14

CVE-2021-22945: UAF and double-free in MQTT sending

Share:

TIMELINE

z2_ submitted a report to curl. Jul 19th (about 1 year ago)

Vulnerability Description

libcurl version 7.77.0 has a Use-After-Free and a Double-Free in `lib/mqtt.c` in the function `mqtt_doing` on lines 556 - 563:

Code 202 Bytes Wrap lines Copy Download

```
if(mq->nsend) {
  /* send the remainder of an outgoing packet */
  char *ptr = mq->sendleftovers;
  result = mqtt_send(data, mq->sendleftovers, mq->nsend);
  free(ptr);
  if(result)
    return result;
}
```

As can be seen in the code above `mq->sendleftovers` gets freed in line 560 but not set to `NULL`. If `mqtt_doing` gets called repeatedly and the values of `mq->nsend` and `mq->sendleftovers` don't change this can result in

1. Sending the metadata of the freed chunk over the network via `mqtt_send`
2. Freeing `mq->sendleftovers` multiple times

`mq->nsend` and `mq->sendleftovers` get set in the function `mqtt_send` if `Curl_write` cannot send all bytes in the write-buffer at once. This can e.g. happen if `write()` returns `EAGAIN` or `EWOULDBLOCK`. Then `Curl_write` sets the number of written bytes to `0` and returns `CURLE_OK`.

This can trigger the vulnerabilities as follows:

1. Supply an `mqtt://` URL to curl
2. Have some successfull transmissions with `mqtt_send`
3. At some point have an unsuccessfull transmission such that not all bytes of the write-buffer can be sent. This causes `mq->sendleftovers` and `mq->nsend` to be set.
4. Have another invocation of `mqtt_doing`. The code mentioned above gets executed. `mq->sendleftovers` gets freed. If `mqtt_send` could send all remaining bytes successfully `mq->sendleftovers` and `mq->nsend` don't get reset.
5. Have another invocation of `mqtt_doing`. Since `mq->nsend` didn't change curl tries to send the leftover bytes again, triggering the vulnerabilities

How to reproduce the bug

1. Checkout tag `curl-7_77_0` in the curl repository
2. Apply the following patch that artificially creates a scenario as described above:

Code 870 Bytes Wrap lines Copy Download

```
diff --git a/lib/sendf.c b/lib/sendf.c
index e41bb805f..773d4b5b6 100644
--- a/lib/sendf.c
+++ b/lib/sendf.c
@@ -294,6 +294,7 @@ void Curl_failf(struct Curl_easy *data, const char *fmt, ...)
  * If the write would block (CURLE_AGAIN), we return CURLE_OK and
  * (*written == 0). Otherwise we return regular CURLcode value.
  */
+static int CUSTOM_blocked = 0;
 CURLcode Curl_write(struct Curl_easy *data,
                     curl_socket_t sockfd,
                     const void *mem,
@@ -322,8 +323,13 @@ CURLcode Curl_write(struct Curl_easy *data,
   }
 #endif
   bytes_written = conn->send[num](data, num, mem, len, &result);
+  if(!CUSTOM_blocked) {
+    bytes_written = 0;
+    CUSTOM_blocked = 1;
+  }

   *written = bytes_written;
+
   if(bytes_written >= 0)
     /* we completely ignore the curlcode value when subzero is not returned */
     return CURLE_OK;

```

3. Rebuild curl
4. Start a simple netcat session with: `nc -lp 5678`
5. Invoke curl with: `curl mqtt://127.0.0.1:5678/`

The output:

And in the terminal where netcat was launched it can be seen
that the content of the freed heap chunk was sent.

Impact

Since double frees of tcache chunks are not detected until glibc version 2.29
this vulnerability is perfectly exploitable for operationg systems using an older
glibc. Causing `write()` to return `EAGAIN` is more difficult but not impossible
to manage, e.g. this can always be the case if the peer is not reading as fast as
the curl client is writing (source).
At minimum this can be used to leak heap metadata which can help in exploitation.

