SP 800-90B Non-Proprietary Public Use Document for Userspace CPU Time Jitter RNG Entropy Source

version 2.2.0

Document Version: 1.0

Document Date: 2023-06-13

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

Table of Contents

1 Description 3
2 Security Boundary 3
3 Operating Conditions 4
4 Configuration Settings 4
5 Physical Security Mechanisms 5
6 Conceptual Interfaces 5
7 Min-Entropy Rate 5
8 Health Tests 5
9 Maintenance 7
10 Required Testing 7

1 Description

The Userspace CPU Time Jitter RNG version 2.2.0 is a non-physical entropy source, part of the kernel binary, that feeds user space DRBGs. The noise generation of this entropy source is based on the tiny variations in the execution time of the same piece of code. The execution time of this piece of code is made unpredictable by the complexity of the different hardware components that comprise modern CPUs and the different internal states that the operating system can have at a certain point in time.

The entropy source was tested on the operational environments listed in Table 1. The noise source was tested under the assumption that its output is non-IID.

The CPU Jitter RNG runs in kernel space on the Ubuntu 22.04 64-bit operating system in the following operational environments.

Table 1: Operational environment and version

Manufacturer	Model	Operational Environment and Version	Processor
Supermicro	SYS-1019P-WTR	Ubuntu 22.04	Intel Xeon Gold 6226
Amazon Web Services (AWS)	c6g.metal	Ubuntu 22.04	AWS Graviton2
IBM	IBM z15	Ubuntu 22.04	IBM z15

2 Security Boundary

The boundary for this non-physical, software-based entropy source is the executable binary file. It is compiled from the C code that implements it (i.e., kernel source code).

Figure 1 depicts the overall design of the entropy source and its core operations.

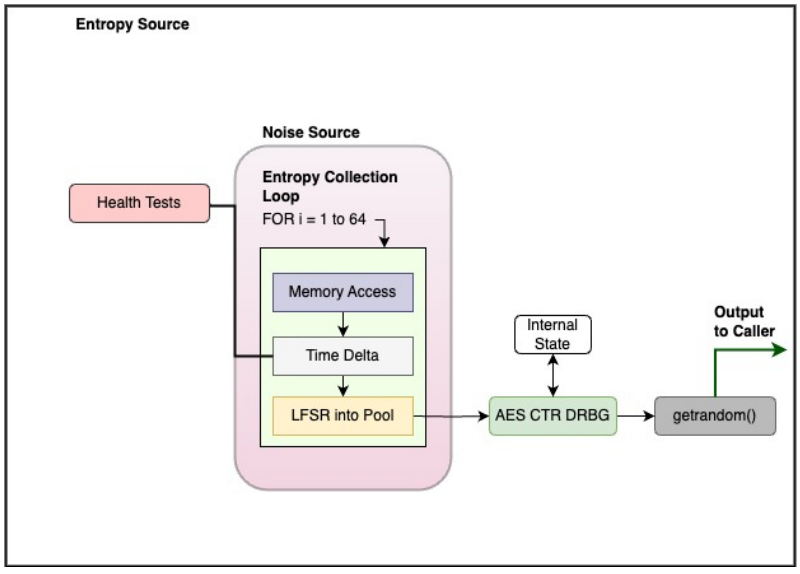

Figure 1: CPU Jitter 2.2.0 Design

The noise source is implemented by collecting and accumulating variances of the execution time of a defined set of instructions. The accumulation is done with the assistance of a Linear-Feedback Shift Register (LFSR) function that works as the conditioning component of the entropy source, as well as the entropy pool that is used to accumulate the acquired entropy and eventually the NIST SP 800-90Ar1-compliant AES-256 CTR DRBG that acts as the second conditioning component (vetted). The time jitter of the execution time is measured over the processing logic of the LFSR and functions that support the implementation.

If the Repetition Count Test (RCT) or the Adaptive Proportional Test (APT) health tests fail, the noise data is discarded, the entropy source halts without outputting any data, and a failure code is returned to the caller. If the health test failure is permanent, the kernel which contains this entropy source will panic.

3 Operating Conditions

The noise source is non-physical, and thus the operating conditions are inherited from the operational environment in which the entropy source is installed.

Manufacturer / Model	Temperature	Voltage	Humidity	Clock Speed	Cache Sizes
Supermicro SYS-1019P-WTR	10C° – 35C°	100-240Vac 5 or 12V	8% - 90 %	3.7 GHz	L1d: 12x32 KiB L1i: 12x32 KiB L2: 12x1 MiB L3: 19.25 MiB
AWS c6g.metal	0C° – 45C°	12V	60 %	2.5 GHz	L1d: 64x64 KiB L1i: 64x64 KiB L2: 64x1 MiB L3: 32 MiB
IBM z/15	5C° – 40C°	200-240Vac or 380-415Vac	N/A	5.2 GHz	L1d: 10x128 KiB L1i: 10x128 KiB L2d: 10x4 MiB L2i: 10x4 MiB L3: 256 MiB

4 Configuration Settings

The caller shall use the getrandom system call with only the GRND_RESEED flag set. When this flag is set, the DRBG conditioning component complies with the requirements described in FIPS 140-3 IG D.K, Resolution 5. If such flag is not set, the entropy source cannot be used.

