求時，則**php**會產生一組告爲**$_GET**的全域關連性陣列**(global hash)**來儲存這些變數的資料。
  - 把網頁上的**<Form>……</Form>**包進**HTTP**的**Request Header** 裡，而成網址**(URL)**的一部份。
- POST變數:用**post**當變數傳遞的方式，也是產生一個全域關連性陣列，取名爲**$_POST**。
  - 把**<FORM>…</FORM>**裡面的變數包進此**Request** 裡的內容中，而不是像**GET**的做法是把變數加到**Header** 裡，這個作法的好處是所要傳遞的資料可以很大，而不會因**HTTP**的**Header** 的大小而限制。

Cookie 簡介

- **Cookie:**
  儲存在Browser端的名稱變數，用來記錄所屬Browser和Server端的連線狀態。
  - 包裝在**HTTP**的「**Response Header**」，透過**HTTP Response**中的**Set-Cookie Header** 欄位來達成。
  - **Cookie**皆有其存活的時間，**Browser**包裝此**Cookie**在一系列的**Request**裡，直到其到期爲止。
- **Cookie Manager:**
  - 「**Domain**」參數限制了此**cookie**所能被傳遞的站台**(Site)**，假如此參數沒有設定，那此**Cookie**只能在設定此**cookie**的站台**(Site)**運作。
  - 若**Browser**所要求的網頁沒有在「**Path**」參數所允許的路徑或其子路徑下，則**Cookie**不會被包裝在其**Request**封包裡。這在限制一個**Web**站台下的某些目錄才能執行**Cookie**，很有用。
  - 「**Secure**」的欄位，使其**browser**在傳送此**Cookie**時，利用**SSL(SecureSockets Layer Protocol)**來做一個安全化的連線。

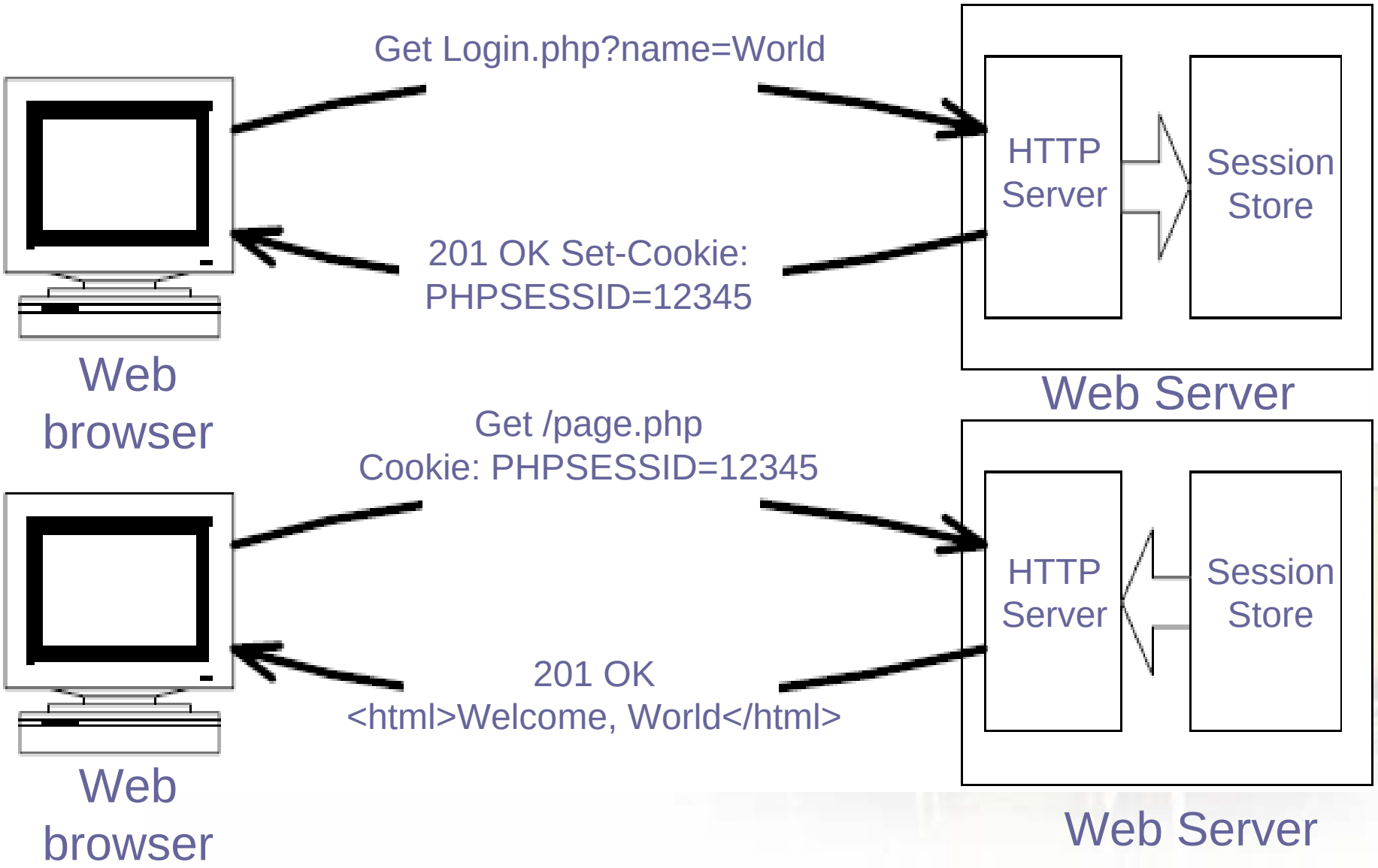
Session 簡介(1)

