2

CVE-2021-22926: CURLOPT_SSLCERT mixup with Secure Transport

Share:

TIMELINE

nyymi submitted a report to curl. Jun 15th (2 ye

Summary:

libcurl Secure Transport SSL backend fails to secure the `CURLOPT_SSLCERT` against current directory file overriding the keychain nickname specified.

This leads to the possibility of locally created file overriding the `CURLOPT_SSLCERT` specified certificate and thus causing denial of service.

Steps To Reproduce:

1. Configure and build curl against Secure Transport: `configure --with-secure-transport && make`
2. Have keychain with client certificate called "testcert"
3. Use testcert from keychain to authenticate: `./src/curl -E testcert https://testsite`
4. In current directory execute `touch testcert`
5. Try authenticating again `./src/curl -E testcert https://testsite`

```
curl: (58) SSL: Can't load the certificate "testcert" and its private key: OSStatus -50
```

The issue stems from the fact that Secure Transport backend code doesn't seem to prefer the keychain over the local file. The documentation says that local file should be prefixed with "./" when used, but the code doesn't have any such checks. Interestingly NSS SSL backend does have the check: https://github.com/curl/curl/blob/master/lib/vtls/nss.c#L432

The impact of this vulnerability is rather limited: In practice it seems to be only usable in causing denial of service against applications using keychain client certific
It could happen in practice for example if executing command in /tmp directory structure or home directory of another user. The user would be able to prevent the
from creating an authenticated connection by creating a file with matching name used for the keychain nickname used by the app.

Impact

Denial of service

dgustafsson curl staff posted a comment. Jun 15th (2 ye

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 15th (2 ye

I think this behavior is actually documented in the CURLOPT_SSLCERT man page, even if not perhaps clearly enough. And perhaps its not a feature we should hav

nyymi posted a comment. Updated Jun 15th (2 ye

> I think this behavior is actually documented in the CURLOPT_SSLCERT man page

The man page does document a behaviour as NSS implement it.
Secure Transport unfortunately implements it wrong hence the problem documented here.

> And perhaps its not a feature we should have.

It probably would be better to use a separate CURLOPT for it, at least.

bagder curl staff changed the status to **Triaged**. Jun 17th (about 1 y

However unlikely this is to actually hurt anyone, there's at least a risk. The fix is probably to just require a file name to use at least one slash:

Code 379 Bytes Wrap lines Copy Dow

```
--- a/lib/vtls/sectransp.c
+++ b/lib/vtls/sectransp.c
@@ -1327,10 +1327,15 @@ CF_INLINE bool is_file(const char *filename)
   struct_stat st;

   if(!filename)
     return false;

+  /* require at least one slash in a file name */
+  n = strchr(filename, '/');
+  if(!n)
+    return false;
+
   if(stat(filename, &st) == 0)
     return S_ISREG(st.st_mode);
   return false;
 }

```

dgustafsson curl staff posted a comment. Jun 17th (about 1 y nyymi posted a comment. Jun 21st (about 1 y

Indeed the issue also extends to other way around as well, if nick names resembling paths are allowed by the "key managers".

Naturally the problem in fixing this properly (by separating the options) is that it will break some existing use cases. However, I'm not sure how often the "nicknam feature is really used. Do we have any idea how common that is? If it's super rare use case it could be changed I suppose...

agder curl staff posted a comment. Jun 21st (about 1 y

Agreed, but we can't introduce a new setopt in a security patch so we need to come up with something that at least reduces the risk down to a minimum. By insist on a slash for a file name the risk that the name would work for a local file as well as for a name should be very slim.

I can't figure out a way to patch this TOTALLY securely that doesn't at the same time break all applications that use one of the methods.

The *by-name* feature is the original functionality. The file name support was added later. I have no idea how commonly used any of them are.

agder curl staff posted a comment. Jun 21st (about 1 y

CWE-295: Improper Certificate Validation seems like the least bad CWE I can find to describe this. Any better suggestions?

nyymi posted a comment. Jun 21st (about 1 y

I tried to look for one myself -- This is a tricky one!

Technically this is not about certificate validation, but anything else is overly broad on the other hand. The choice is between CWE-295 and some business logic g CWE I guess. Either way it's not going to be fully accurate, whichever CWE is chosen.

agder curl staff posted a comment. Jun 21st (about 1 y

Advisory attempt (also attached)

CURLOPT_SSLCERT mixup with Secure Transport

Project curl Security Advisory, July 21st 2021 -
Permalink

VULNERABILITY

libcurl-using applications can ask for a specific client certificate to be used in a transfer. This is done with the `CURLOPT_SSLCERT` option (`--cert` with the command line tool).

