3

CVE-2021-22925: TELNET stack contents disclosure again

Share:

TIMELINE

thoger submitted a report to curl. Jun 11th (2 years ago)

Summary:

CVE-2021-22898: TELNET stack contents disclosure (#1176461) issue was recently reported for curl and it was addressed in curl 7.77.0:

https://curl.se/docs/CVE-2021-22898.html
https://github.com/curl/curl/commit/39ce47f219b09c380b81f89fe54ac586c8db6bde
https://hackerone.com/reports/1176461

However, the fix applied is not correct and does not completely address the issue. It helps in cases when long environment variable name is used (`'a'*256 + ',b'`), but not when the name is short and only the value is long (`'a,' + 'b'*256` , which is the example mentioned in the curl project advisory).

Steps To Reproduce:

Follow the steps form #1176461, only use NEW_ENV option with short name and long value, such as:

Code 75 Bytes Wrap lines Copy Download

```
$ curl telnet://127.0.0.1:23 -t NEW_ENV=`python -c "print('a,' + 'b'*256)"`
```

Supporting Material/References:

When parsing NEW_ENV option value with short name and long value, sscanf() returns 2, as it writes to both `varname` and `varval` , even though the data in `varval` is truncated. Hence such variable is not skipped and is added to the `temp[]` buffer. However, the `len` counter which tracks the amount of data that was already written to `temp[]` is not updated based on the data written to the buffer in the `msnprintf()` call, but rather based on the length of the original unparsed data that is stored in `tmplen` . The relevant code is here:

https://github.com/curl/curl/blob/curl-7_77_0/lib/telnet.c#L926-L929

When value stored in `varval` is truncated, `len` is increased too much and a chunk of uninitialized memory is created in `temp[]` . The `len` should only be incremented by `strlen(varname) + strlen(varval) + 2` .

I wonder if the original fix should be preserved or re-worked. In addition to not fixing the info leak problem properly, it also causes certain valid option values to be ignored and not sent to a server any more. Rejected values are of the forms `NEW_ENV=a` or `NEW_ENV=a,` . At least the second one seems like an obviously valid way to set variable `a` to an empty string. RFC 1572 defines that environment variable can be sent with empty value and hence `NEW_ENV=a,` should remain supported. It also defines that variable can be sent with no value, making `NEW_ENV=a` a valid option as well. Note that curl prior to 7.77.0 actually did handle `NEW_ENV=a` that way, but it looks more like an unintended side effect of how `len` was incremented by `tmplen` , as the empty value part was written to `temp[]` and only subsequently overwritten. As the telnet protocol support in curl is not likely to be used widely these days, possibly only to interact with some legacy systems, it seems reasonable to prefer a fix that changes behaviour as little as possible.

Impact

Leak of an uninitialized stack memory.

Report #1176461 and the matching curl advisory provide some estimates on how much data can be leaked. I believe the amount of leaked data is smaller and is less than a half of the `temp[]` size. The reason for that is in the `check_telnet_options()` where option arguments are truncated to 255 characters, and at least half of that must part of the defined variable name or value.

https://github.com/curl/curl/blob/curl-7_77_0/lib/telnet.c#L799-L800

bagder curl staff posted a comment. Jun 11th (2 years 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!

bagder curl staff posted a comment. Jun 12th (2 years ago)

It is painful, but I agree with your conclusions. The attempt to fix this problem was not done properly. Here's an attempt to fix it now and also support a blank value:

Code 1.03 KiB Wrap lines Copy Download

```
--- a/lib/telnet.c
+++ b/lib/telnet.c
@@ -920,15 +920,16 @@ static void suboption(struct Curl_easy *data)

       for(v = tn->telnet_vars; v; v = v->next) {
         size_t tmplen = (strlen(v->data) + 1);
         /* Add the variable only if it fits */
         if(len + tmplen < (int)sizeof(temp)-6) {
-          if(sscanf(v->data, "%127[^,],%127s", varname, varval) == 2) {
-             msnprintf((char *)&temp[len], sizeof(temp) - len,
-                       "%c%s%c%s", CURL_NEW_ENV_VAR, varname,
-                       CURL_NEW_ENV_VALUE, varval);
-             len += tmplen;
+           varval[0] = 0;
+           if(sscanf(v->data, "%127[^,],%127s", varname, varval) >= 1) {
+             int store = msnprintf((char *)&temp[len], sizeof(temp) - len,
+                                   "%c%s%c%s", CURL_NEW_ENV_VAR, varname,
          }
        }
        msnprintf((char *)&temp[len], sizeof(temp) - len,
                  "%c%c", CURL_IAC, CURL_SE);

```

bagder curl staff changed the status to Triaged. Jun 12th (2 years ago)

This flaw can *still* leak stack contents, even if perhaps a little less this time...

thoger posted a comment. Jun 13th (2 years ago)