- Session:利用對應到每一個使用者的Session變數來識別和管理的方法。
  - 當Session起始時，此Client會有一個Session識別碼，用來加進接下來一連串的Request 裡，通常這個識別碼是個Cookie。
  - 相較於把所有需要去維護的變數及其值加入每一個請求（Request），Browser只儲存了單一個Session識別碼，用來尋找和初始在Web Server上面存放的變數。
- Session管理上的特性：
  - 資訊或狀態需被儲存，例如，在多個請求（Requests）下，信用卡的卡號需被小心維護。
  - 每一個HTTP請求（Request）必須含有一個識別碼，使主機能用來處理此請求（Request）的內容。
  - Session皆需有一個Timeout值，否則，當使用者一離開此網站，就沒有方法能去終結此Session。

Session 簡介(2)

- **Session** 運作方式:在使用Session機制的web主機上，當使用者對開啓Session的網頁作第一次的請求（Request）時，PHP產生一個Session ID及創造一個用來存放Session相關變數的檔案，PHP設定一個存有Session ID的Cookie，回應(Response)給Browser。這個Browser把此Cookie變數記錄下來，且把此Cookie加在接下來一連串的Request 裡。

Session 運作方式



PHP 的Session 管理

- **PHP Session管理:利用disk-base爲主的檔案來存放變數，另一常見的方式是利用資料庫的方式。**
  - **1.開啓Session：第一次PHP script呼叫session_start()，一個session識別碼被產生，且一個Set-Cookie Header 欄位被加入回應(response)封包裡。**
    - **•建立一個名叫PHPSESSID的session cookie，且其值就是這個session的識別碼是一32個十六進位數字所組成的字串。**
    - **•PHP的session管理機制自動把此cookie加入回應(response)裡頭，而不用呼叫setcookie()或header()兩函式。**

