**SUMMARY**

A concurrency bug in Core-Networking

**DETAILS**

Some details can also be found at: https://bugzilla.mozilla.org/show\_bug.cgi?id=52111

This bug is due to a data race.

The root cause seem to be a race between nsFileTransport.cpp and nsAsyncStreamListener.cpp.

nsFileTransport at several places do stuff like:

mStatus = mOutputStream->WriteFrom(mSource, mTransferAmount, &writeAmt);

if (mStatus == NS\_BASE\_STREAM\_WOULD\_BLOCK) {

mStatus = NS\_OK;

return;

}

This seemingly can run in parallel with nsOnDataAvailableEvent::HandleEvent() which calls mChannel->GetStatus(). If due to bad voodo the GetStatus() call happens at a time between the WriteFrom() call and the mStatus=NS\_OK statement the Event will be ignored and lost, leading to no initial window. This could hit on non-SMP too, but the probability is very low.

|  |  |
| --- | --- |
| Thread1 (nsFileTransport.cpp) | Thread2 (nsFileTransport.cpp) |
| nsFileTransport at several places do stuff like:  mStatus = mOutputStream->WriteFrom(mSource, mTransferAmount, &writeAmt);  if (mStatus == NS\_BASE\_STREAM\_WOULD\_BLOCK) {  mStatus = NS\_OK;  return;  } | nsFileTransport at several places do stuff like:  mStatus = mOutputStream->WriteFrom(mSource, mTransferAmount, &writeAmt);  if (mStatus == NS\_BASE\_STREAM\_WOULD\_BLOCK) {  mStatus = NS\_OK;  return;  } |
| 0-lock | 0-lock |