This fix looks reasonable to me, I was actually considering a very similar fix. Note that this fix slightly changes the handling of the `NEW_ENV=a` case - instead of only sending variable name with no value, it sends name plus an empty value, i.e. making it equivalent to `NEW_ENV=a,` . Considering that it's a fairly minor change, and for the protocol that should be rarely used these days, it may not be worth putting more effort into a fix that would preserve the behaviour for the special case.

thoger posted a comment. Jun 15th (2 years ago)

One idea on how to preserve different handling of `NEW_ENV=a` vs. `NEW_ENV=a,` without doing major changes to the code would be to change `sscanf` to parse the comma separator form the input data to a variable, and use its return value to check if comma was found. Something like:

Code 561 Bytes — Wrap lines Copy Download

```
          int rv;
          char sep[2] = "";

          varval[0] = 0;
          rv = sscanf(v->data, "%127[^,]%1[,]%127s", varname, sep, varval);
          if (rv == 1) {
            len += msnprintf((char *)&temp[len], sizeof(temp) - len,
                                      "%c%s", CURL_NEW_ENV_VAR, varname);
          } else if (rv >= 2) {
             len += msnprintf((char *)&temp[len], sizeof(temp) - len,
                                       "%c%s%c%s", CURL_NEW_ENV_VAR, varname,
                                       CURL_NEW_ENV_VALUE, varval);
           }
```

bagder curl staff posted a comment. Jun 17th (about 1 year ago)

Right! But we never supported that style of variable-only sending before so I don't think we need to start doing that now.

BTW, I didn't say it before but I would like us to work out a fix and advisory for this and release them in sync with the next release: curl 7.78.0 which is targeted for release on July 21, 2021.

thoger posted a comment. Jun 17th (about 1 year ago)

Sending of variable name only without any value actually worked before, even though it seems it was unintentional and therefore may be considered as never really supported. I briefly mentioned that in the initial report, but let me explain in detail.

Let's assume `NEW_ENV=a` is specified. The first line to look at is:

https://github.com/curl/curl/blob/curl-7_76_1/lib/telnet.c#L921

Code 50 Bytes — Wrap lines Copy Download

```
921         size_t tmplen = (strlen(v->data) + 1);
```

The `v->data` here is `a` , and hence `tmplen` is set to 2. Note that this line apparently assumes that the data provided contains `,` - the size of output is `1 + srtlen(varname) + 1 + strlen(varval)` . The two extra bytes beyond the length of the variable name and value are for `CURL_NEW_ENV_VAR` and `CURL_NEW_ENV_VALUE` bytes written to output. The `+ 1` in the `tmplen` calculation only accounts for one of those bytes, the other seems to be assumed to be covered by the `,` separator included in the input.

Subsequent `sscanf` initializes `varname` and leaves `varval` uninitialized / set to its previous value. The following `msnprintf` call is:

https://github.com/curl/curl/blob/curl-7_76_1/lib/telnet.c#L925-L927

Code 185 Bytes — Wrap lines Copy Download

```
925           msnprintf((char *)&temp[len], sizeof(temp) - len,
926                     "%c%s%c%s", CURL_NEW_ENV_VAR, varname,
927                     CURL_NEW_ENV_VALUE, varval);
```

This unconditionally writes `CURL_NEW_ENV_VALUE` and `varval` to `temp[]` . However, those values are overwritten in subsequent `msnprintf` calls - either when writing the next variable, or trailing `CURL_IAC` and `CURL_SE` (https://github.com/curl/curl/blob/curl-7_76_1/lib/telnet.c#L932-L933). The reason for that is this line:

https://github.com/curl/curl/blob/curl-7_76_1/lib/telnet.c#L928

Code 30 Bytes — Wrap lines Copy Download

```
928           len += tmplen;
```

As `tmplen` is 2 in our example, `len` increase only covers `CURL_NEW_ENV_VAR` and `varname` and the next write to `temp[]` starts where `CURL_NEW_ENV_VALUE` was written.

On the other hand, in the `NEW_ENV=a,` case, `tmplen` is 3, and hence `len` increase covers the `CURL_NEW_ENV_VALUE` byte as well.

Data sent over the wire is `ff fa 27 00 00 61 ff f0` vs. `ff fa 27 00 00 61 01 ff f0` , where leading `ff fa 27 00` and trailing `ff f0` are `CURL_IAC, CURL_SB, CURL_TELOPT_NEW_ENVIRON, CURL_TELQUAL_IS` and `CURL_IAC, CURL_SE` respectively.

tnoger posted a comment. Jun 17th (about 1 year ago)

As I consider this to be a Low impact issue, I have no issue with making the fix in the next planned release.

agder curl staff posted a comment. Jun 18th (about 1 year ago)

Thanks a lot for your detailed explanation. I agree that it is indeed rather interesting that the variable-only style actually worked. So let's keep that. Updated patch:

Code 1.24 KiB Wrap lines Copy Download

```
--- a/lib/telnet.c
+++ b/lib/telnet.c
@@ -920,16 +920,21 @@ static void suboption(struct Curl_easy *data)

       for(v = tn->telnet_vars; v; v = v->next) {
         size_t tmplen = (strlen(v->data) + 1);
         /* Add the variable only if it fits */
         if(len + tmplen < (int)sizeof(temp)-6) {
-          if(sscanf(v->data, "%127[^,],%127s", varname, varval) == 2) {
-            msnprintf((char *)&temp[len], sizeof(temp) - len,
-                      "%c%s%c%s", CURL_NEW_ENV_VAR, varname,
-                      CURL_NEW_ENV_VALUE, varval);
-            len += tmplen;
-          }
+          int rv;
+          char sep[2] = "";
+          varval[0] = 0;
+          rv = sscanf(v->data, "%127[^,]%1[,]%127s", varname, sep, varval);
+          if(rv == 1)
+            len += msnprintf((char *)&temp[len], sizeof(temp) - len,
+                             "%c%s", CURL_NEW_ENV_VAR, varname);
+          else if(rv >= 2)
+            len += msnprintf((char *)&temp[len], sizeof(temp) - len,
+                             "%c%s%c%s", CURL_NEW_ENV_VAR, varname,
+                             CURL_NEW_ENV_VALUE, varval);
         }
       }
       msnprintf((char *)&temp[len], sizeof(temp) - len,
                 "%c%c", CURL_IAC, CURL_SE);
       len += 2;

```

agder curl staff posted a comment. Jun 18th (about 1 year ago)

Here's my first shot at a security advisory for this flaw:

TELNET stack contents disclosure again

Project curl Security Advisory, July 21st 2021 -
Permalink

VULNERABILITY

curl supports the `-t` command line option, known as `CURLOPT_TELNETOPTIONS` in libcurl. This rarely used option is used to send variable=content pairs to TELNET servers.

Due to flaw in the option parser for sending `NEW_ENV` variables, libcurl could be made to pass on uninitialized data from a stack based buffer to the server. Therefore potentially revealing sensitive internal information to the server using a clear-text network protocol.

This could happen because curl did not call and use sscanf() correctly when parsing the string provided by the application.

The previous curl security vulnerability CVE-2021-22898 is almost identical to this one but the fix was insufficient so this security vulnerability remained.

We are not aware of any exploit of this flaw.

INFO

This flaw has existed in curl since commit a1d6ad2610 in libcurl 7.7, released on March 22, 2001. There was a previous attempt to fix this issue in curl 7.77.0 but it was not done proper.

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

CWE-457: Use of Uninitialized Variable

Severity: Medium

AFFECTED VERSIONS

Also note that libcurl is used by many applications, and not always advertised as such.

THE SOLUTION

Use sscanf() properly and only use properly filled-in buffers.

A fix for CVE-2021-22898

RECOMMENDATIONS

A - Upgrade curl to version 7.78.0

B - Apply the patch to your local version

C - Avoid using `CURLOPT_TELNETOPTIONS`

TIMELINE

This issue was reported to the curl project on June 11, 2021.

This advisory was posted on July 21, 2021.

CREDITS

This issue was reported and patched by Tomas Hoger.

Thanks a lot!

1 attachment:
F1344193: CVE-2021-TTTTT.md

bagder curl staff posted a comment. Jun 21st (about 1 year ago)
The advisory draft had an old CVE leftover in there as a copy-and-paste mistake. I've removed that in my local copy.

nyymi joined this report as a participant. Jun 21st (about 1 year ago)

nyymi posted a comment. Jun 21st (about 1 year ago)
Just to comment: I feel slightly responsible for this one as well, as I spotted the original flaw but didn't realize the fix I proposed didn't actually fix the issue. Goes to show that it's super easy to miss these things...

thoger posted a comment. Jun 24th (about 1 year ago)
Please use "Red Hat Product Security" instead of my name in the Credits section of the advisory. Thank you!

bagder curl staff posted a comment. Jun 24th (about 1 year ago)
Yes of course, updated my local copy accordingly.

bagder curl staff updated CVE reference to CVE-2021-22925. Jun 28th (about 1 year ago)

Jun 28th (about 1 year ago)
bagder curl staff changed the report title from **curl: Incorrect fix for CVE-2021-22898 - TELNET stack contents disclosure** to **CVE-2021-22925: TELNET stack contents disclosure again**.

curl rewarded thoger with a **$800** bounty. Jul 5th (about 1 year ago)
The curl security team has decided to reward hacker @thoger with the amount of 800 USD for finding and reporting this issue. Many thanks for your great work!

bagder curl staff closed the report and changed the status to **Resolved**. Jul 21st (about 1 year ago)

bagder curl staff requested to disclose this report. Jul 21st (about 1 year ago)

thoger agreed to disclose this report. Jul 21st (about 1 year ago)

This report has been disclosed. Jul 21st (about 1 year ago)