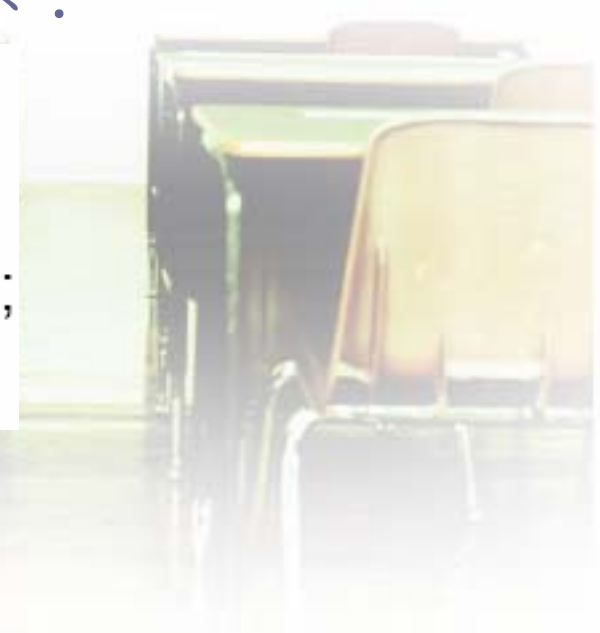
PHP 的Session 管理

- 2.使用Session變數：被session使用的變數需使用session_register()函式來註冊(register)。用session_unregister()函式來移除。
- •Session變數可以是Boolean、整數(integer)、double、字串(string)、物件或陣列等php允許的變數形態。
- 3.結束Session：在某些時候，假如使用者登出時，script能呼叫session_destroy()函式把所屬的session從系統消除掉，但沒有移除在browser裡的PHPSESSID cookie，直到此browser被關閉。

Cookie及Session實例解析(1)

- Session及Cookie都是爲了追蹤使用者所運育而生的機制，兩者最主要的差異是:Cookie是把記錄儲存在Client端，而Session卻把記錄反置於Server端。
- cookie：
  - 若有一**php檔(cookie.php)**，內容如下：

```
<?php
$thisCookie++;
setcookie("thisCookie", $thisCookie);
?>
```

