TP-LINK®			**Name:** Description - Automated PIN-unlock on AJAX-based webservers.docx
№	**Number, version**	**Date**	**Author**
1	Revision 1.0	2016-02-01	Konstantin Mauch, Senior Software Engineer

Introduction

After drivers are installed by platform for LTE-modem K3773, the PIN-code needs to be unlocked on it; unless it’s done the traffic (outgoing one, with HTTP-requests, targeted onto real internet addresses) stops at local interface of LTE-modem.

To unlock PIN-code on this modem it’s required to open local web-page and insert character sequence ‘1234’ (it’s a default one, and it can be altered on same we-page). Having received such character sequence the web-server of LTE-modem stops blocking internet requests, so the outgoing and incoming traffic is available on CPE via LTE0 interface, provided by LTE-modem K3773 plugged into USB port of CPE.

This document describes the two competing approached to automate the insertion PIN-code to modem’s web-page, also it defines some practical use of similar solutions for different tasks.

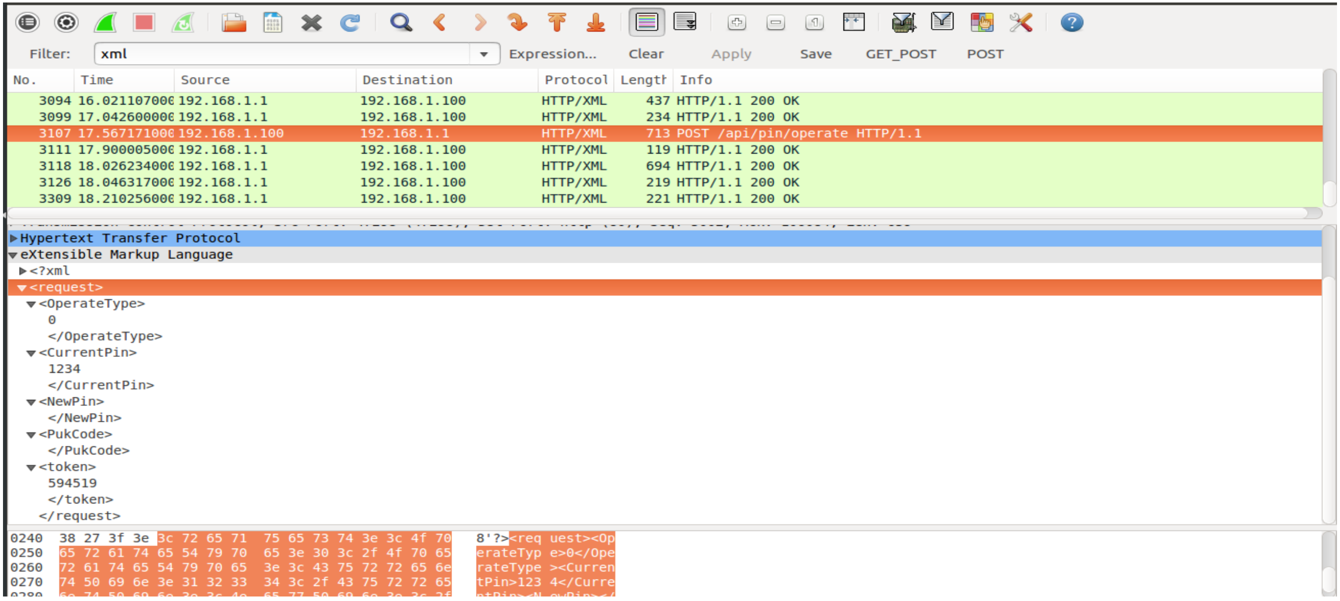
First approach, libCURL-based solution

Having analyzed the HTTP (and bare TCP) traffic which flows between CPE and K3773 it was discovered that series of HTTP/XML requests are generated on clicking (necessary for PIN-insertion) buttons of web-page. Apparently, these requests caotain all mandatory data to be passed to web-server of K3773 in order to get PIN-code unlocked. One of such packets is:

```
<?xml version='1.0' encoding='UTF-8'?>
<request>
    <OperateType>
        0
    </OperateType>
    <CurrentPin>
        1234
    </CurrentPin>
    <NewPin>
    </NewPin>
    <PukCode>
    </PukCode>
    <token>
        594519
    </token>
</request>
```

Also there are few other more or less important packets, but first let’s check out the content of above packet. As one can see on [PICTURE 1], besides PIN-code itself it contains some specific value, called ‘Tocken’. Without passing the correct value of ‘Tocken’ to web-server, the last one won’t accept it as correct one, so the whole packet will have no effect on server’s side.

