Use of a predictable seed in a secure random number generator (CWE-335)

Category: 2018-12-10_16-48-19_CWE > CWE-335

Description: Using a predictable seed in a pseudo-random number generator can lead to predictability of the numbers generated by it.

Using a predictable seed in a pseudo-random number generator can lead to predictability of the numbers generated by it.

Recommendation

If the predictability of the pseudo-random number generator does not matter then consider using the faster `Random` class from `java.util`. If it is important that the pseudo-random number generator produces completely unpredictable values then either let the generator securely seed itself by not specifying a seed or specify a randomly generated, unpredictable seed.

Example

In the first example shown here, a constant value is used as a seed. Depending on the implementation of `SecureRandom`, this could lead to the same random number being generated each time the code is executed.

In the second example shown here, the system time is used as a seed. Depending on the implementation of `SecureRandom`, if an attacker knows what time the code was run, they could predict the generated random number.

In the third example shown here, the random number generator is allowed to generate its own seed, which it will do in a secure way.

```
1  SecureRandom prng = new SecureRandom();
2  int randomData = 0;
3
4  // BAD: Using a constant value as a seed for a random number generator means all numbers it
generates are predictable.
5  prng.setSeed(12345L);
6  randomData = prng.next(32);
7
8  // BAD: System.currentTimeMillis() returns the system time which is predictable.
9  prng.setSeed(System.currentTimeMillis());
10 randomData = prng.next(32);
11
12 // GOOD: SecureRandom implementations seed themselves securely by default.
13 prng = new SecureRandom();
14 randomData = prng.next(32);
```

References