Bug 1866838 (CVE-2020-14354) - CVE-2020-14354 c-ares: ares_destroy() with pending ares_getaddrinfo() leads to Use-After-Free

Keywords:	Security ×	**Reported:**	2020-08-06 14:21 UTC by Michael Kaplan
Status:	CLOSED NOTABUG	**Modified:**	2021-03-17 08:41 UTC (History)
Alias:	CVE-2020-14354	**CC List:**	14 users (show)
Product:	Security Response	**Fixed In Version:**	c-ares 1.16.1
Component:	vulnerability	**Doc Type:**	If docs needed, set a value
Version:	unspecified	**Doc Text:**	A possible use-after-free and double-free in c-ares lib version 1.16.0 if ares_destroy() is called prior to ares_getaddrinfo() completing. This flaw possibly allows an attacker to crash the service that uses c-ares lib. The highest threat from this vulnerability is to this service availability.
Hardware:	All	**Clone Of:**	
OS:	Linux	**Environment:**	
Priority:	medium	**Last Closed:**	2020-08-12 12:43:59 UTC
Severity:	medium		
Target Milestone:	---		
Assignee:	Red Hat Product Security		
QA Contact:			
Docs Contact:			
URL:			
Whiteboard:			
Depends On:	~~1866849~~		
Blocks:	1866841 1939846		
TreeView+	depends on / blocked		

Attachments	(Terms of Use)
Add an attachment (proposed patch, testcase, etc.)	

Michael Kaplan 2020-08-06 14:21:37 UTC Description

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)
==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
```

References:

https://packetstormsecurity.com/files/158755/GS20200804145053.txt

Created nodejs tracking bugs for this issue:

Affects: fedora-all [~~Bug 1866846~~]

Alex 2020-08-12 10:25:14 UTC Comment 2

Acknowledgments:

Name: Jann Horn (Google Project Zero)

Alex 2020-08-12 10:56:06 UTC Comment 5

Mitigation:

If calling wait_ares(channel) before ares_destroy() in the service that uses c-ares, then this should prevent this bug.

Alex 2020-08-12 10:59:23 UTC Comment 7

The Red Hat Enterprise Linux not affected (all versions), because the latest version of c-ares being used is 1.13.0. And the bug introduced in 1.16.0 (and then fixed in 1.16.1).

Alex 2020-08-12 11:06:19 UTC Comment 10

External References:

https://packetstormsecurity.com/files/158755/GS20200804145053.txt
https://c-ares.haxx.se/changelog.html
https://github.com/c-ares/c-ares/commit/1cc7e83c3bdfaafbc5919c95025592d8de3a170e

Alex 2020-08-12 11:14:15 UTC Comment 11

Statement:

This flaw is rated as a having Moderate impact, because it can happen only during close of the service (or when handling ARES_ECONNREFUSED). The Red Hat Enterprise Linux not affected (all versions), because the version of c-ares being used is 1.13.0, and the bug introduced in 1.16.0 (and then fixed in 1.16.1).