When libcurl is built to use the macOS native TLS library Secure Transport, an application can ask for the client certificate by name or with a file name - using the same option. If the name exists as a file, it will be used instead of by name.

If the appliction runs with a current working directory that is writable by other users (like `/tmp`), a malicious user can create a file name with the same name as the app wants to use by name, and thereby trick the application to use the file based cert instead of the one referred to by name making libcurl send the wrong client certificate in the TLS connection handshake.

We are not aware of any exploit of this flaw.

INFO

This flaw has existed in curl since commit
d2fe616e7e in libcurl
7.33.0, released on October 14, 2013.

The fix for this problem does not completely remove the risk. It will now only check if the name is a file if the given name contains a slash. Make sure that used certificate names cannot double function as file names!

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

CWE-295: Improper Certificate Validation

Severity: Medium

AFFECTED VERSIONS

Using libcurl on macOS built to use Secure Transport.

- Affected versions: curl 7.33.0 to and including 7.77.0
- Not affected versions: curl < 7.33.0 and curl >= 7.78.0

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

THE SOLUTION

File names used in this option must contain at least one slash.

A fix for CVE-2021-SSSSS

RECOMMENDATIONS

A - Upgrade curl to version 7.78.0

files.

TIMELINE

This issue was reported to the curl project on June 15, 2021.

This advisory was posted on July 21, 2021.

CREDITS

This issue was reported by Harry Sintonen. Patched by Daniel Stenberg.

Thanks a lot!

1 attachment:
F1346420: CVE-2021-SSSSS.md

dgustafsson curl staff posted a comment. Jun 21st (about 1 y
Are we really solving much by requiring a slash in the name? Since the attackvector is the certificate in a file, doesn't it make more sense to enforce ordering and a check truststores (macOS keychains and NSS databases) before filesystems?

bagder curl staff posted a comment. Jun 21st (about 1 y
Ah good points. How about this?

Code 1.97 KiB Wrap lines Copy Dow

```
--- a/lib/vtls/sectransp.c
+++ b/lib/vtls/sectransp.c
@@ -1867,28 +1867,31 @@ static CURLcode sectransp_connect_step1(struct Curl_easy *data,
   if(ssl_cert || ssl_cert_blob) {
     bool is_cert_data = ssl_cert_blob != NULL;
     bool is_cert_file = (!is_cert_data) && is_file(ssl_cert);
     SecIdentityRef cert_and_key = NULL;

-    /* User wants to authenticate with a client cert. Look for it:
-       If we detect that this is a file on disk, then let's load it.
-       Otherwise, assume that the user wants to use an identity loaded
-       from the Keychain. */
-    if(is_cert_file || is_cert_data) {
+    /* User wants to authenticate with a client cert. Look for it. Assume that
+       the user wants to use an identity loaded from the Keychain. If not, try
+       it as a file on disk */
+
+    if(!is_cert_data)
+      err = CopyIdentityWithLabel(ssl_cert, &cert_and_key);
+    else
+      err = !noErr;
+    if((err != noErr) && (is_cert_file || is_cert_data)) {
       if(!SSL_SET_OPTION(cert_type))
         infof(data, "WARNING: SSL: Certificate type not set, assuming "
-                    "PKCS#12 format.\n");
+              "PKCS#12 format.\n");
       else if(strncmp(SSL_SET_OPTION(cert_type), "P12",
-        strlen(SSL_SET_OPTION(cert_type))) != 0)
+                      strlen(SSL_SET_OPTION(cert_type))) != 0)
         infof(data, "WARNING: SSL: The Security framework only supports "
-                    "loading identities that are in PKCS#12 format.\n");
+              "loading identities that are in PKCS#12 format.\n");

       err = CopyIdentityFromPKCS12File(ssl_cert, ssl_cert_blob,
-        SSL_SET_OPTION(key_passwd), &cert_and_key);
+                                        SSL_SET_OPTION(key_passwd),
+                                        &cert_and_key);
     }
-    else
-      err = CopyIdentityWithLabel(ssl_cert, &cert_and_key);

     if(err == noErr && cert_and_key) {
       SecCertificateRef cert = NULL;
       CFTypeRef certs_c[1];
       CFArrayRef certs;

```

1 attachment:
F1346451: 0001-sectransp-check-for-client-certs-by-name-first-then-.patch

dgustafsson curl staff posted a comment. Jun 21st (about 1 y
I haven't tested the patch but it seems right from reading.

bagder curl staff posted a comment. Jun 21st (about 1 y

While at it, fix the `strncmp(SSL_SET_OPTION(cert_type), "P12", strlen(SSL_SET_OPTION(cert_type)))`

It should be just `strcmp` really. Currently if you pass "", "P" or "P1" as cert_type it will also match here... That looks like a mistake to me.