1 attachment:
F1382088: patch

agder curl staff posted a comment. Jul 19th (about 1 year ago)

Thank you for your report!

We will take some time and investigate your reports and get back to you with details and possible follow-up questions as soon as we can!

agder curl staff posted a comment. Jul 19th (about 1 year ago)

> this vulnerability is perfectly exploitable

To me, it seems very hard for an attacker to trick an application and libcurl into performing the necessary sequence and exact steps and for the second free to actually do anything "useful". Causing a minor heap memory leak per transfer is not really an attack, just a nuisance.

2_ posted a comment. Jul 20th (about 1 year ago)

> for the second free to actually do anything "useful"

A double free of a tcache chunk does a lot of damage. If a malloc() happens that returns a double-freed chunk it is possible to manipulate the freelist and cause malloc() to return an arbitrary pointer leading to an arbitrary write. This is known as tcache poisoning.

> Causing a heap memory leak per transfer is not really an attack

This is true. However when the double free goes undetected, a leak of 8 bytes can be already enough to (partially) reconstruct the heap layout and together with the arbitrary write described above this can directly lead to RCE or to more heap metadata corruption (which also leads to RCE).

agder curl staff posted a comment. Jul 23rd (about 1 year ago)

> If a malloc() happens that returns a double-freed chunk it is possible to manipulate the freelist and

How exactly would an attacker use this flaw to manipulate the freelist?

2_ posted a comment. Jul 23rd (about 1 year ago)

> How exactly would an attacker use this flaw to manipulate the freelist?

Since a tcache is a stack realized as a singly linked list, when a chunk gets freed twice its `next` pointer points to itself.
When a call to `malloc()` returns a double freed chunk whatever gets written to the first 8 bytes of the chunk will overwrite the `next` pointer.
The second call to `malloc()` returns the chunk again (since it was freed twice) and whatever is the `next` pointer will become the top of the stack.
On a third call to `malloc()` this manipulated pointer, the top of the stack, gets returned and when an attacker can control what gets written after the third allocation one can achieve an arbitrary write.

I don't want to go into full details here, the point relevant to this discussion is that double-frees are something that have to be considered serious since an arbitrary write is "game over".

dgustafsson curl staff posted a comment. Jul 23rd (about 1 year ago)

I think a point relevant to the discussion is whether this attack is theoretical or practical, and I am too unimaginative to see how an attacker could achieve and arbitrary write here. Did you make a PoC of an attack that you can share?

agder curl staff posted a comment. Jul 30th (about 1 year ago)

> I don't want to go into full details here

But please do go into full details! We do not consider occasional semi-random crashes to be security flaws. They would need to either be possible to control to make trigger or to be somewhat plausible to happen by chance.

2_ posted a comment. Jul 31st (about 1 year ago)

First of all I want to apologize if I came across cocky with "I don't want to go into full details here".
Next time I'll choose my words more carefully.

The following PoC demonstrates how a program can be exploited if it
uses libcurl to communicate with a malicious MQTT broker.
There are four files in `PoC.tar`:

1. `Vagrantfile`: This sets up a VM where exploitation of a double free is possible: Ubuntu Bionic with glibc 2.27.
2. `client.c`: An example client that uses libcurl to communicate via MQTT and has some calls to `malloc()` and `free()` after using libcurl
3. `server.py`: This script implements a malicious MQTT broker that can trigger the double free in `client.c` and executes a tcache-poisoning attack to spawn a shell.
4. `patch`: For all this a patched version of libcurl is needed. The patch makes `Curl_write` return 0 at the right point in time such that `sendleftovers` can be set.

How to run the PoC:

1. Download all files and put them into one folder

5. Spawn the server in the background: `./server.py &`
6. Run the client: `./client`

If nothing went wrong it can be seen that a shell was spawned. It may take 2-3 times for the exploit to work.

After executing the exploit, exit the VM and execute `vagrant halt` to stop the VM.

1 attachment:
F1395265: PoC.tar

bagder curl staff changed the status to **Triaged**. Aug 9th (about 1 year ago)

(Sorry for the slowness, I was away on vacation for a while.)

