Entropy transfers in the Linux Random Number Generator

François Goichon, Cédric Lauradoux, Guillaume Salagnac, and Thibaut Vuillemin

INRIA, Université de Lyon,
INSA de Lyon, CITI Laboratory

Non-deterministic random number generators also known as true random number generators are a key element in security architecture to produce secrets, nonces or masking values. They are naturally found in most modern operating systems. Despite their importance and the numerous breaches they have caused in the past, this class of generators has not been intensively studied. We throw new light on a popular generator: the Linux random number generator (LRNG) [4].

Motivation – A few papers dedicated to the Linux random number generator exist. In [1, 3], the design of the LRNG is analyzed and receives several criticisms. In 2006, Gutterman *et al.* have exploited in [2] several weaknesses in the entropy sources of the LRNG to mount attacks. Consequently, the LRNG has been modified. These previous works take a cryptographic point of view to understand the LRNG. In this work, we take a system perspective: the LRNG is a mechanism that produces *"entropy"* for other applications. Our goal is to identify the applications which consumes entropy and how it is produced by the LRNG. To our knowledge, this work has never been done. We believe that it is highly valuable to know the needs in order to design a solution.

Observations – We briefly describe the achievements of our experimentation, more details will be provided during the presentation as well as a detailed description of the LRNG. We have designed a probe to observe the entropy transfers in a Linux system through the two user-level accesses (`/dev/random` and `/dev/urandom`) and a kernel-level access through the `get_random_bytes()` function. The main challenge for the probe was to be as less intrusive as possible for the monitored system. We solved this problem by using the kernel socket APIs to send an appropriate number of UDP packets.

We have considered different setups running the same operating systems (Linux Ubuntu 12.04): a desktop computer, a file server and a computing server. In all cases, we have monitored which applications consumed entropy and which events produce it. We made three notable observations. First, **the blocking device (`/dev/random`) of the LRNG is never used**. Second, **the kernel is the biggest entropy consumer in all our experiments**. Third, **the disk activity is by far the highest source of entropy** ahead of the keyboard, the mouse and other inputs.

References

1. Boaz Barak and Shai Halevi. A model and architecture for pseudo-random generation with applications to /dev/random. In *ACM conference on Computer and communications security - CCS '05*, pages 203–212, Alexandria, VA, USA, 2005. ACM.
2. Zvi Gutterman, Benny Pinkas, and Tzachy Reinman. Vulnerabilities of the Linux Random Number Generator. In *Black Hat 2006*, Las Vegas, NE, USA, 2006.
3. Patrick Lacharme, Andrea Röck, Vincent Strubel, and Marion Videau. The Linux Pseudorandom Number Generator Revisited. Cryptology ePrint Archive, Report 2012/251, 2012.
4. Matt Mackall and Theodore Ts'o. random.c – a strong random number generator, April 2012. in Linux Kernel 3.3.6.