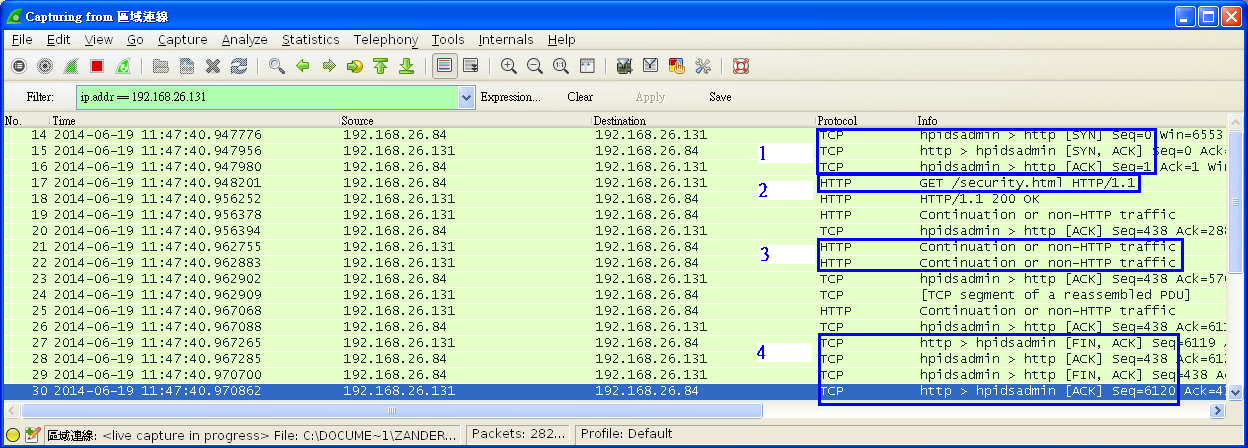
**Web Service Zander 2014/06/19**

**目的：**

本文件主要針對Web Service，資源以及網頁開啟流程做一介紹。旨在即便後續抽換了TCP/IP Stack，仍能理解目前Web Server架構，以便評估是否將來面臨Web Server需要加強、或調整，甚至替換的決策。

**HTTP GET封包Sequence：**

以一般 Web Get Request為例。 當使用者從瀏覽器輸入(或點)一網址URL，此時會對Server開啟一HTTP連線請求，連線成功送出HTTP Request。