Thank you @z2_ for this awesome recipe. This really did it for me and I'm definitely in the "security problem" camp now!

I would like us to work on a fix and advisory for this to get published with the next release, on September 15th unless someone thinks it needs faster treatment?

@z2_, do you have any proposed patch for this perhaps? If not I can propose one within shortly.

z2_ posted a comment. Aug 9th (about 1 year ago)

I would patch it like this:

Code 333 Bytes Wrap lines Copy Download

```
diff --git a/lib/mqtt.c b/lib/mqtt.c
index f077e6c3d..fcd40b41e 100644
--- a/lib/mqtt.c
+++ b/lib/mqtt.c
@@ -128,6 +128,10 @@ static CURLcode mqtt_send(struct Curl_easy *data,
     mq->sendleftovers = sendleftovers;
     mq->nsend = nsend;
   }
+  else {
+    mq->sendleftovers = NULL;
+    mq->nsend = 0;
+  }
   return result;
 }

```

bagder curl staff posted a comment. Aug 11th (about 1 year ago)

This flaw was introduced in commit 2522903b79 but since MQTT support was marked 'experimental' then and not enabled in the build by default until curl 7.73.0 (October 14, 2020) we count that as the first flawed version.

bagder curl staff posted a comment. Aug 11th (about 1 year ago)

Here's my first advisory draft. Also attached if you want to make a patch.

@z2_ : please let me know how you want to be credited!

UAF and double-free in MQTT sending

Project curl Security Advisory, September 15th 2021 -
Permalink

VULNERABILITY

When sending data to an MQTT server, libcurl could in some circumstances
erroneously keep a pointer to an already freed memory area and both use that
again in a subsequent call to send data and also free it *again*.

We are not aware of any exploit of this flaw.

INFO

This flaw was introduced in commit
2522903b79 but since MQTT
support was marked 'experimental' then and not enabled in the build by default
until curl 7.73.0 (October 14, 2020) we count that as the first flawed
version.

The fixed libcurl version properly clears the pointer when the data has been
sent.

The Common Vulnerabilities and Exposures (CVE) project has assigned the name
CVE-2021-XXXX to this issue.

CWE-415: Double Free

Severity: Medium

AFFECTED VERSIONS

- Affected versions: curl 7.73.0 to and including 7.78.0
- Not affected versions: curl < 7.73.0 and curl >= 7.79.0

THE SOLUTION

A pending

RECOMMENDATIONS

A - Upgrade curl to version 7.79.0

B - Apply the patch to your local version

C - Do not use MQTT

TIMELINE

This issue was reported to the curl project on July 19, 2021.

This advisory was posted on September 15, 2021.

CREDITS

This issue was reported and patched by z2_.

Thanks a lot!

1 attachment:
F1407502: CVE-2021-QQQQ.md

z2_ posted a comment. Aug 11th (about 1 year ago)
The advisory looks good to me, the credits are also fine as they are

bagder curl staff posted a comment. Aug 11th (about 1 year ago)
Excellent. I will get us a CVE for this issue in a week or two, well in advance of the publication date.

curl rewarded z2_ with a **$1,000** bounty. Aug 13th (about 1 year ago)
The curl security team has decided to reward hacker @z2_ with the amount of 1,000 USD for finding and reporting this issue. Many thanks for your great work!

bagder curl staff updated CVE reference to **CVE-2021-22945**. Aug 24th (about 1 year ago)

Aug 24th (about 1 year ago)
bagder curl staff changed the report title from **Use-After-Free and Double-Free in mqtt_doing's handling of sendleftovers** to **CVE-2021-22945: UAF and double-free in MQTT sending**.

bagder curl staff closed the report and changed the status to **Resolved**. Sep 15th (about 1 year ago)
Resolved and made public today at https://curl.se/docs/CVE-2021-22945.html

bagder curl staff requested to disclose this report. Sep 15th (about 1 year ago)

z2_ agreed to disclose this report. Sep 15th (about 1 year ago)

This report has been disclosed. Sep 15th (about 1 year ago)