6

CVE-2022-27780: percent-encoded path separator in URL host

Share:

TIMELINE

haxatron1 submitted a report to curl. Apr 28th (7 months ago)

Summary:

URL decoding the entire proxy string could lead to SSRF filter bypasses. For example,

When the following curl specifies the proxy string `http://example.com%2F127.0.0.1`

- If curl URL parser or another RFC3986 compliant parser parses the initial string http://127.0.0.1%2F.example.com, it will derive 127.0.0.1%2Fexample.com or 127.0.0.1/example.com as the host, if for instance, an SSRF check is used to determine if a host ends with .example.com (.example.com being a allow-listed domain), the check will succeed.
- curl will then URL decode the entire proxy string to http://127.0.0.1/example.com and send it to the server

Code 134 Bytes Wrap lines Copy Download

```
GET http://127.0.0.1/example.com HTTP/1.1
Host: 127.0.0.1/example.com
User-Agent: curl/7.83.0
Accept: */*
Proxy-Connection: Keep-Alive
```

- This proxy string is valid, and proxy servers, even RFC3986-compliant ones will send the request to the host 127.0.0.1

Steps To Reproduce:

I switched things up and used 127.0.0.1 as the allow-listed server and example.com as the target server to make it easier (no need to setup a HTTP server) to reproduce.

1. I used https://github.com/abhinavsingh/proxy.py as my proxy server.
2. Perform the following:

3. You will receive a malformed response

Code 404 Bytes Wrap lines Copy Download

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
        <head>
                <title>400 - Bad Request</title>
        </head>
        <body>
                <h1>400 - Bad Request</h1>
         </body>
 </html>
```

However, this response is actually being returned by example.com, the reason is that proxy.py will forward the Host header, currently 127.0.0.1/example.com curl sends it, making it a Blind SSRF

4. If

- an attacker can control the host header either via curl itself
- the proxy does not forward the host header curl sends,
- or if servers which ignore the Host header entirely such as Express is used, it is possible to read the full response

Code 91 Bytes Wrap lines Copy Download

```
curl -x http://127.0.0.1:8899 -H "Host: example.com" http://example.com%2F127.0.0.1/%
```

Recommended Fix:

The recommended fix for this is to not URL decode the host component of the proxy string when passing to proxy server.

Impact

SSRF filter bypass at if the curl URL parser or a RFC 3986 parser is used, it could lead to blind / full SSRF depending on the proxy used.

"@" - for user-info
":" - for port
"/", "?", "#" - for dividers

bagder curl staff posted a comment. Apr 28th (7 months 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 28th (7 months ago)

It's a URL, not a "proxy string". It took me a while to understand.

Apr 28th (7 months ago)
bagder curl staff
changed the report title from **URL decoding the entire proxy string could lead to SSRF filter bypasses** to **decoding the URL before sending it to proxy could lead to SSRF filter bypasses**.

bagder curl staff posted a comment. Apr 28th (7 months ago)

commit 9a8564a920188e introduced percent-decoding host names, which I believe introduced this issue...

bagder curl staff posted a comment. Apr 28th (7 months ago)

First take at a patch that will error out if the host part decodes to one of the separators. Skipping the URL parsing completely when using HTTP proxy would be a much larger change.

Code 602 Bytes Wrap lines Copy Download

```
diff --git a/lib/urlapi.c b/lib/urlapi.c
index 99a0f6928..833512860 100644
--- a/lib/urlapi.c
+++ b/lib/urlapi.c
@@ -676,12 +676,12 @@ static CURLUcode hostname_check(struct Curl_URL *u, char *hostn
        hostname[hlen] = ']'; /* restore ending bracket */
      }
  #endif
    }
```

```
+    /* letters from the second string are not ok */
+    len = strcspn(hostname, " \r\n\t/:#?!@");
     if(hlen != len)
       /* hostname with bad content */
       return CURLUE_BAD_HOSTNAME;
   }
   if(!hostname[0])

```

haxatron1 posted a comment. Apr 29th (7 months ago)

It's a URL, not a "proxy string". It took me a while to understand.

Sorry about that 😅, I am not really familiar with terminology related to proxies.

I think that this is particularly a bigger problem in general if users use the curl URL API to parse a URL, perform a host check based on the URL result, but then use the full, returned URL from the curl URL API, so I think this problem is not limited to just proxies. The above patch of erroring out when it sees a URL separator in the host component is the correct way to go about this.

I can confirm the above patch fixes the issue:

Code 60 Bytes Wrap lines Copy Download

```
curl -x http://127.0.0.1:8899 http://example.com%2F127.0.0.1
```

will now return:

Code 53 Bytes Wrap lines Copy Download

```
curl: (3) URL using bad/illegal format or missing URL
```

bagder curl staff changed the status to ○ **Triaged**. Apr 29th (7 months ago)

Confirmed security problem.

bagder curl staff updated CVE reference to **CVE-2022-27780**. Apr 29th (7 months ago)

Apr 29th (7 months ago)

bagder curl staff

bagder curl staff posted a comment. Apr 29th (7 months ago)

Advisory draft.

1 attachment:

F1711748: CVE-2022-27780.md

bagder curl staff updated the severity from High to Medium. Apr 29th (7 months ago)

I'm "downgrading" to medium since this flaw requires rather specific circumstances for this to be practical to get abused.

haxatron1 posted a comment. Apr 29th (7 months ago)

Minor discrepancy:
Recommendations section should not have
part C and part D.

Otherwise, details looks good!

bagder curl staff posted a comment. Apr 29th (7 months ago)

Oops, copy and paste error. Fixing.

bagder curl staff posted a comment. Apr 29th (7 months ago)

The plan is right now to do a patch release (7.83.1) on May 11 with this flaw patched and announce this vulnerability in sync with that.

bagder curl staff posted a comment. May 5th (7 months ago)

I have notified distros@openwall about this issue now. Set for announcement with the pending release on May 11.

bagder curl staff closed the report and changed the status to ○ **Resolved**. May 11th (7 months ago)

Published. This issue is now eligible for a bounty claim from IBB.

bagder curl staff requested to disclose this report. May 11th (7 months ago)

haxatron1 agreed to disclose this report. May 11th (7 months ago)

Thanks for your work. The actual monetary reward part for this issue is managed by the Internet Bug Bounty so the curl project itself therefor sets the reward amount to **zero USD**. If you haven't already, please submit your reward request to them and refer back to this issue.