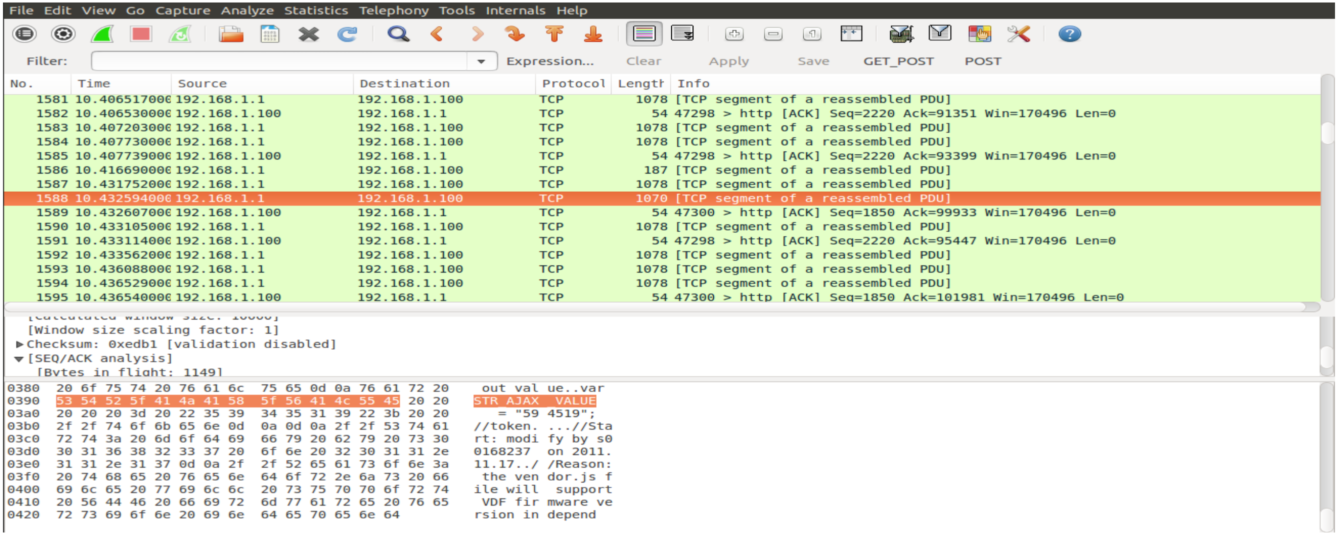
PICTURE 1. PIN-unlock HTTP/XML packet requires 'Tocken' value to defined correctly



```
Filter: xml

No.   Time             Source          Destination     Protocol Length Info
 3094 16.021107000 192.168.1.1     192.168.1.100   HTTP/XML    437 HTTP/1.1 200 OK
 3099 17.042600000 192.168.1.1     192.168.1.100   HTTP/XML    234 HTTP/1.1 200 OK
 3107 17.567171000 192.168.1.100   192.168.1.1     HTTP/XML    713 POST /api/pin/operate HTTP/1.1
 3111 17.900005000 192.168.1.1     192.168.1.100   HTTP/XML    119 HTTP/1.1 200 OK
 3118 18.026234000 192.168.1.1     192.168.1.100   HTTP/XML    694 HTTP/1.1 200 OK
 3126 18.046317000 192.168.1.1     192.168.1.100   HTTP/XML    219 HTTP/1.1 200 OK
 3309 18.210256000 192.168.1.1     192.168.1.100   HTTP/XML    221 HTTP/1.1 200 OK

▶Hypertext Transfer Protocol
▼eXtensible Markup Language
 ▶<?xml
 ▼<request>
  ▼<OperateType>
     0
     </OperateType>
  ▼<CurrentPin>
     1234
     </CurrentPin>
  ▼<NewPin>
     </NewPin>
  ▼<PukCode>
     </PukCode>
  ▼<token>
     594519
     </token>
   </request>

0240  38 27 3f 3e 3c 72 65 71  75 65 73 74 3e 3c 4f 70   8'?><req uest><Op
0250  65 72 61 74 65 54 79 70  65 3e 30 3c 2f 4f 70 65   erateTyp e>0</Ope
0260  72 61 74 65 54 79 70 65  3e 3c 43 75 72 72 65 6e   rateType ><Curren
0270  74 50 69 6e 3e 31 32 33  34 3c 2f 43 75 72 72 65   tPin>123 4</Curre
```

This means that the correct 'Tocken' value should be computed first. This value – traffic analysis shows that – we can not find it among HTTP packets preceding to PIN-code packet shown on the [PICTURE 1]. But it should be somewhere near, and by the way should precede to PIN-code packet (see [PICTURE 1]) otherwise at the moment o sending PIN-code packet the client side (CPE) would not know which value to use as 'Tocken'. Due this the more deeper search was undertaken, and 'Tocken' value was found as value of variable 'STR_AJAX_VALUE', but this time not inside the HTTP-traffic, rather inside bare TCP (since AJAX puts parts of its answers on TCP). Please check [PICTURE 2].

PICTURE 2. Value of 'Tocken' comes as value of ST_AVAX_VALUE variable over TCP traffic