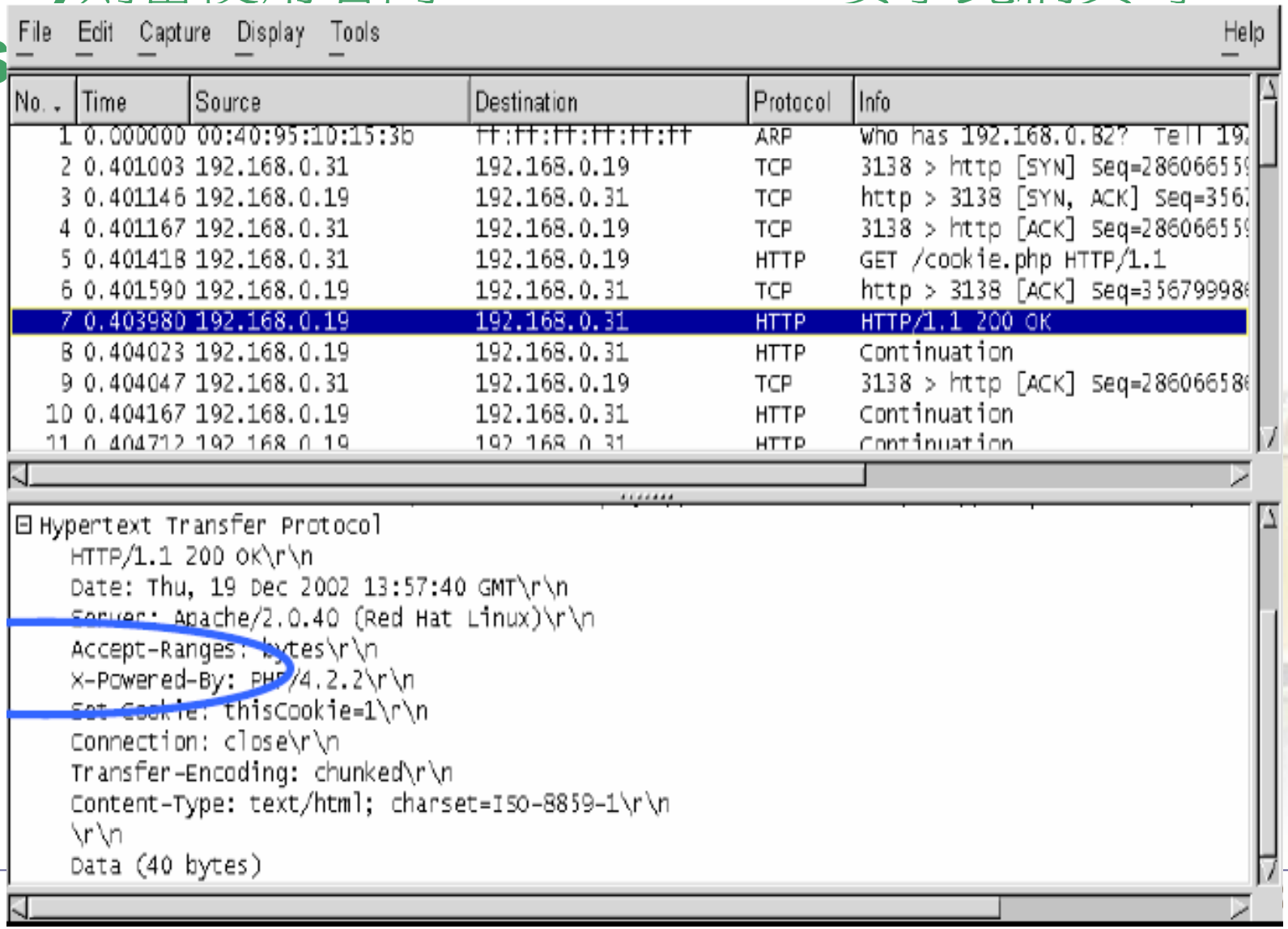


Cookie及Session實例解析(1)

- 在使用者第一次要求此網頁時，Server在回應(Response)結果的過程中，會在HTTP的回應Header (Response header)裡，多一個Set-Cookie的欄位，然後把Cookie填入。
- 而當PHP在執行的過程中，若有動到此Cookie變數，則Web Server在回應(Response)的過程中，又利用Set-Cookie的欄位重設此Cookie的值。

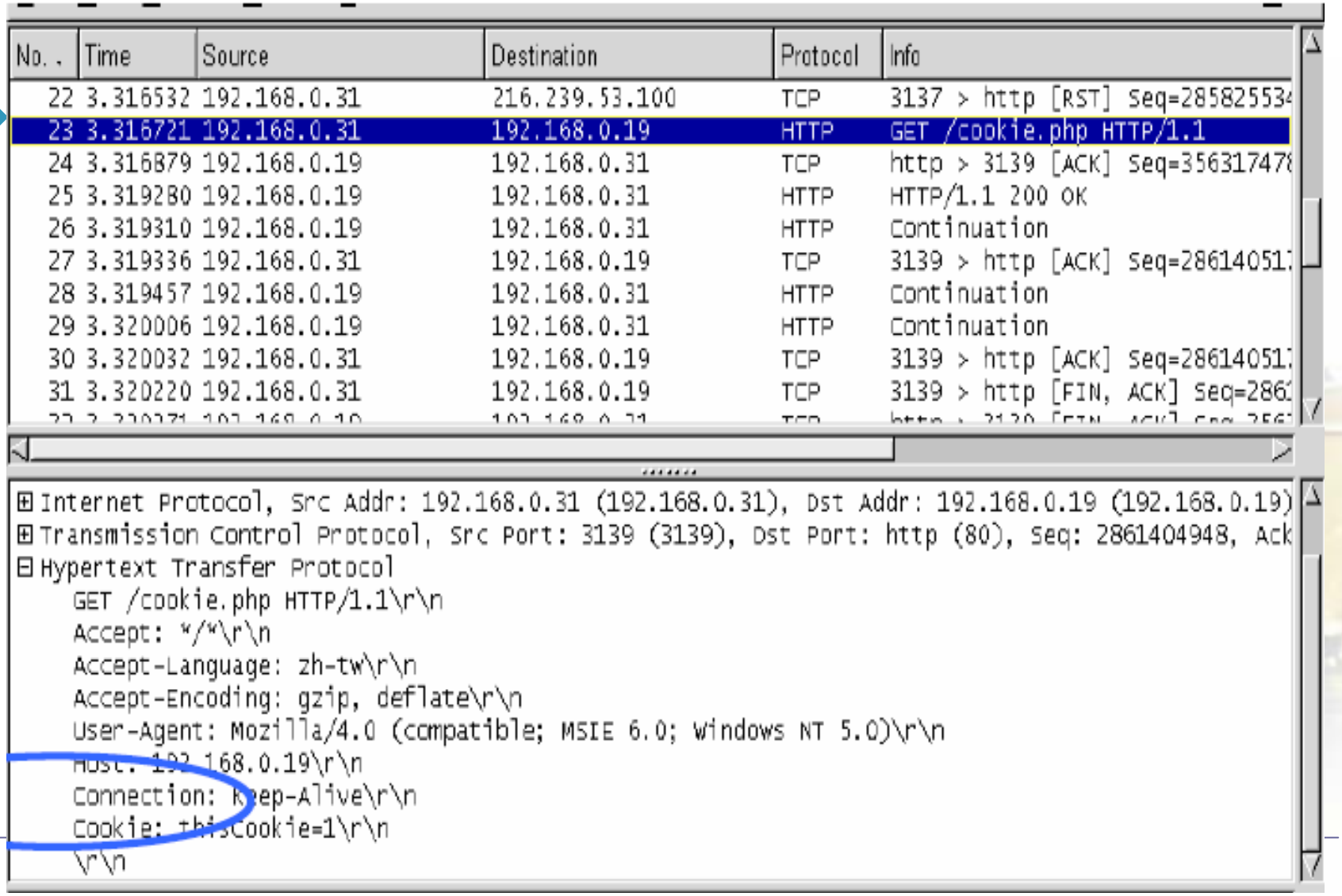
Cookie及Session實例解析(2)

