Java blocking on /dev/random

A Jira server I administer just hung, and a stack trace shows lots of BLOCKED threads; either like:

```
"ajp-nio-127.0.0.100-8009-exec-182" #640282 daemon prio=5 os_prio=0 cpu=0.51ms elapsed=147.51s allocated=11696B
defined_classes=0 tid=0x00007f6ce847c000 nid=0x2102f6 waiting for monitor entry  [0x00007f6c97b97000]
   java.lang.Thread.State: BLOCKED (on object monitor)
        at sun.security.provider.NativePRNG$RandomIO.implNextBytes(java.base@11.0.17/NativePRNG.java:544)
        - waiting to lock <0x0000000500fc0270> (a java.lang.Object)
        at sun.security.provider.NativePRNG.engineNextBytes(java.base@11.0.17/NativePRNG.java:220)
        at java.security.SecureRandom.nextBytes(java.base@11.0.17/SecureRandom.java:751)
        at java.util.UUID.randomUUID(java.base@11.0.17/UUID.java:150)
        at com.atlassian.jira.instrumentation.DefaultInstrumentationListenerManager.startContext
(DefaultInstrumentationListenerManager.java:149)
...
```

or:

```
"ajp-nio-127.0.0.100-8009-exec-186" #640286 daemon prio=5 os_prio=0 cpu=0.96ms elapsed=106.49s allocated=126K
defined_classes=0 tid=0x00007f6ce83e9800 nid=0x210991 waiting for monitor entry  [0x00007f6c9b524000]
   java.lang.Thread.State: BLOCKED (on object monitor)
        at sun.security.provider.NativePRNG$RandomIO.implNextBytes(java.base@11.0.17/NativePRNG.java:544)
        - waiting to lock <0x0000000500fc0270> (a java.lang.Object)
        at sun.security.provider.NativePRNG.engineNextBytes(java.base@11.0.17/NativePRNG.java:220)
        at java.security.SecureRandom.nextBytes(java.base@11.0.17/SecureRandom.java:751)
        at com.atlassian.security.random.DefaultSecureRandomService.nextBytes(DefaultSecureRandomService.java:
46)
        at com.atlassian.security.random.DefaultSecureTokenGenerator.generateToken(DefaultSecureTokenGenerator.
java:47)
        at com.atlassian.jira.security.xsrf.XsrfTokenStrategy.createToken(XsrfTokenStrategy.java:87)
...
```

These thread holding lock 'c0270 there is:

```
"DelayedLoginStore:thread-0" #573 daemon prio=5 os_prio=0 cpu=126392.47ms elapsed=1031762.88s allocated=18315M
defined_classes=19 tid=0x00007f6d2b25a800 nid=0x429cb waiting for monitor entry  [0x00007f6ca24a4000]
   java.lang.Thread.State: BLOCKED (on object monitor)
        at sun.security.provider.NativePRNG$RandomIO.implNextBytes(java.base@11.0.17/NativePRNG.java:544)
        - locked <0x0000000500fc0270> (a java.lang.Object)
        at sun.security.provider.NativePRNG.engineNextBytes(java.base@11.0.17/NativePRNG.java:220)
        at java.security.SecureRandom.nextBytes(java.base@11.0.17/SecureRandom.java:751)
        at java.util.UUID.randomUUID(java.base@11.0.17/UUID.java:150)
        at com.atlassian.jira.instrumentation.DefaultInstrumentationListenerManager.startContext
(DefaultInstrumentationListenerManager.java:149)
...
```

This is with OpenJDK version "11.0.17" 2022-10-18".

What went wrong?

This appears to be an instance of a fairly notorious problem where apps exhaust the entropy of a system, expressed as availability of bytes from /dev /random, which per the Javadocs for NativePRNG.java), is the "default seed source".

I'm not convinced though, because my threads don't contain a `FileInputStream.readBytes()` call, as would be the case if `/dev/[u]random` was really being read, as shown in this example of /dev/random blocking in Bamboo. If anyone has suggestions, please comment.

The fix

Stackoverflow to the rescue. Per the advice there for Java 11, I now set:

```
JVM_SUPPORT_RECOMMENDED_ARGS="-Djava.security.egd=file:/dev/./urandom"
```

In Jira and Confluence's `bin/setenv.sh`

If you set this (or any system-wide) flag, *document why and* when. The flag might be obsoleted in future, or even cause trouble if left when no longer needed.