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression...
Clear
Apply
Save
GET_POST
POST

```
No.   Time             Source          Destination     Protocol Length Info
 1581 10.406517000 192.168.1.1     192.168.1.100   TCP        1078 [TCP segment of a reassembled PDU]
 1582 10.406530000 192.168.1.100   192.168.1.1     TCP          54 47298 > http [ACK] Seq=2220 Ack=91351 Win=170496 Len=0
 1583 10.407203000 192.168.1.1     192.168.1.100   TCP        1078 [TCP segment of a reassembled PDU]
 1584 10.407730000 192.168.1.1     192.168.1.100   TCP        1078 [TCP segment of a reassembled PDU]
 1585 10.407739000 192.168.1.100   192.168.1.1     TCP          54 47298 > http [ACK] Seq=2220 Ack=93399 Win=170496 Len=0
 1586 10.416690000 192.168.1.1     192.168.1.100   TCP         187 [TCP segment of a reassembled PDU]
 1587 10.431752000 192.168.1.1     192.168.1.100   TCP        1078 [TCP segment of a reassembled PDU]
 1588 10.432594000 192.168.1.1     192.168.1.100   TCP        1070 [TCP segment of a reassembled PDU]
 1589 10.432607000 192.168.1.100   192.168.1.1     TCP          54 47300 > http [ACK] Seq=1850 Ack=99933 Win=170496 Len=0
 1590 10.433105000 192.168.1.1     192.168.1.100   TCP        1078 [TCP segment of a reassembled PDU]
 1591 10.433114000 192.168.1.100   192.168.1.1     TCP          54 47298 > http [ACK] Seq=2220 Ack=95447 Win=170496 Len=0
 1592 10.433562000 192.168.1.1     192.168.1.100   TCP        1078 [TCP segment of a reassembled PDU]
 1593 10.436088000 192.168.1.1     192.168.1.100   TCP        1078 [TCP segment of a reassembled PDU]
 1594 10.436529000 192.168.1.1     192.168.1.100   TCP        1078 [TCP segment of a reassembled PDU]
 1595 10.436540000 192.168.1.100   192.168.1.1     TCP          54 47300 > http [ACK] Seq=1850 Ack=101981 Win=170496 Len=0

  [Window size scaling factor: 1]
 ▶Checksum: 0xedb1 [validation disabled]
 ▼[SEQ/ACK analysis]
   [Bytes in flight: 1149]

0380  20 6f 75 74 20 76 61 6c  75 65 0d 0a 76 61 72 20    out val ue..var
0390  53 54 52 5f 41 4a 41 58  5f 56 41 4c 55 45 20 20   STR AJAX  VALUE
03a0  20 20 20 3d 20 22 35 39  34 35 31 39 22 3b 20 20      = "59 4519";
03b0  2f 2f 74 6f 6b 65 6e 0d  0a 0d 0a 2f 2f 53 74 61   //token. ...//Sta
03c0  72 74 3a 20 6d 6f 64 69  66 79 20 62 79 20 73 30   rt: modi fy by s0
03d0  30 31 36 38 32 33 37 20  6f 6e 20 32 30 31 31 2e   0168237  on 2011.
03e0  31 31 2e 31 37 0d 0a 2f  2f 52 65 61 73 6f 6e 3a   11.17../ /Reason:
03f0  20 74 68 65 20 76 65 6e  64 6f 72 2e 6a 73 20 66    the ven dor.js f
0400  69 6c 65 20 77 69 6c 6c  20 73 75 70 70 6f 72 74   ile will  support
0410  20 56 44 46 20 66 69 72  6d 77 61 72 65 20 76 65    VDF fir mware ve
0420  72 73 69 6f 6e 20 69 6e  64 65 70 65 6e 64         rsion in depend
```

Having captured the 'Tocken' from TCP stream, and then having embedded it into HTTP/XML request (within PIN-code packet) we achieve simulation of passing a *correct* HTTP-data to web-server of K3773, thus the PIN-code becomes unlocked on server side.

This solution was implemented in code. (Please, see listing 1.) Advantage of such solution is it given 100%-guaranteed result because we simulate whole HTTP traffic which web-server expects from CPS side (i.e. we simulate presence of internet browser on CPE's side). Disadvantages: the traffic being simulated seems to be a bit excessive, and for linking the program (please, see listing 1) requires libCURL library, which can tax platform's resource (permanent memory needed for storage of library in filesystem, and random-access memory needed during program and library's runtime). The solution (please, see listing 1) has been implemented and tested on TD VG5612 (instance of CPE), and has proven to work well with K3773.

Second approach, socket-based solution

Assuming that disadvantages described in previous chapter may become an impediment on platforms with high or specific request as for free system resources, the another solution was invented and implemented. This time the exploit of exactly K3773 was exploited, namely: passing PIN-code packet with 'Tocken' value equal to '0' at the begin of K3773 system's life will have the same effect as passing correct value (discovered in the way described by previous chapter). Also, this time there's no need to deploy libCURL, because simple instance of HTTP traffic is easy reproducible by means of TCP sockets. (Please, see listing 2.)

The approach represented by listing 2 was implemented and tested on same TD VG5612 and has proven to work same well with K3773.

Obviously such approach implies modifications most apparent of which are:

a) Simulating and entire set of HTTP-requests - see variable [aAddrArray] in listing 1 – over TCP sockets;
b) Deployment of brute force attack on K3773 once the 'Tocken' value of ***0*** appears to be inappropriate.

The second modification was implemented and tested (see listing 3) on same TD VG5612 and has proven to affect the K3773. Originally it was expected that that going throughout first 500 000 {???} possible values of 'Tocken' during 20-30 minutes attack will result in passing obtaining correct response form server side as soon as value of 'Tocken' reaches the correct one, but profound testing has shown that K3773 goes into reboot within first 2-3 minutes , i.e. 'Tocken':: <0, … , 4069> . So this approach, despite *is correct from theoretical point of view*, was recognized as *practically unusable* one and for that reason was skipped form further consideration.

Practical use of similar solutions in *other* tasks related to platform evaluation

Both approaches implemented and exposed for check by this document are effective on exact model of USB-modem only, namely – on LTE-modem Vodafone-HUAWEI K3773. Which means that on other modems this code will have no effect (in best case) and slight negative effect in average cases. For this reason none of the above solutions is recommended for insertion into platform code.

Nevertheless, the similar solutions may appear of some help while platform evaluation, namely automated passing HTTP/XML packets from ACS server to CWMP may save a lot of time while doing tasks such as "Description - extended IGMP Diagnostics in CPE - Handling Channel Protocol value.pdf". For instance, to fill the parameter names and their values in ACS server form (please, check the p. 9 of "Description - extended IGMP Diagnostics in CPE - Handling Channel Protocol value.pdf") it was necessary up to 5 minutes (with all preparations, and double-checks). Once the libCURL-based automation of this process was introduced the total time of task processing would be 1-2 hours less.