- (1)則當使用者向Web Server要求此網頁時
- S



Cookie及Session實例解析(3)

- (2)使用者重覆向Web Server Request此網



No. .	Time	Source	Destination	Protocol	Info
22	3.316532	192.168.0.31	216.239.53.100	TCP	3137 > http [RST] Seq=28582553
23	3.316721	192.168.0.31	192.168.0.19	HTTP	GET /cookie.php HTTP/1.1
24	3.316879	192.168.0.19	192.168.0.31	TCP	http > 3139 [ACK] Seq=35631747
25	3.319280	192.168.0.19	192.168.0.31	HTTP	HTTP/1.1 200 OK
26	3.319310	192.168.0.19	192.168.0.31	HTTP	Continuation
27	3.319336	192.168.0.31	192.168.0.19	TCP	3139 > http [ACK] Seq=28614051
28	3.319457	192.168.0.19	192.168.0.31	HTTP	Continuation
29	3.320006	192.168.0.19	192.168.0.31	HTTP	Continuation
30	3.320032	192.168.0.31	192.168.0.19	TCP	3139 > http [ACK] Seq=28614051
31	3.320220	192.168.0.31	192.168.0.19	TCP	3139 > http [FIN, ACK] Seq=286

```
⊞ Internet Protocol, Src Addr: 192.168.0.31 (192.168.0.31), Dst Addr: 192.168.0.19 (192.168.0.19)
⊞ Transmission Control Protocol, Src Port: 3139 (3139), Dst Port: http (80), Seq: 2861404948, Ack
⊟ Hypertext Transfer Protocol
    GET /cookie.php HTTP/1.1\r\n
    Accept: */*\r\n
    Accept-Language: zh-tw\r\n
    Accept-Encoding: gzip, deflate\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)\r\n
    Host: 192.168.0.19\r\n
    Connection: Keep-Alive\r\n
    Cookie: thisCookie=1\r\n
    \r\n
```

Cookie及Session實例解析(4)

- Session：

若有一**PHP檔(Session1.php)**如下:

```
<?php

session_start();

$user = "nagano";

if ( session_register(user)) {

        echo("Set Session\n");

}else {

        echo("Failed to register\n");

}

?>
```

- Web Server在回應(Response)封包裡，多加了Set-Cookie的欄位，而Cookie的名稱爲PHPSESSID(在php.ini中可自定這個名稱)而其值就是PHP用來在/tmp目錄下所產生的Session檔。

Cookie及Session實例解析(5)