Source欄位192.168.26.84為PC，192.168.26.131為RMCARD。

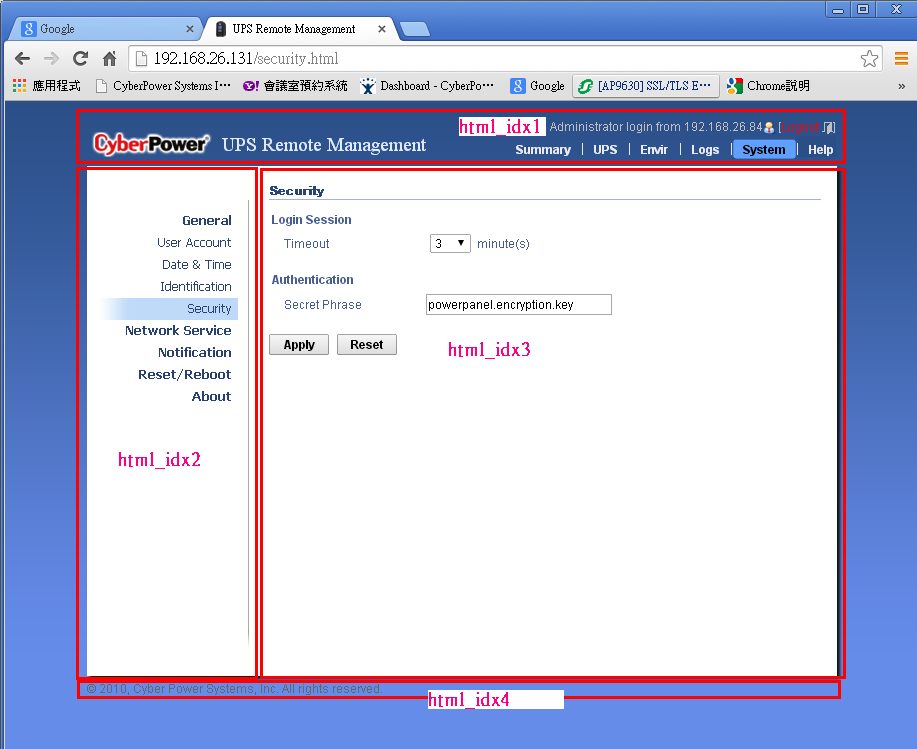
藍色編號及意義：

1：TCP三方交握，Info欄位有標示TCP旗標，標準程序：SYN => SYN/ACK => ACK。

***2：PC對RMCARD發起HTTP GET Request，GET /security.html 網頁。***

3：RMCARD(HV2.0某個版本開始)支援Sliding Window(加強HTTP效能)。

4：TCP通訊關閉流程，FIN/ACK=>ACK=>FIN/ACK=>ACK

**網頁資料結構：**

*struct fsdata\_file {*

*const struct fsdata\_file \*next;*

*const char \*name;*

*const HTML\_IDX html\_idx1;*

*const HTML\_IDX html\_idx2;*

*const HTML\_IDX html\_idx3;*

*const HTML\_IDX html\_idx4;*

*const BYTE page\_access\_identity;*

*const HTML\_IDX count;*

*};*

|  |  |
| --- | --- |
| 結構成員 |  |
| const struct fsdata\_file \*next | 指向相同結構的指標，名為next，可理解該結構為一單向Link List |
| const char \*name | 一指向固定字串的指標，為URL網址的名稱 (不含參數) |
| const HTML\_IDX html\_idx[1:4] | Display handle id |
| const BYTE page\_access\_identity | 建構的網頁存取權限。(admin/outlet user/viewer，屬於UPS/ATS/UPS+ATS…) |
| const HTML\_IDX count; | Action handle id |

以右圖為例，其結構內容為：

*const struct fsdata\_file file\_security\_html[] = {{*

*file\_security\_cgi, //下一個網頁結構為 security\_cgi*

*security\_html, //本結構的 unique name 為 security\_html (“/security.html”)*

*dx\_header\_SYS\_security, //header id*

*idx\_SYS\_security, //left menu id*

*idx\_security\_html, //body id*

*idx\_footer, //footer id*

*FsFile\_UandA|FsFile\_AUser, //access-able for UPS and ATS, Admin User Only*

*idx\_security\_html //action handle id*

*}};*

參考 fsdata.c，串列鏈結：

(上面還有一大堆一大堆一大堆網頁)

…

*//[==================== LOGIN ====================*

*const struct fsdata\_file file\_login\_counter\_html[] = {{*

*file\_ups\_summary\_html, login\_counter\_html,*

*idx\_login\_counter,*

*idx\_null,*

*idx\_null,*

*idx\_null,*

*FsFile\_UandA|FsFile\_AODUser,*

*idx\_login\_html}};*

*const struct fsdata\_file file\_login\_html[] = {{*

*file\_login\_counter\_html, login\_html,*

*idx\_set\_cookies,*

*idx\_login\_header,*

*idx\_login\_html,*

*idx\_tiny\_footer,*

*FsFile\_UandA|FsFile\_AODUser,*

*idx\_login\_html}};*

*//==================== LOGIN ====================]*

*#define FS\_ROOT file\_login\_html*

*//每個GET Request都從 FS\_ROOT 開始 Search，直到有名稱Match (or 找不到FAIL=>開error網頁)。*

**GET Request 流程：**

1) 收到HTTP(S)封包，GET /security.html …. 內容。

2) “/security.html”逐一比對網頁鏈結的\*name，比對成功將要回覆的網頁內容指標指向該結構。

3) httpd\_cgi\_check 檢驗 action handle id。(\*.html 大多認為是單純為網頁開啟的請求，若為cgi則通常有action)

