4 CVE-2022-27781: CERTINFO never-ending busy-loop

Share:

TIMELINE

sybr submitted a report to curl. Apr 30th (7 months ago)

Summary:

Curl is prone to a DoS attack in case the NSS TLS library is used and the CERTINFO option is enabled. Using maliciously crafted certificates on a server, an attacker can make curl run into an endless loop when connecting to the server. The bug is located in the following code segment (https://github.com/curl/curl/blob/master/lib/vtls/nss.c#L1014):

Code 368 Bytes Wrap lines Copy Download

```
/* Count certificates in chain. */
int i = 1;
now = PR_Now();
if(!cert->isRoot) {
  cert2 = CERT_FindCertIssuer(cert, now, certUsageSSLCA);
  while(cert2) {
    i++;
    if(cert2->isRoot) {
      CERT_DestroyCertificate(cert2);
       break;
    }
    cert3 = CERT_FindCertIssuer(cert2, now, certUsageSSLCA);
    CERT_DestroyCertificate(cert2);
    cert2 = cert3;
  }
}
```

When CERTINFO is set, display_conn_info() executes the above shown code, which tries to count the certificates in the chain received from servers via TLS. To this end, display_conn_info() starts with the leaf certificate and attempts to find its issuer certificate in the chain. The issuer certificate then becomes the origin for the next iteration. This step is

CA certificates that mutually list each other as issuers (see attached PoC).

Steps To Reproduce:

I have implemented a small PoC where a Webserver uses a maliciously crafted certificate chain that contains a loop. To this end, the end-entity certificate for localhost is issued by CA2, whose certificate is issued by CA1, whose certificate in turn is issued by CA2 (-> loop). The Python script for the Webserver and the certificate chain are attached to this report. To trigger the DoS in curl, the following steps need to be executed:

1. Modify URL in certinfo example (https://github.com/curl/curl/blob/master/docs/examples/certinfo.c#L46) to point to `https://localhost:4443/` instead of `https://www.example.com/` (`url_easy_setopt(curl, CURLOPT_URL, "https://localhost:4443/")`)
2. Build curl with NSS TLS library (./configure --with-nss) and with examples (make examples)
3. Execute python script attached to this report to start the attacker's Webserver
4. Execute certinfo (doc/examples/certinfo)

Supporting Material/References:

[list any additional material (e.g. screenshots, logs, etc.)]

- https_server.py (poc webserver)
- key.pem (poc webserver key)
- combined_loop.pem (poc webserver certificate chain)

Impact

An attacker who controls a server that a libcurl-using application (with NSS and enabled CERTINFO) connects to, can trigger a DoS. In this case, the application runs into an infinite loop and consumes nearly 100% CPU.

Using the CVSS calculator, I initially came up with medium severity (5.3). However, because the vulnerabilities relies on CERTINFO being enabled and NSS being used, which is not that popular and will soon be deprecated (https://curl.se/dev/deprecate.html), I eventually estimate the severity to be low.

3 attachments:

F1712953: https_server.py

F1712954: key.pem

Thank you for your report, we will investigate and get back to yo shortly.

bagder curl staff posted a comment. Apr 30th (7 months ago)

Using your recipe, I can reproduce a never-ending busy-loop. I'm inclined to agree with this being a DOS vulnerability.

bagder curl staff posted a comment. Apr 30th (7 months ago)

Initial attempt at a patch

Code 1.13 KiB Wrap lines Copy Download

```
1  diff --git a/lib/vtls/nss.c b/lib/vtls/nss.c
2  index 5b7de9f81..569c0628f 100644
3  --- a/lib/vtls/nss.c
4  +++ b/lib/vtls/nss.c
5  @@ -981,10 +981,13 @@ static void display_cert_info(struct Curl_easy *data,
6     PR_Free(subject);
7     PR_Free(issuer);
8     PR_Free(common_name);
9   }
10
11  +/* A number of certs that will never occur in a real server handshake */
12  +#define TOO_MANY_CERTS 300
13  +
14   static CURLcode display_conn_info(struct Curl_easy *data, PRFileDesc *sock)
15   {
16     CURLcode result = CURLE_OK;
17     SSLChannelInfo channel;
18     SSLCipherSuiteInfo suite;
19  @@ -1016,10 +1019,15 @@ static CURLcode display_conn_info(struct Curl_easy *data, PR
20         now = PR_Now();
21         if(!cert->isRoot) {
22           cert2 = CERT_FindCertIssuer(cert, now, certUsageSSLCA);
23           while(cert2) {
24             i++;
25  +          if(i >= TOO_MANY_CERTS) {
26  +            CERT_DestroyCertificate(cert2);
27  +            failf(data, "certificate loop");
```

```
31              CERT_DestroyCertificate(cert2);
32              break;
33            }
34            cert3 = CERT_FindCertIssuer(cert2, now, certUsageSSLCA);
35
```

sybr posted a comment. May 1st (7 months ago)

Thank you for the very quick reply!

I can confirm that the patch fixes the DoS vulnerability and I also assume that 300 is an upper bound for certificate chain lengths that will not be reached in practice.

Just a little remark: Openssl and GnuTLS actually count and output the certificate chain until the loop is reached, i.e., all 3 certificates in my PoC. If curls behavior should be consistent across all crypto libraries, either the patch or the behavior of Openssl and GnuTLS need to be adapted. However, as loops in certificate chains are undefined, there is probably no need to make curls behavior consistent for all libraries. Thus, patch is fine for me :).

bagder curl staff posted a comment. May 1st (7 months ago)

> I also assume that 300 is an upper bound for certificate chain lengths

I just picked an arbitrary large number I figured could not happen for a "real" use case. Completely unscientific.

> Openssl and GnuTLS actually count and output the certificate chain until the loop is reached

Sure, but I figured that would count more as a minor bug/improvement we can handle outside of this security issue. To keep our security patch as small as possible for the issue at hand only.

bagder curl staff changed the status to ○ **Triaged**. May 1st (7 months ago)

Confirmed. I will provide a first advisory draft within shortly.

bagder curl staff updated CVE reference to CVE-2022-27781. May 1st (7 months ago)

27781: CERTINFO never-ending busy-loop.

bagder curl staff posted a comment. May 1st (7 months ago)

@sybr How would you like me to credit you in the advisory and elsewhere?

Right now my advisory says: `This issue was reported by sybr at Hackerone. Patched by Daniel Stenberg.`

bagder curl staff posted a comment. May 1st (7 months ago)

Pending the credit question, here's my first advisory draft. This issue has been present since 7.34.0.

1 attachment:
F1713697: CVE-2022-27781.md

sybr posted a comment. May 2nd (7 months ago)

The advisory looks good to me. Please credit me the following way: "This issue was reported by Florian Kohnhäuser."

Thank you for the incredibly fast and professional handling of the issue!

bagder curl staff posted a comment. May 2nd (7 months ago)

Updated my local copy.

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)

curl has decided that this report is not eligible for a bounty. May 13th (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**.

sybr posted a comment. May 13th (7 months ago)

Thank you for patching the bug and publishing the security advisory.

I plan to submit my reward request to the Internet Bug Bounty in the coming days.

bagder curl staff disclosed this report. May 16th (6 months ago)

Disclosing this.