TP-LINK®			**Name:** Description - Automated PIN-unlock on AJAX-based webservers.docx
№	**Number, version**	**Date**	**Author**
1	Revision 1.0	2016-02-01	Konstantin Mauch, Senior Software Engineer

Contents

Listing 1. libCURL-based solution

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <curl/curl.h>

#define DEF_ABSENT_TOKEN (-1)

#define CORRECT_RESPONCE_SIZE           65
#define SECONDS_TO_RELAX      1

#define SUCCESS                   0
#define ERR_NONE_FOUND                          (-1)
#define ERR_CANT_SEND                           (-2)


int m_Tocken;

char * aAddrArray[] = {
		"http://192.168.1.1/",
		"http://192.168.1.1/html/launch.html",
		"http://192.168.1.1/html/js/lib/jquery-1.6.1.min.js",
		"http://192.168.1.1/html/js/launch.js",
		"http://192.168.1.1/api/monitoring/converged-status",
		"http://192.168.1.1/favicon.ico",
		"http://192.168.1.1/html/home.htm?startPage=pin-required",
		"http://192.168.1.1/html/css/main.css",
		"http://192.168.1.1/html/css/structure.css",
		"http://192.168.1.1/html/css/menu.css",

		"http://192.168.1.1/html/css/modules.css",
		"http://192.168.1.1/html/css/common.css",
		"http://192.168.1.1/html/css/site.css",
		"http://192.168.1.1/html/js/lib/jquery-1.6.1.min.js",
		"http://192.168.1.1/html/js/lib/jquery.showhide.js",
		"http://192.168.1.1/html/js/lib/jquery.tmpl.min.js",
		"http://192.168.1.1/html/css/print-home.css",
		"http://192.168.1.1/html/js/lib/log4javascript.js",
		"http://192.168.1.1/html/js/Extend_jQuery.js",
		"http://192.168.1.1/html/js/json2.js",

		"http://192.168.1.1/html/js/Util.js",
		"http://192.168.1.1/html/js/ConnectionManager.js",
		"http://192.168.1.1/html/js/CallbackWrapper.js",
		"http://192.168.1.1/html/js/APIProvider.js",
		"http://192.168.1.1/html/js/HTMLTemplates.js",
		"http://192.168.1.1/html/js/Injections.js",
		"http://192.168.1.1/html/js/WebUIProductName.js",
		"http://192.168.1.1/html/js/WebUIVersion.js",
		"http://192.168.1.1/html/js/SMS.js",
		"http://192.168.1.1/html/js/TagSubstitution.js",
		"http://192.168.1.1/html/js/UIUpdate.js",

		"http://192.168.1.1/html/js/UserNotification.js",
		"http://192.168.1.1/html/js/WifiShared.js",
		"http://192.168.1.1/html/js/WifiUIUpdate.js",
		"http://192.168.1.1/html/js/authentication.js",
		"http://192.168.1.1/html/js/moduleControls.js",
		"http://192.168.1.1/html/js/XmlCache.js",
		"http://192.168.1.1/html/js/PageLoader.js",
		"http://192.168.1.1/html/js/FileUploader.js",
		"http://192.168.1.1/html/js/device.js",
		"http://192.168.1.1/html/js/vendor.js",

#if 0
```

```
/* by shuttling this comment-out border _upside we may
  minimize the amount of packets to send. But it's not
  recommended, since only entire set is guaranteed to
  be sifficient in order to make modem to give the
  responce with 'Tocken' value encapsulate. */

            "http://192.168.1.1/html/js/vendorWifi.js",
            "http://192.168.1.1/html/js/DeviceSpecificHomeImage.js",
            "http://192.168.1.1/html/js/home.js",
            "http://192.168.1.1/html/img/icons/caution.gif",
            "http://192.168.1.1/html/img/icons/arrowUpSpeed.gif",
            "http://192.168.1.1/html/img/icons/arrowDownSpeed.gif",
            "http://192.168.1.1/html/img/icons/loading.gif",
            "http://192.168.1.1/html/js/lib/jquery.cycle.all.min.js",
            "http://192.168.1.1/api/ussd/release",
            "http://192.168.1.1/api/monitoring/status",
            "http://192.168.1.1/api/net/current-plmn",
            "http://192.168.1.1/api/device/information",
            "http://192.168.1.1/api/dhcp/settings",
            "http://192.168.1.1/html/OpCo/00.js?_=1453778840227",
            "http://192.168.1.1/html/img/backgrounds/body_bg_high.gif",
            "http://192.168.1.1/api/dialup/connection",
            "http://192.168.1.1/html/img/icons/battery3Bars.gif?1453778840514",
            "http://192.168.1.1/api/monitoring/traffic-statistics",
            "http://192.168.1.1/api/device/information",
            "http://192.168.1.1/api/dialup/profiles",
            "http://192.168.1.1/api/dialup/profiles",
            "http://192.168.1.1/api/language/current-language",
            "http://192.168.1.1/html/Language/en-gb.js",
            "http://192.168.1.1/html/Language/en-gb.js?_=1453778840871",
            "http://192.168.1.1/html/pin-required.htm",
            /* (POST) /api/language/current-language", */
            "http://192.168.1.1/api/device/nv-configuration",
            "http://192.168.1.1/api/device/nv-configuration",
            "http://192.168.1.1/api/device/nv-configuration",
            "http://192.168.1.1/api/device/nv-configuration",
            "http://192.168.1.1/api/device/nv-configuration",
            "http://192.168.1.1/api/device/nv-configuration",
            "http://192.168.1.1/api/device/nv-configuration",
            "http://192.168.1.1/api/device/nv-configuration",
            "http://192.168.1.1/html/js/pin-required.js",
            "http://192.168.1.1/html/img/backgrounds/drop_shadow.gif",
            "http://192.168.1.1/html/img/landing/quickstart_heroimage.jpg",
            "http://192.168.1.1/html/quickStartHeader.htm",
            "http://192.168.1.1/api/pin/status",
            "http://192.168.1.1/api/pin/status",
            "http://192.168.1.1/api/pin/status",
            "http://192.168.1.1/",
            "http://192.168.1.1/",
            "http://192.168.1.1/",
            "http://192.168.1.1/html/img/tabs/tab_global_active_left.gif",
            "http://192.168.1.1/html/img/tabs/tab_global_active_right.gif",
            "http://192.168.1.1/html/img/tabs/tab_global_inactive_left.gif",
            "http://192.168.1.1/html/img/tabs/tab_global_inactive_right.gif",
            "http://192.168.1.1/html/img/menu/menu_bg.gif",
            "http://192.168.1.1/html/img/backgrounds/gradient.gif",
            "http://192.168.1.1/api/device/nv-configuration",
            "http://192.168.1.1/html/img/buttons/btn_darkgreen_sprite.png",
            "http://192.168.1.1/html/img/backgrounds/sidebar_accordion_sprite.gif GET /",
            "http://192.168.1.1/html/img/icons/signal0Bars.gif",
            "http://192.168.1.1/html/img/icons/networkStatusCross.gif",
            "http://192.168.1.1/html/img/icons/time.gif",
            "http://192.168.1.1/api/device/nv-configuration",
            "http://192.168.1.1/html/img/menu/logo.gif",
            "http://192.168.1.1/api/monitoring/check-notifications",
            "http://192.168.1.1/api/monitoring/status",
            "http://192.168.1.1/api/net/current-plmn",
            "http://192.168.1.1/api/dialup/connection",
            "http://192.168.1.1/api/monitoring/traffic-statistics",
            "http://192.168.1.1/api/monitoring/check-notifications",
            "http://192.168.1.1/api/monitoring/status",
            "http://192.168.1.1/api/net/current-plmn",
            "http://192.168.1.1/api/dialup/connection",
            "http://192.168.1.1/api/monitoring/traffic-statistics",
            "http://192.168.1.1/api/monitoring/check-notifications",
        #endif
            ""
        };