However, I have no idea what the intent here was. It look like there was some idea of doing partial match somehow. Maybe it was to match anything prefixed "P12"

nyymi posted a comment. Jun 21st (about 1 y

`CURLOPT_SSLCERTTYPE` documentation only mentions "P12" however, there don't seem to be multiple "P12" prefixed variants or anything like that. So I guess `strcmp(..., "P12")` would be correct.

agder curl staff posted a comment. Jun 22nd (about 1 y

Yes, and I think it would also be prudent to return error if a type other than P12 is selected. Updated version:

Code 2.13 KiB Wrap lines Copy Dow

```
--- a/lib/vtls/sectransp.c
+++ b/lib/vtls/sectransp.c
@@ -1867,28 +1867,32 @@ static CURLcode sectransp_connect_step1(struct Curl_easy *data,
   if(ssl_cert || ssl_cert_blob) {
     bool is_cert_data = ssl_cert_blob != NULL;
     bool is_cert_file = (!is_cert_data) && is_file(ssl_cert);
     SecIdentityRef cert_and_key = NULL;

-    /* User wants to authenticate with a client cert. Look for it:
-       If we detect that this is a file on disk, then let's load it.
-       Otherwise, assume that the user wants to use an identity loaded
-       from the Keychain. */
-    if(is_cert_file || is_cert_data) {
+    /* User wants to authenticate with a client cert. Look for it. Assume that
+       the user wants to use an identity loaded from the Keychain. If not, try
+       it as a file on disk */
+
+    if(!is_cert_data)
+      err = CopyIdentityWithLabel(ssl_cert, &cert_and_key);
+    else
+      err = !noErr;
+    if((err != noErr) && (is_cert_file || is_cert_data)) {
       if(!SSL_SET_OPTION(cert_type))
-        infof(data, "WARNING: SSL: Certificate type not set, assuming "
-                    "PKCS#12 format.\n");
-      else if(strncmp(SSL_SET_OPTION(cert_type), "P12",
-        strlen(SSL_SET_OPTION(cert_type))) != 0)
-        infof(data, "WARNING: SSL: The Security framework only supports "
-                    "loading identities that are in PKCS#12 format.\n");
+        infof(data, "SSL: Certificate type not set, assuming "
+              "PKCS#12 format.\n");
+      else if(strcmp(SSL_SET_OPTION(cert_type), "P12")) {
+        failf(data, "SSL: The Security framework only supports "
+              "loading identities that are in PKCS#12 format.");
+        return CURLE_SSL_CERTPROBLEM;
+      }

       err = CopyIdentityFromPKCS12File(ssl_cert, ssl_cert_blob,
-        SSL_SET_OPTION(key_passwd), &cert_and_key);
+                                       SSL_SET_OPTION(key_passwd),
+                                       &cert_and_key);
     }
-    else
-      err = CopyIdentityWithLabel(ssl_cert, &cert_and_key);

     if(err == noErr && cert_and_key) {
       SecCertificateRef cert = NULL;
       CFTypeRef certs_c[1];
       CFArrayRef certs;

```

dgustafsson curl staff posted a comment. Updated Jun 22nd (about 1 y

The Schannel implementation checks case insensitive so I think we should be consistent across TLS backends doing that:

Code 125 Bytes Wrap lines Copy Dow

```
553     if((fInCert || blob) && (data->set.ssl.cert_type) &&
554         (!strcasecompare(data->set.ssl.cert_type, "P12"))) {
```

In OpenSSL we also use case insensitive matching.

agder curl staff posted a comment. Jun 22nd (about 1 y

Good catch, I'll update.

- NSS considers regular, fifo and char devices a file
- Secure Transport only accepts regular files

bagder curl staff posted a comment. Jun 22nd (about 1 y

I think we should unify those checks after this patch has landed and is published. Better not do it before as it'll draw attention to this particular area and risk getting issue out in the open prematurely.

dgustafsson curl staff posted a comment. Jun 22nd (about 1 y

+1. I wouldn't be surprised if there are more differences between the backends which have gone unnoticed, a project to unify as much as possible would be a good

bagder curl staff updated CVE reference to CVE-2021-22926. Jun 28th (about 1 y

bagder curl staff changed the report title from **CURLOPT_SSLCERT confusion with Secure Transport** to **CVE-2021-22926: CURLOPT_SSLCERT mixup with Secure Transport**. Jun 28th (about 1 y

curl rewarded nyymi with a **$1,000** bounty. Jul 5th (about 1 y

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!

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

bagder curl staff requested to disclose this report. Jul 21st (about 1 y

nyymi agreed to disclose this report. Jul 21st (about 1 y

This report has been disclosed. Jul 21st (about 1 y