**SUMMARY**

A concurrency bug in NSS-Libraries 3.8

**DETAILS**

Some details can also be found at: https://bugzilla.mozilla.org/show\_bug.cgi?id=225525

This bug is due to a data race.

The NSSCertificate structure has a member named "decoding", which points to an nssDecodedCert structure. There are several places in NSS that assign a value to this member (e.g. c->decoding). Typical code looks like

if (!c->decoding)

c->decoding = somefunction(some,arguments);

Such code is found in pki/certificate.c, pki/pki3hack.c, pki/pkibase.c

None of these places uses a lock or other mutual exclusion device to make the test and the update atomic.

Multiple threads can find c->decoding to be NULL, and then get pre-empted right after the test. Only one of the threads will get to do the assignment. The other threads will still leak their object when they resume and overwrite the first copy.

|  |  |
| --- | --- |
| Thread1 (certificate.c) | Thread2 (certificate.c) |
| NSS\_IMPLEMENT nssDecodedCert \*  nssCertificate\_GetDecoding (  NSSCertificate \*c  )  {  nssDecodedCert\* deco = NULL;  if (c->type == NSSCertificateType\_PKIX) {  (void)STAN\_GetCERTCertificate(c);  }  nssPKIObject\_Lock(&c->object);  if (!c->decoding) {  deco = nssDecodedCert\_Create(NULL, &c->encoding, c->type);  PORT\_Assert(!c->decoding);  c->decoding = deco;  } else {  deco = c->decoding;  } | NSS\_IMPLEMENT nssDecodedCert \*  nssCertificate\_GetDecoding (  NSSCertificate \*c  )  {  nssDecodedCert\* deco = NULL;  if (c->type == NSSCertificateType\_PKIX) {  (void)STAN\_GetCERTCertificate(c);  }  nssPKIObject\_Lock(&c->object);  if (!c->decoding) {  deco = nssDecodedCert\_Create(NULL, &c->encoding, c->type);  PORT\_Assert(!c->decoding);  c->decoding = deco;  } else {  deco = c->decoding;  } |
| 0-lock | 0-lock |