        static int m_TockenFound;

        static void dump(const char *text, FILE *stream, unsigned char *ptr, size_t size)
        {
        size_t i;
        size_t c;
        unsigned int width=0x10;
        int iTocken=0;
        char cBuf[512];
        char * cpAjaxPtr;
        char * cp1, * cp2;

            fprintf(stream, "%s, %10.10ld bytes (0x%8.8lx)\n", text, (long)size, (long)size);

            if ( NULL != ( cpAjaxPtr = strstr(ptr, "var STR_AJAX_VALUE") ) )
            {
                cp1 =  strtok(cpAjaxPtr, "\"");

                cp2 =  strtok(NULL, "\"");

                m_TockenFound = 1;

                m_Tocken = atoi (cp2);

                printf("[%s] iTocken = <%s>  m_Tocken = <%d>. SUCCESS. TERMINATING. <m_TockenFound=%d>\n", cpAjaxPtr, cp2, m_Tocken, m_TockenFound);

            }

            for(i=0; i<size; i+= width)
            {
                fprintf(stream, "%4.4lx: ", (long)i);

                /* show hex to the left */
                for(c = 0; c < width; c++)
                {
                    if(i+c < size)

                        fprintf(stream, "%02x ", ptr[i+c]);
                    else
                        fputs("   ", stream);
                }

                /* show data on the right */
                for(c = 0; (c < width) && (i+c < size); c++)

                    fputc((ptr[i+c]>=0x20) && (ptr[i+c]<0x80)?ptr[i+c]:'.', stream);

                fputc('\n', stream); /* newline */

            }
        }

        static int my_trace(CURL *handle, curl_infotype type, char *data, size_t size, void *userp)
        {
        const char *text;

        (void)handle;
```

```
    switch (type)
    {
        case CURLINFO_TEXT:
            //fprintf(stderr, "== Info: %s", data);
            fprintf(stdout, "== Info: %s", data);
        default: /* in case a new one is introduced to shock us */
        return 0;

        case CURLINFO_HEADER_OUT:
            text = "=> Send header";
        break;

        case CURLINFO_DATA_OUT:
            text = "=> Send data";
        break;

        case CURLINFO_SSL_DATA_OUT:
            text = "=> Send SSL data";
        break;

        case CURLINFO_HEADER_IN:
            text = "<= Recv header";
        break;

        case CURLINFO_DATA_IN:
            text = "<= Recv data";
        break;

        case CURLINFO_SSL_DATA_IN:
            text = "<= Recv SSL data";
        break;
    }

    dump(text, stdout, (unsigned char *)data, size);
    return 0;
} /* int my_trace */


int find_tocken(CURL * curl)
{
int iTocken, i=0;
CURLcode iRes;

    iTocken = m_Tocken = DEF_ABSENT_TOKEN;

    while (aAddrArray[i] != "")
    {
        iRes = curl_easy_setopt(curl, CURLOPT_URL, aAddrArray[i]);

        iRes = curl_easy_perform(curl);

        printf  ("find_tocken:m_TockenFound = <%d>\n", m_TockenFound);

        if (m_TockenFound)
        {

            iTocken = m_Tocken;

            printf ("Tocken found <%d> , near addr str idx. <%d> \n", iTocken, i);

            break;
        }

        i++;
    }

    return iTocken;

} /* int find_tocken */
```

```
/* POST'ing a XML payload over here we put the PIN code into a modem */
const char * PIN_PAGE = "http://192.168.1.1/api/pin/operate";

/* On thie page we reconnect reconnect  */
const char * DIAL_PAGE = "http://192.168.1.1/api/dialup/dial";


typedef struct _RespStruct
{
        char * cpResponce;

        size_t iSize;

} RespStruct, *pRespStruct;