4) 依序傳回 html\_idx1, 2… 4。

\*網頁資料是存在外部Flash中，參考 fs.c Eng\_fs\_sub\_frame 結構。該結構定義了網頁html\_idx存在Flash中的起始位置，以及長度。**特別注意HTML\_IDX定義的id順序必須與fs\_sub\_frame順序一致(Mapping)。**

5) fs\_loadfixfile負責塞送出的封包

填入各Frame Page適當的”@”變動內容。例如RMCARD顯示UPS Model Name。(Runtime才能確定顯示的參數或狀態… )

**套用至HTTP/HTTPS**

基本的Web網頁資料結構及流程如上，切割乾淨之後，便是上層使用API的部分。若已明確定義收到的內容如何指向結構，剩下的只是開啟的動作如何達成。以下粗略跑一下HTTP/HTTPS的流程。

**Step 1. 收到一完整的HTTP URL Request。**

HTTP收的介面：*http\_recv*

HTTPS收的介面：*ssl\_read*

**Step 2. 收完統一透過以下介面解析，並handle設定/動作(if valid)：**

*/\* Fn@ parse\_http\_cmd*

*Arg@ char\* data : 收到的完整封包*

*u16\_t len : 封包長度*

*void \*arg : 指向name相符的網頁結構*

*Ret@ HTTP\_CMD\_PARSE\_RES :*

*返回parse的結果，可能為COMMAND GET/POST/UNKNOW/REJECT*

*\*/*

*HTTP\_CMD\_PARSE\_RES parse\_http\_cmd(char \*data, u16\_t len, void\* arg);*

*解析過程中，會呼叫以下介面設定/執行動作。*

*/\* Fn@ httpd\_cgi\_check*

*Arg@ const char \*str1 : HTTP URL*

*const HTML\_IDX idx : handle*

*Ret@ void*

*\*/*

*void httpd\_cgi\_check(const char \*str1, const HTML\_IDX idx)*

**Step 3. 取得網頁回傳時的必要資訊**

HTTP取得\*arg結構，\*arg結構的必要資訊記錄在該次transaction(http\_state \*hs)。以便記錄網頁回覆的細節。*HTTP比較麻煩，因為每個封包的收/送都必須自己Handle，故網頁傳到哪裡必須詳實記錄。*

HTTP透過以下介面完成收送流程：

*static err\_t send\_data(struct tcp\_pcb \*pcb, struct http\_state \*hs) //send handle (fs\_loadfixfile)*

*static err\_t http\_sent(void \*arg, struct tcp\_pcb \*pcb, u16\_t len) //ack handle*

*static err\_t http\_poll(void \*arg, struct tcp\_pcb \*pcb) //poll handle*

送完close tcp\_pcb。

HTTPS取得\*arg結構，透過以下介面回覆：

*void ssl\_DynPage(ssl\_context \*ssl, void \*arg) //將網頁\*arg結構回送給ssl peer (fs\_loadfixfile)*

