Talos Vulnerability Report

TALOS-2020-0984

CoTURN HTTP Server POST-parsing information leak vulnerability

FEBRUARY 18, 2020

CVE NUMBER

CVE-2020-6061

Summary

An exploitable heap out-of-bounds read vulnerability exists in the way CoTURN 4.5.1.1 web server parses POST requests. A specially crafted HTTP POST request can lead to information leaks and other misbehavior. An attacker needs to send an HTTPS request to trigger this vulnerability.

Tested Versions

CoTURN 4.5.1.1

Product URLs

https://github.com/coturn/coturn

CVSSv3 Score

7.0 - CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:H

CWE

CWE-125: Out-of-bounds Read

Details

CoTURN is a TURN server implementation. A TURN Server is a VoIP media traffic NAT traversal server and gateway. CoTURN can be used as a general-purpose network traffic TURN server and gateway.

For administration purposes, it includes a web server. Code responsible for parsing POST request body variables contains a bug that can lead to out of bounds memory access.

When preparing to parse the POST request body, the following code is executed:

```
static struct headers_list * post_parse(char *data, size_t data_len)
{
  while((*data=='\r')||(*data=='\n')) ++data;                    [1]
  char *post_data = (char*)calloc(data_len + 1, sizeof(char));
  memcpy(post_data, data, data_len);                             [2]
```

To start, at [1] newline and carriage return characters are skipped in order to get to the start of POST data. However, while `data` pointer is incremented, the `data_len` isn't decremented. Then, at [2] , the `memcpy` call will copy data from incremented `data` pointer into newly allocated `post_data` memory buffer using the unchanged `data_len`. This results in bytes beyond the end of original `data` buffer being accessed.

A POST request of following form can be used to trigger this issue:

```
"POST /logon HTTP/1.1\r\nContent-Length: 32717\r\n\r\n"  + "\x0d"*33000 + "u\r\n\r\n\r\n"
```

Using extra `\r` or `\n` characters, we can control how much the `data` pointer gets incremented at [1]. Also, the content length header controls the allocation size. By aligning those two, we can have the `while` loop at [1] skip till the actual end of the `data` buffer which would result in a large out of bounds access at [2].

Depending on the memory layout, this could potentially result in further memory corruption, access to sensitive information from other requests and other unforeseen consequences.

Timeline

2020-02-11 - Vendor Disclosure
2020-02-17 - Vendor patched
2020-02-18 - Public Release

CREDIT

Discovered by Aleksandar Nikolic of Cisco Talos.