/* Suppose 32K is enough to include _any HTML responce frmo _this LTE-modem */
char cBuffer[0x400*32];


/* Callback to execute on arrival of HTML responce */
static size_t RecvClbk(void *contents, size_t size, size_t nmemb, void *userp)
{
size_t realsize = size * nmemb;

        RespStruct * HtmlRespStruct = (RespStruct *)userp;

        HtmlRespStruct->cpResponce = (char*)(cBuffer);

        memcpy(HtmlRespStruct->cpResponce, contents, realsize);

        HtmlRespStruct->cpResponce[realsize] = 0;

        HtmlRespStruct->iSize = realsize;

        return realsize;
} /* size_t RecvClbk */



int main(void)
{
CURL *curl;
CURLcode res, iRes;
int iTocken, i=0;
RespStruct RespStr;

        curl = curl_easy_init();

        if(curl)
        {
                iRes = curl_easy_setopt(curl, CURLOPT_DEBUGFUNCTION, my_trace);

                /* the DEBUGFUNCTION has no effect until we enable VERBOSE */
                iRes = curl_easy_setopt(curl, CURLOPT_VERBOSE, 1L);

                /* we tell libcurl to follow redirection */
                iRes = curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);

                m_TockenFound = 0;

                /* find actual tocken to wirk with */
                iTocken = find_tocken(curl);


                i = 0; while (aAddrArray[i++] != "") printf("%d\n", i);

                if (DEF_ABSENT_TOKEN  != iTocken)
        {
#if defined(PRECAREOUS)
            curl_easy_setopt(curl, CURLOPT_ACCEPT_ENCODING, "gzip, deflate");
            curl_easy_setopt(curl, CURLOPT_VERBOSE, 1L);
            curl_easy_setopt(curl, CURLOPT_COOKIEFILE, "");
#endif /* (PRECAREOUS) */

            /* send all data to this function  */
            curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, RecvClbk);

            /* we pass our 'RespStr' struct to the callback function */
            curl_easy_setopt(curl, CURLOPT_WRITEDATA, (void *)&RespStr);
#if defined(PRECAREOUS)
            curl_easy_setopt(curl, CURLOPT_USERAGENT, "libcurl-agent/1.0");
#endif /* (PRECAREOUS) */

                /* Buffer to compose a XML payload */
                char cAutoPostString[1024];

                sprintf (cAutoPostString, "<?xml version='1.0' encoding='UTF-
8'?><request><OperateType>0</OperateType><CurrentPin>1234</CurrentPin><NewPin></NewPin><PukCode></PukCode><token>%d</token></request>",
iTocken);

                curl_easy_setopt(curl, CURLOPT_POSTFIELDS, cAutoPostString);
                curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, (long)strlen(cAutoPostString));
                curl_easy_setopt(curl, CURLOPT_URL, PIN_PAGE);

                /* Clean the buffer before receiving a responce into it  */
                memset (&RespStr, sizeof (struct _RespStruct) , 0);

                iRes = curl_easy_perform(curl);

                /* Proccess error */
                if(iRes != CURLE_OK)
                {
                    fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(iRes));

                    return    ERR_CANT_SEND;
                }
                else
                {
                    if (CORRECT_RESPONCE_SIZE == strlen(RespStr.cpResponce) )
                    {
                        printf("%s\n[SUCCESS] Tocken: %d; retieved: %d\n", RespStr.cpResponce, iTocken,
RespStr.iSize);


                        //break;
                        return SUCCESS;
                    }
                    else if ((101 == RespStr.iSize) && ('1'==RespStr.cpResponce[55] &&
'0'==RespStr.cpResponce[56] &&'0'==RespStr.cpResponce[57] &&'3'==RespStr.cpResponce[58] &&'0'==RespStr.cpResponce[59] &&
'2'==RespStr.cpResponce[60] ) )
                    {
                        printf("%s[%c%c%c%c%c%c]\n[PIN ALREADY INSERTED] Tocken: %d; retieved: %d\n",

                        RespStr.cpResponce,
                        RespStr.cpResponce[55],
                        RespStr.cpResponce[56],
                        RespStr.cpResponce[57],
                        RespStr.cpResponce[58],
                        RespStr.cpResponce[59],
                        RespStr.cpResponce[60],
                         iTocken, RespStr.iSize);
                    }
                    else
                        printf("%s\n[FAILURE] Tocken: %d; retieved: %d\n", RespStr.cpResponce, iTocken,
RespStr.iSize);

                } /* CURLE_OK == iRes */
```

```
                        } /* if (DEF_ABSENT_TOKEN  != iTocken) */

                curl_easy_cleanup(curl);
        } /*            if(curl) */


        return 0;
}
```

Listing 2. Second approach: socket-based solution

```
#include <netinet/in.h>
#include <stdbool.h>
#include <memory.h>
#include <string.h>
#include <assert.h>
#include <stdio.h>

#define _512                    0x200
#define SEND2RECV_TMO   300000
#define MODEM_PAGE_ADDR    "192.168.1.1"
#define MODEM_PAGE_PORT 80
#define NUM_ATTEMPTS          3

static int tcpConnect(int * piFd, char *server, int port);
static int sendMsg(int fd, char *msg, int len);
static int recvMsg(int fd, char *msg, int len);

char * _disconnect =  "POST /api/dialup/dial HTTP/1.1\r\n"
            "Host: 192.168.1.1\r\n"
            "Accept: */*\r\n"
            "Content-Length: 91\r\n"
            "Content-Type: application/x-www-form-urlencoded\r\n"
            "\r\n"
            "<?xml version='1.0' encoding='UTF-8'?><request><Action>0</Action><token>0</token></request>";