送完close net\_conn client\_fd。

**實例：一個網頁的原始檔及開啟的完整內容，cgi handle以及顯示。**

以security.cgi為例吧。

該網頁的結構如下：

*const struct fsdata\_file file\_security\_cgi[] = {{*

*file\_tcpip6\_html,*

*security\_cgi, //”security.cgi”*

*idx\_header\_SYS\_security,*

*idx\_SYS\_security,*

*idx\_security\_html,*

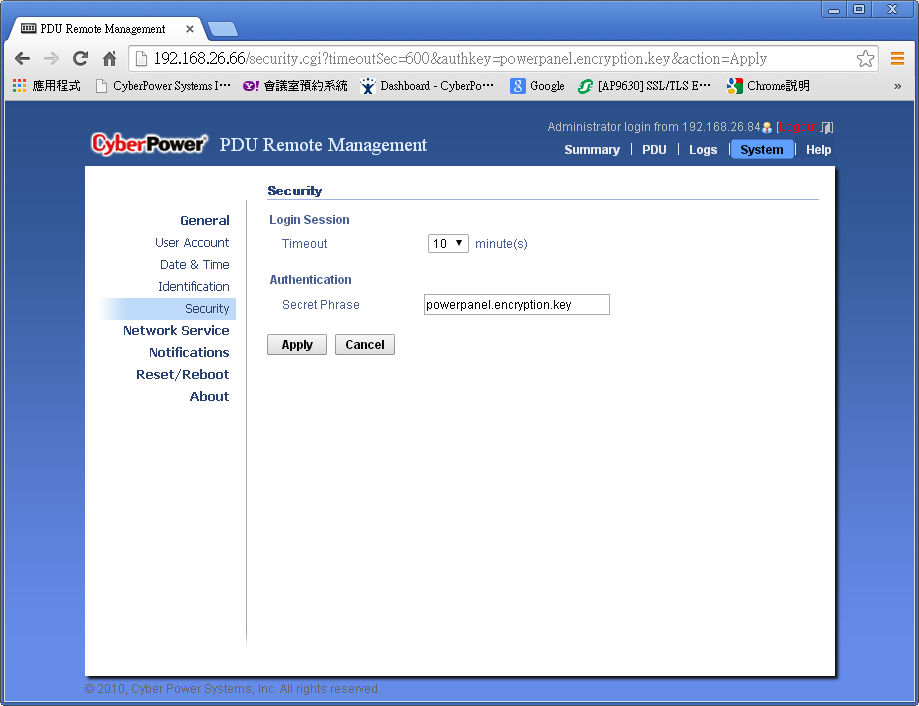
*idx\_footer,*

*FsFile\_UandA|FsFile\_AUser,*

*idx\_security\_cgi*

*}};*

使用者在security.html點下apply按鍵，則會送給RMCARD security.cgi網址(如下圖)。



結構宣告了該cgi對應的Action Handle Id為idx\_security\_cgi。

*void httpd\_cgi\_check(const char \*str1, const HTML\_IDX idx) {*

*switch(idx) {*

*…*

*case idx\_security\_cgi:*

*//http://192.168.10.202/security.cgi?timeoutSec=60&phrase=&action=Apply*

*paidx=nfind(str1,TIMEOUTSEC,sizeof(TIMEOUTSEC)); //TIMEOUTSEC: const char “timeoutSec”*

*if(paidx!=NFIND\_NEXIST\_CODE) {*

*input\_decode(eep\_data\_buf,str1,paidx);*

*strnum2val(eep\_data\_buf, 0, &temp2);*

*if(temp2==60) //case 1*

*eep\_data\_buf[0]=1;*

*…*

*else if(temp2==600)*

*eep\_data\_buf[0]=10;*

*else //giving default value of timeout~*

*eep\_data\_buf[0]=5;*

*session\_timeout=temp2; //update session\_timeout minutes*

*I2C\_EEPROM\_BufferWrite(NIT\_BASE, NIT\_SESSIONTIMEOUT, eep\_data\_buf,1); //write eep*

*}*

*paidx=nfind(str1, "authkey", 7);*

*if(paidx!=NFIND\_NEXIST\_CODE) {*

*input\_decode(eep\_data\_buf,str1,paidx);*

*I2C\_EEPROM\_BufferWrite(SECURITY\_BASE, AUTH\_KEY, eep\_data\_buf,PPBE\_KEY\_MAXLEN);*

