The sys admin's daily grind: haveged

Random Release

Practical cryptography is often an encounter with many random numbers in just a few moments. Entropy is the raw material that gives birth to the random number, but it's harder to come by than you might think. *By Charly Kühnast*

Creating havoc with a computer is an easy thing to do, but today, I need a more ordered kind of chaos: entropy. I need to generate high-quality random numbers with minimal predictability. Anybody who plumbs the depths of cryptographic functions will need a good and fast entropy generator at some time, say, to make sure that keys really are generated from genuinely random data, rather than predictable numbers.

You need to be creative to achieve high-quality entropy. Creative people have invented methods for generating random numbers from the bubbles in a lava lamp, the noise generated by feedback from a microphone input, and the luminance values of a camera sensor in a closed black box. All of these work really well, but they're slow.

The same dilemma occurs on Linux systems. The `/dev/random` block device has a store of random data that the kernel generates from unpredictable hardware interrupts (keyboard, mouse, discs, etc.), which is why some key generators ask you to type garbage at the keyboard while the key is generated. The entropy quality generated by `/dev/random` is fine for most cryptographic applications, but not the volume.

You can also use `/dev/urandom`, which generates a huge volume of low-quality random data that are useless for any kind of encryption technique. To complicate matters, however, these two actually get in each other's way. If you ask `/dev/urandom` for random numbers, it first drains the entropy pool `/dev/random` to polish the results.

Processor Flutter

HAVEGE (Hardware Volatile Entropy Gathering and Expansion) [1] takes another approach. Because modern processors use many hardware mechanisms to improve branch prediction, caching, pipelines, and much more, any normal use of the CPU will trigger a barrage of state changes from thousands of these elements, and it is precisely this that HAVEGE uses to generate a large volume of quality entropy.

It comes as little surprise that the Linux implementation of the HAVEGE method goes by the name of haveged [2]. It is included with most Linux distributions, and the installation is straightforward without involving any configuration. On my Ubuntu test system, the size of the available entropy pool increased tenfold within a couple of seconds after launching haveged (Figure 1). The kernel keeps you up to date with the entropy pool fill level in `/proc/sys/kernel/random/entropy_avail`.

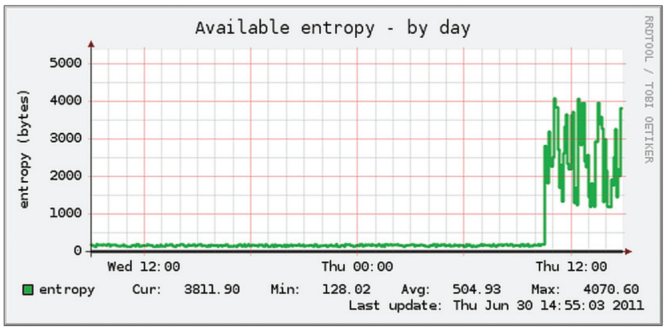

Figure 1: The entropy pool (*y* axis in bytes) available on Charly's test system increases dramatically within a couple of seconds after launching haveged.

Pure Coincidence?

If you work with cryptographic functions, even if you just need to generate a key now and then, you should install haveged. The daemon is inconspicuous and more or less maintenance-free, but it is very effective. It's definitely not just coincidence that it has become part of my standard toolbox – at least until I find a cryptographic method for mapping the chaos in my office. ■■■

INFO

[1] HAVEGE: *http://www.irisa.fr/caps/projects/hipsor/*
[2] haveged: *http://www.issihosts.com/haveged/*

AUTHOR

Charly Kühnast is a Unix operating system administrator at the Data Center in Moers, Germany. His tasks include firewall and DMZ security and availability. He divides his leisure time into hot, wet, and eastern sectors, where he enjoys cooking, freshwater aquariums, and learning Japanese, respectively.