- (1)當使用者第一次向Web Server Request
- S

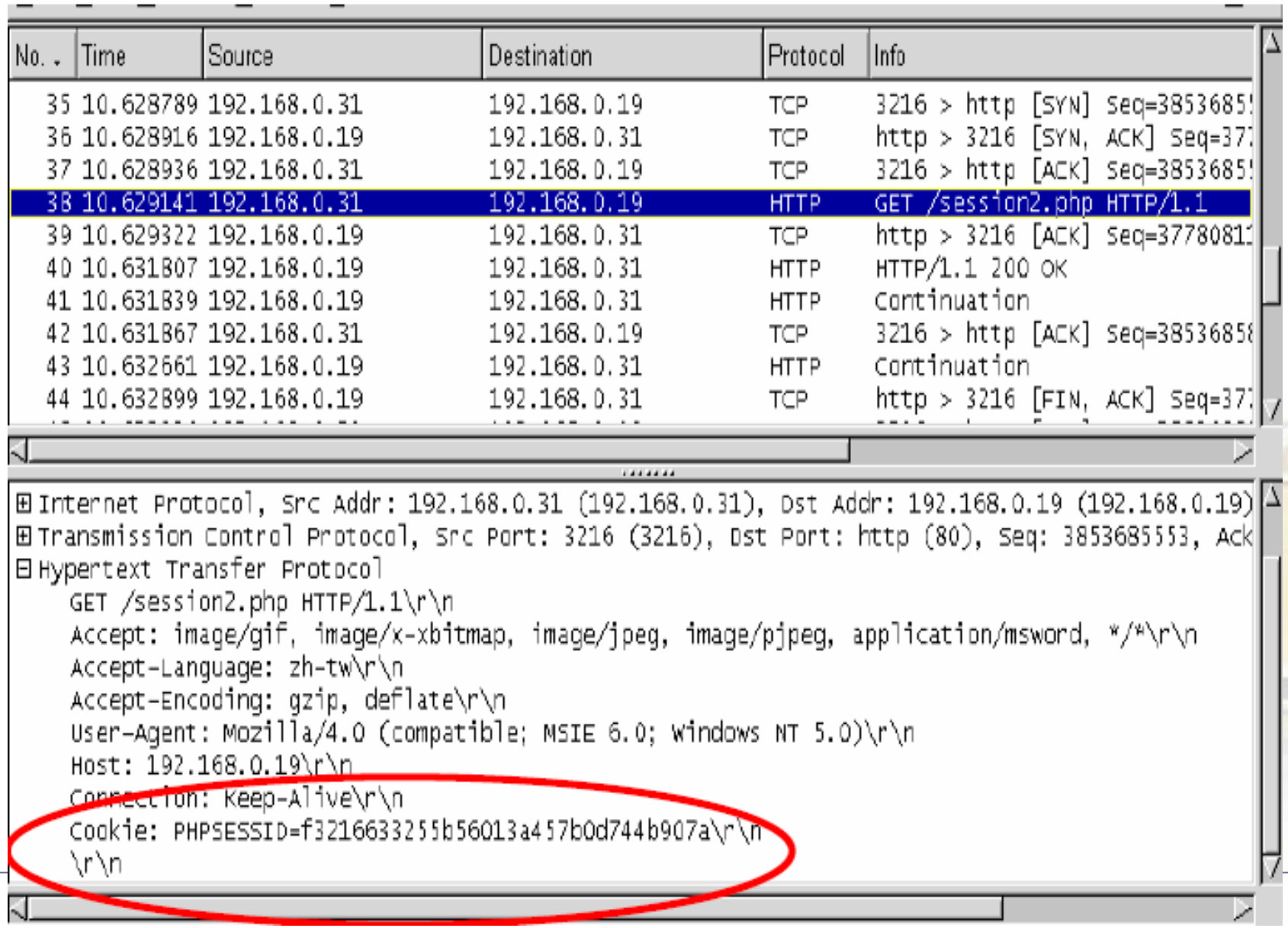
No. .	Time	Source	Destination	Protocol	Info
4	2.837021	192.168.0.19	192.168.0.31	TCP	http > 3215 [SYN, ACK] Seq=37
5	2.837053	192.168.0.31	192.168.0.19	TCP	3215 > http [ACK] Seq=3851843
6	2.837542	192.168.0.31	192.168.0.19	HTTP	GET /session1.php HTTP/1.1
7	2.837730	192.168.0.19	192.168.0.31	TCP	http > 3215 [ACK] Seq=3775572
8	2.843865	192.168.0.19	192.168.0.31	HTTP	HTTP/1.1 200 OK
9	2.844898	192.168.0.19	192.168.0.31	HTTP	Continuation
10	2.844931	192.168.0.31	192.168.0.19	TCP	3215 > http [ACK] Seq=3851844
11	2.845090	192.168.0.31	192.168.0.19	TCP	3215 > http [FIN, ACK] Seq=385
12	2.845528	192.168.0.19	192.168.0.31	TCP	http > 3215 [FIN, ACK] Seq=37
13	2.845567	192.168.0.31	192.168.0.19	TCP	3215 > http [ACK] Seq=3851844

```
⊞ Frame 8 (507 on wire, 507 captured)
⊞ Ethernet II
⊞ Internet Protocol, Src Addr: 192.168.0.19 (192.168.0.19), Dst Addr: 192.168.0.31 (192.168.0.31)
⊞ Transmission Control Protocol, Src Port: http (80), Dst Port: 3215 (3215), Seq: 3775572132, Ack
⊟ Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
    Date: Thu, 19 Dec 2002 15:08:49 GMT\r\n
    Server: Apache/2.0.40 (Red Hat Linux)\r\n
    Accept-Ranges: bytes\r\n
    X-Powered-By: PHP/4.2.2\r\n
    Set-Cookie: PHPSESSID=f3216633255b56013a457b0d744b907a; path=/\r\n
    Expires: Thu, 19 Nov 1981 08:52:00 GMT\r\n
```

