**Problem**

Ratpack versions before 1.6.1 generate a session ID using a cryptographically weak PRNG in the JDK's ThreadLocalRandom. This means that if an attacker can determine a small window for the server start time and obtain a session ID value, they can theoretically determine the sequence of session IDs.

java.util.concurrent : Class ThreadLocalRandom

nextLong:

* public long nextLong()

Returns a pseudorandom long value. Overrides:

* nextLong in class Random

Returns: a pseudorandom long value

A random number generator isolated to the current thread. Like the global Random generator used by the Math class, a ThreadLocalRandom is initialized with an internally generated seed that may not otherwise be modified. When applicable, use of ThreadLocalRandom rather than shared Random objects in concurrent programs will typically encounter much less overhead and contention. Use of ThreadLocalRandom is particularly appropriate when multiple tasks (for example, each a ForkJoinTask) use random numbers in parallel in thread pools. Usages of this class should typically be of the form: ThreadLocalRandom.current().nextX(...) (where X is Int, Long, etc). When all usages are of this form, it is never possible to accidently share a ThreadLocalRandom across multiple threads.

This class also provides additional commonly used bounded random generation methods.

**Instances of** **ThreadLocalRandom are not cryptographically secure.** Consider instead using SecureRandom in security-sensitive applications. Additionally, default-constructed instances do not use a cryptographically random seed unless the system property java.util.secureRandomSeed is set to true.

**java.util.concurrent.ThreadLocalRandom**

**DefaultSessionIdGenerator.java**

public AsciiString generateSessionId() {

**ThreadLocalRandom random = ThreadLocalRandom.current();**

UUID uuid = new UUID(**random.nextLong()**, **random.nextLong()**);

return AsciiString.of(uuid.toString());

}

**Fix**

**DefaultSessionIdGenerator.java**

public AsciiString generateSessionId() {

return AsciiString.cached**(UUID.randomUUID()**.toString());

}