char * _connect =      "POST /api/dialup/dial HTTP/1.1\r\n"
            "Host: 192.168.1.1\r\n"
            "Accept: */*\r\n"
            "Content-Length: 91\r\n"
            "Content-Type: application/x-www-form-urlencoded\r\n"
            "\r\n"
            "<?xml version='1.0' encoding='UTF-8'?><request><Action>0</Action><token>1</token></request>";

char * _pin =             "POST /api/pin/operate HTTP/1.1\r\n"
            "Host: 192.168.1.1\r\n"
            "Accept: */*\r\n"
            "Content-Length: 166\r\n"
            "Content-Type: application/x-www-form-urlencoded\r\n"
            "\r\n"
            "<?xml version='1.0' encoding='UTF-
8'?><request><OperateType>0</OperateType><CurrentPin>1234</CurrentPin><NewPin></NewPin><PukCode></PukCode><token>0</token></request>";


int post (int fd, char *serverIp, int port, char *_str, int recvTimes)
{
char recvBuf[_512]    = {0};

            if (strlen(_str) != sendMsg(fd, _str, strlen(_str)))
            {
                        printf("message was not sent\n");

                        return -1;
            }
            else
                        printf("message sent: %s\n", _str);

            usleep(SEND2RECV_TMO);

            memset(recvBuf, 0, _512);

            if (0 >= recvMsg(fd, recvBuf, _512) )
            {
                        printf("message was not received\n");

                        return -1;
            }
            else
```

```
    {
        printf("message received:\n");

        int i;

        for (i = 0; i < _512; i++)
            printf("%c",recvBuf[i]);
    }

} /* post (int fd, char *serverIp, int port, char *_str, int recvTimes) */

static int sendMsg(int fd, char *msg, int len)
{
int nsend = -1;

    assert(NULL != msg);

    nsend = send(fd, msg, len, 0);

    if (nsend < len)
    {
        printf("wrong length while sending message\n");

        return -1;
    }

    return nsend;
} /* sendMsg(int fd, char *msg, int len) */

static int recvMsg(int fd, char *msg, int len)
{
    assert(NULL != msg);

    return recv(fd, msg, len, 0);
} /* recvMsg(int fd, char *msg, int len) */


static int tcpConnect(int * piFD, char * server, int port)
{
struct sockaddr_in serverAddr;

    *piFD = socket(PF_INET, SOCK_STREAM, 0);

    if (*piFD <= 0)
    {
        printf("Fail to create socket.\n");

        return -1;
    }

    memset(&serverAddr, 0, sizeof(struct sockaddr_in));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(port);

    if (0 == inet_aton(server, &(serverAddr.sin_addr)))
    {
        printf(" can't get host entry %s   ,  %d\n", server, inet_aton(server, &(serverAddr.sin_addr))  );

        close(*piFD);

        return -1;
    }
    else
    {
        printf("addresss is valid  %s  \n", server  );
    }

    if (-1 == connect(*piFD, (struct sockaddr *)&serverAddr, sizeof(serverAddr)) )
    {
```

```
                printf("Fail to connect to %s.\n", server);

                close(*piFD);

                return -1;
        }

        printf("Connected to %s allright.\n", server);

        return *piFD;

} /* tcpConnect(char * server, int port) */

void main()
{
int iFD;

        if ( -1 == tcpConnect(&iFD, MODEM_PAGE_ADDR, MODEM_PAGE_PORT) )
        {
                printf("ERROR: Can't connect socket. Exiting.");
        }

        post (iFD, MODEM_PAGE_ADDR, MODEM_PAGE_PORT, _disconnect,  NUM_ATTEMPTS);

        post (iFD, MODEM_PAGE_ADDR, MODEM_PAGE_PORT, _connect,  NUM_ATTEMPTS);

        post (iFD, MODEM_PAGE_ADDR, MODEM_PAGE_PORT, _pin,  NUM_ATTEMPTS);

        close(iFD);
}
```

Listing 3. ‘Brute force attack’ based solution

```
/* strlen() */
#include <string.h>

/* sleep() */
#include <unistd.h>

/* curl_easy_init() */
#include <curl/curl.h>

/* Length of packed with 'OK' idetifier */
#define CORRECT_RESPONCE_SIZE           65

/* Length of packed with 'already assigned' err. code 100302 */
#define ALREADY_ASSIGNED_SIZE           101

/* Amount of seconds to wait between POST ranges; i.e. time to relax on modem' side. */
#define RELAX_TIME                  1

#define SUCCESS                     0

#define ERR_INIT                    (-1)
#define ERR_SEND                    (-2)
#define ERR_FIND                    (-3)


/* POST'ing a XML payload over here we put the PIN code into a modem */
const char * PIN_PAGE = "http://192.168.1.1/api/pin/operate";

#if defined(PRECAREOUS)
/* On this page we disconnect and connect */
const char * DIAL_PAGE = "http://192.168.1.1/api/dialup/dial";
#endif /* (PRECAREOUS) */

/* Struct to incorporate HTML responce data and its length */
typedef struct _RespStruct
{
        char * cpResponce;

        size_t iSize;

} RespStruct, *pRespStruct;

/* Suppose 32K is enough to include HTML responce */
char cBuffer[0x400*32];

/* Callback to execute on arrival of HTML responce */
static size_t RecvClbk(void *contents, size_t size, size_t nmemb, void *userp)
{
size_t realsize = size * nmemb;

        RespStruct * HtmlRespStruct = (RespStruct *)userp;

        HtmlRespStruct->cpResponce = (char*)(cBuffer);

        memcpy(HtmlRespStruct->cpResponce, contents, realsize);

        HtmlRespStruct->cpResponce[realsize] = 0;

