**SUMMARY**

A concurrency bug in NSS-Libraries 3.4

**DETAILS**

Some details can also be found at: https://bugzilla.mozilla.org/show\_bug.cgi?id=341323

This bug is due to a data race.

There is a race condition in that code. NSSCryptoContext\_ImportCertificate is a no-op if the cert already exists in the context's cert store. It just checks the content of the hash table, without incrementing any refcount or returning any object.

So, when we get to the next line, the NSSCryptoContext\_FindCertificateByIssuerAndSerialNumber call, it's possible for the cert to have been removed from the context by another thread.

|  |  |
| --- | --- |
| Thread1 (stanpcertdb.c) | Thread2 (stanpcertdb.c) |
| nssrv = NSSCryptoContext\_ImportCertificate(gCC, c);  if (nssrv != PR\_SUCCESS) {  goto loser;  }  /\* so find the entry in the temp store \*/  tempCert = NSSCryptoContext\_FindCertificateByIssuerAndSerialNumber(gCC,&c->issuer,&c->serial);  /\* destroy the copy \*/  NSSCertificate\_Destroy(c);  if (tempCert) {  /\* and use the "official" entry \*/  c = tempCert;  cc = STAN\_GetCERTCertificateOrRelease(c);  if (!cc) {  return NULL;  }  } else {  return NULL;  } | nssrv = NSSCryptoContext\_ImportCertificate(gCC, c);  if (nssrv != PR\_SUCCESS) {  goto loser;  }  /\* so find the entry in the temp store \*/  tempCert = NSSCryptoContext\_FindCertificateByIssuerAndSerialNumber(gCC,&c->issuer,&c->serial);  /\* destroy the copy \*/  NSSCertificate\_Destroy(c);  if (tempCert) {  /\* and use the "official" entry \*/  c = tempCert;  cc = STAN\_GetCERTCertificateOrRelease(c);  if (!cc) {  return NULL;  }  } else {  return NULL;  } |
| (cryptocontext.c)  NSS\_IMPLEMENT PRStatus  NSSCryptoContext\_ImportCertificate (  NSSCryptoContext \*cc,  NSSCertificate \*c  )  {  PRStatus nssrv;  if (!cc->certStore) {  cc->certStore = nssCertificateStore\_Create(cc->arena);  if (!cc->certStore) {  return PR\_FAILURE;  }  }  nssrv = nssCertificateStore\_Add(cc->certStore, c);  if (nssrv == PR\_SUCCESS) {  c->object.cryptoContext = cc;  }  return nssrv;  } |  |
|  |  |
| 0-lock |  |