Cookie及Session實例解析(6)

- (2)在接下來的進一步Request ，Bowser會自動把此Cookie加在請求Header (Request Header)裡。
  - 設定**Cookie** 後，之後的傳送自動將**Cookie** 加入，此**Cookie**就是用來取得位在**Server**端的**Session**檔

Cookie及Session實例解析(6)



No.	Time	Source	Destination	Protocol	Info
35	10.628789	192.168.0.31	192.168.0.19	TCP	3216 > http [SYN] Seq=3853685
36	10.628916	192.168.0.19	192.168.0.31	TCP	http > 3216 [SYN, ACK] Seq=37
37	10.628936	192.168.0.31	192.168.0.19	TCP	3216 > http [ACK] Seq=3853685
38	10.629141	192.168.0.31	192.168.0.19	HTTP	GET /session2.php HTTP/1.1
39	10.629322	192.168.0.19	192.168.0.31	TCP	http > 3216 [ACK] Seq=3778081
40	10.631807	192.168.0.19	192.168.0.31	HTTP	HTTP/1.1 200 OK
41	10.631839	192.168.0.19	192.168.0.31	HTTP	Continuation
42	10.631867	192.168.0.31	192.168.0.19	TCP	3216 > http [ACK] Seq=3853685
43	10.632661	192.168.0.19	192.168.0.31	HTTP	Continuation
44	10.632899	192.168.0.19	192.168.0.31	TCP	http > 3216 [FIN, ACK] Seq=37

```
⊞ Internet Protocol, Src Addr: 192.168.0.31 (192.168.0.31), Dst Addr: 192.168.0.19 (192.168.0.19)
⊞ Transmission Control Protocol, Src Port: 3216 (3216), Dst Port: http (80), Seq: 3853685553, Ack
⊟ Hypertext Transfer Protocol
    GET /session2.php HTTP/1.1\r\n
    Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/msword, */*\r\n
    Accept-Language: zh-tw\r\n
    Accept-Encoding: gzip, deflate\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)\r\n
    Host: 192.168.0.19\r\n
    Connection: Keep-Alive\r\n
    Cookie: PHPSESSID=f3216633255b56013a457b0d744b907a\r\n
    \r\n
```