        HtmlRespStruct->iSize = realsize;

        return realsize;
}

int main(void)
{
CURLcode rc;
```

```
CURL *curl;

RespStruct RespStr;

/* Buffer to compose a XML payload of PIN-code POST message */
char cPinCodeXML[4096];

#if defined(PRECAREOUS)
/* Buffer to compose a XML payload of Connect/Disconnect POST message */
char cReconnectXML[512];
#endif /* (PRECAREOUS) */


// 100 - ok; 1024 - ok; 2048 - bad; 4096 - bad;
int iDelta = 2048, iMinToken, iMaxToken;

/* Externa cycle counter */
int j;

        curl_global_init(CURL_GLOBAL_ALL);

        curl = curl_easy_init();

        if (!curl)
        {
                printf("can't init cURL object\n");

                return ERR_INIT;
        }

#if defined(PRECAREOUS)
        /* Feturing behavior of browser */
        curl_easy_setopt(curl, CURLOPT_ACCEPT_ENCODING, "gzip, deflate");
        curl_easy_setopt(curl, CURLOPT_VERBOSE, 0/*1L*/);
        curl_easy_setopt(curl, CURLOPT_COOKIEFILE, "");

        /* Satisfying special instance of webserver */
        curl_easy_setopt(curl, CURLOPT_USERAGENT, "libcurl-agent/1.0");

        /* Disconnect. We need to reconnect to put modem's page into initial state. */
        sprintf (cReconnectXML, "<?xml version='1.0' encoding='UTF-8'?><request><Action>0</Action><token>%d</token></request>", 0);
        curl_easy_setopt(curl, CURLOPT_POSTFIELDS, cReconnectXML);
        curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, (long)strlen(cReconnectXML));
        curl_easy_setopt(curl, CURLOPT_URL, DIAL_PAGE);
        rc = curl_easy_perform(curl);
        printf("\ndisconnected\n");
        sleep (RELAX_TIME);

        /* Connect. We need to reconnect to put modem's page into initial state. */
        sprintf (cReconnectXML, "<?xml version='1.0' encoding='UTF-8'?><request><Action>1</Action><token>%d</token></request>", 0);
        curl_easy_setopt(curl, CURLOPT_POSTFIELDS, cReconnectXML);
        curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, (long)strlen(cReconnectXML));
        curl_easy_setopt(curl, CURLOPT_URL, DIAL_PAGE);
        rc = curl_easy_perform(curl);
        printf("\nconnected\n");
        sleep (RELAX_TIME);
#endif /* (PRECAREOUS) */

        /* Send all data to this function  */
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, RecvClbk);

        /* We pass our 'RespStr' struct to the callback function */
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, (void *)&RespStr);

        iMinToken = 0;
        iMaxToken = iDelta-1;

        /* Process a bunch of ranges */
        for (j = 0; j < (0x80*0x4); j++) // 'j' < 0x20 - is ok (as lon as iDelta is 1024), let's try bigger 'j'
        {
                /* For all except first run lets shift borders ahead */
                if (j) iMinToken += iDelta, iMaxToken += iDelta;

printf ( "processing range [%d.. %d] \n", iMinToken, iMaxToken);

                /* Internal cycle counter */
                int i;

                /* Proceed current range */
                for (i=iMinToken; i<=iMaxToken; i++)
                {
                        sprintf (cPinCodeXML, "<?xml version='1.0' encoding='UTF-
8'?><request><OperateType>0</OperateType><CurrentPin>1234</CurrentPin><NewPin></NewPin><PukCode></PukCode><token>%d</token></request>", i);

                        curl_easy_setopt(curl, CURLOPT_POSTFIELDS, cPinCodeXML);
                        curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, (long)strlen(cPinCodeXML));
                        curl_easy_setopt(curl, CURLOPT_URL, PIN_PAGE);

                        /* Clean the buffer before receiving a responce into it  */
                        memset (&RespStr, sizeof (struct _RespStruct) , 0);

                        rc = curl_easy_perform(curl);

                        /* Proccess error */
                        if(rc != CURLE_OK)
                        {
                                printf("curl_easy_perform() failed: %s\n", curl_easy_strerror(rc));

                                return  ERR_SEND;
                        }
                        else
                        {       /* Check if size is <65> and mesage is 'OK' */
                                if((CORRECT_RESPONCE_SIZE==RespStr.iSize) && ('O'==RespStr.cpResponce[50] && 'K'== RespStr.cpResponce[51]))
                                {
                                        printf("%s\n[SUCCESS] Token: %d; retieved: %d\n", RespStr.cpResponce, i, RespStr.iSize);

                                        return SUCCESS;
                                }
                                /* Check if size is <101> and mesage is '100302' */
                                else if (( ALREADY_ASSIGNED_SIZE == RespStr.iSize) && ('1'==RespStr.cpResponce[55] &&
'0'==RespStr.cpResponce[56] &&'0'==RespStr.cpResponce[57] &&'3'==RespStr.cpResponce[58] &&'0'==RespStr.cpResponce[59] &&
'2'==RespStr.cpResponce[60] ) )
                                {
                                        printf("%s\n[PIN ALREADY ASSIGNED] Token: %d; retieved: %d\n",
                                        RespStr.cpResponce, i, RespStr.iSize);

                                        return SUCCESS;
                                }
                                else
                                        printf("%s\n[FAILURE] Token: %d; retieved: %d\n", RespStr.cpResponce, i, RespStr.iSize);
                        }
                } /* Internal cycle <i> */

                /* Hopefully it eases the communication process on modem's side */
                sleep (RELAX_TIME);

        } /* External cycle <j> */

        /* Tip: the connection may be re-used on next run */
        curl_easy_cleanup(curl);

        return ERR_FIND;
}
```