*}*

*break;*

*…*

*}*

開啟網頁，依序回覆Header、Menu、Body and Footer Frame：

*idx\_header\_SYS\_security*

*idx\_SYS\_security*

*idx\_security\_html*

*idx\_footer*

idx\_header\_SYS\_security：Header部分，Security網頁屬於System Header Menu。

Idx\_SYS\_security：Menu部分，Security網頁屬於System Menu，並且security被選擇。

Idx\_security\_html：Body部分，顯示Security網頁內容。

Idx\_footer：Footer部分，一般登入後的頁面都共用同一個footer。

舉例回Body網頁時，idx\_security\_html的Index對到fs\_sub\_frame的表，可找到：

*security\_Add, BufferSize\_security, //idx\_security\_html,*

據此初始flash要去讀security\_Add的位址，總長度為BufferSize\_security。讀出來之後丟給fs\_loadfixfile送出適當的內容，並返回資訊以記錄下次flash讀取的位置。

Security.html原始檔：

*<div class="header">\084</div>*

*<div class="content">*

*<form action="security.cgi" method="get">*

*<div class="caption">\096</div>*

*<div class="gap2 securityLb">*

*<span class="lb">\097</span>*

*<select name="timeoutSec" class="autoWidth">*

*<option value="60" @10>1</option>*

*<option value="120" @11>2</option>*

*<option value="180" @12>3</option>*

*<option value="240" @13>4</option>*

*<option value="300" @14>5</option>*

*<option value="360" @15>6</option>*

*<option value="420" @16>7</option>*

*<option value="480" @17>8</option>*

*<option value="540" @18>9</option>*

*<option value="600" @19>10</option>*

*</select>*

*<span class="unit">\aa</span><br/>*

*</div>*

*<div class="caption">\098</div>*

*<div class="gap2 securityLb">*

*<span class="lb">\099</span><input type="text" maxlength="31" size="31" name="authkey" value="@00"/><br/>*

*</div>*

*<input type="submit" name="action" value="\014" class="btn\001"/>&nbsp;*

*<input type="reset" name="cancel" value="\015" class="btn\001"/>*

*</div>*

*</form>*

fs\_loadfixfile送出適當的內容，顯示的Handle：

*int fs\_switch\_com(char \*dest, char \*source , HTML\_IDX idx, char \*b, u16\_t max\_send\_len) {*

*case idx\_security\_html:*

*if(!strcmp(b,"00")) {*

*GetEEPPPBESecretPhrase(eep\_data\_buf);*

*tx\_string\_byte(eep\_data\_buf);*

*}*

*else if(\*b == '1') {*

*if(((\*(b+1) - '/')\*60) == session\_timeout)*

*tx\_string\_byte(SelectedStr);*

*}*

*break;*

*…*

*}*

**特別的機制及易遺漏：**

1. 如何讓使用者點Apply的動作，網址列不顯示其cgi。(不然F5重整又會發同樣的Apply cgi)

=> 將 cgi 轉址給 seeother，讓seeother重新導向目標網址。(ex. html)

2. 程式中有一個hostname，其用意是？

=> 使用者如是透過DN連到RMCARD，則HTML標籤hostname需記錄並回應。若RMCARD hostname欄位一律回Local IP Address，則會導致透過DN連結的Web User無法被正確的導向，導致存取異常。

3. @為變動填入的內容，其長度限制？

=> 一般建議不超過200。

4. 例如event logs網頁，裏面最多可以2048筆log，所以有2048個’@’？

=> 此類內容 (1)多 (2)有序，通常以『續傳』的方式實作。僅使用一個@處理。

5. 動態網頁 AJAX？

=> 可參考EPDU outlet control、Summary、Email Test、NTP… 網頁。

6. 網頁多國語言

=> 目前僅EPDU有完整法文版本，結構部分以’\’為關鍵字，後接3碼數值為通用字串。接2碼字元(a~z)為特屬於該網頁的字串。(分兩類為節省Flash存的字串量，以避免Flash滿出來… )

***\*css中會用到’\’，所以此規則需特別排除css類型文件。***

7. POST如何實現？

=> 看Code

**結論：**

讓我們看Code吧。實際Handle的細節非常多，文件目前只有針對GET以及網頁結構做介紹。實際上還有POST這個東西要處理。Web Server原本的架構就不針對POST處理，因此有關POST的內容較散亂。