One thing to note is that, after the getrandom system call, at most 256 bits of data are returned to the caller. If the caller requires more entropy output, the caller must use the getrandom syscall, multiple times. This ensures the DRBG conditioning component is properly reseeded.

5 Physical Security Mechanisms

The noise source is non-physical. The physical security mechanisms only apply to the hardware component of the operational environment in which the entropy source is installed, and thus the entropy source inherits those mechanisms.

6 Conceptual Interfaces

The entropy source provides the following interfaces:

- getrandom() with only the GRND_RESEED flag set: Obtains conditioned entropy for the caller. This is the main function of the entropy source, the one that shall be used to request entropy data. This interface corresponds to the GetEntropy() conceptual interface from SP 800-90B.
- jent_lfsr_time(): Obtains raw noise data for testing purposes. This interface corresponds to the GetNoise() conceptual interface from SP800-90B.

Note: /dev/random and /dev/urandom cannot be used to access the Entropy Source described in this document.

7 Min-Entropy Rate

The noise source provides an entropy rate for each time delta of $H_{submitter}$ = 1 bit/bit.

The entropy source collects 64 time deltas of 64 bits each (4096 bits) as input to the LFSR conditioning component. This corresponds to 64 bits of entropy. The output entropy rate of the LFSR is assessed to be 0.928683 bits/bit.

Then, five 64-bit output blocks of the LFSR are used to seed the AES-256 CTR DRBG conditioning component, which corresponds to 297 bits of entropy. The DRBG conditioning component outputs a 256 bit sample, which is assessed to contain 256 bits of entropy.

As a result, the entropy source provides 256 bits/bit of entropy rate at its output.

8 Health Tests

The entropy source implements the following continuous health tests:

- Repetition Count Test conforming to SP 800-90B section 4.4.1.
  - $H = 1$ bit of entropy per 64-bit sample.
  - Intermittent failure alpha value $\alpha_i = 2^{-30}$.
  - Permanent failure alpha value $\alpha_p = 2^{-60}$.
  - Intermittent failure cutoff value $C_i = 31$
  - Permanent failure cutoff value $C_p = 61$
- Adaptive Proportion test conforming to SP 800-90B section 4.4.2.
  - $W = 512$
  - $H = 1$ bit of entropy per 64-bit sample
  - Intermittent failure alpha value $\alpha_i = 2^{-30}$
  - Permanent failure alpha value $\alpha_p = 2^{-60}$
  - Intermittent failure cutoff value $C_i = 325$

    - Permanent failure cutoff value $C_p = 355$

- Stuck (Non-Permanent) Test: The stuck test computes the first, second and third discrete derivatives of the time value that will be processed by the LFSR. If any of these derivatives are zero, then the received time delta is considered stuck. In this case the LFSR is still updated, but the entropy value is not counted and the output of the LFSR is not used for insertion into the entropy pool. The stuck test then triggers the RCT for further processing. The second derivative is in fact the RCT itself.

As allowed by Section 4.3 of SP 800-90B, the entropy source defines two types of health test failures for the RCT and the APT: intermittent failures and permanent failures.

An intermittent failure is characterized by a false positive probability $\alpha_i = 2^{-30}$, which lies within the recommended range of $2^{-20} \leq \alpha \leq 2^{-40}$. When an intermittent failure is detected, the CPU Jitter RNG is automatically reset (which includes clearing the entropy pool and resetting the DRBG conditioning component), and the caller is notified of this failure. The only exception to this rule is during the startup tests, where intermittent failures will be treated as permanent.

Permanent failures are characterized by a false positive probability of $\alpha_p = 2^{-60}$. When a permanent failure is detected, the CPU Jitter RNG is also reset, but the Linux kernel that contains this entropy source immediately enters the error state. In practice, this results in a kernel panic.

The stuck test is considered non-permanent, as positive stuck tests will be registered but will not immediately halt the entropy source.

Startup tests conduct the same set and parameters of the continuous health tests on 1024 samples of noise data. The data is discarded after the startup tests complete successfully. Any health test failure during the startup tests will always be treated as a permanent failure, which results in the permanent shutdown of the entropy source.

On-demand health tests of the noise source may be performed by rebooting the operational environment, which results in the immediate execution of the start-up tests. Typically, this entropy source designed for kernel space cannot be reloaded without rebooting the entire kernel. Similarly, the data used for the on-demand health tests are discarded after successful completion.

When entropy is requested from user space modules through the getrandom() syscall, if the requested entropy is not available, the EAGAIN error is returned and, provided that GRND_NONBLOCK flag was not set, the user space module hangs.

9 Maintenance

There are no maintenance requirements as the entropy source is software-based.

10 Required Testing

To test the entropy source, raw data samples must be collected using a test harness that is capable of accessing the jent_lfsr_time() noise interface from the entropy source. The test harness and accessory kernel tools must be supplied by the vendor.

Raw noise data consisting of at least 1,000,000 64-bit samples truncated to 8 bits must be collected from the operational environment at its normal operating conditions and processed by the SP 800-90B entropy tool that is provided by NIST. The expected min-entropy rate must approach the one in Section 7.

Restart data must be collected at normal operating conditions through the jent_lfsr_time() interface following the restart procedure specified in SP 800-90B (i.e., 1,000 samples from 1,000 restarts each) and processed by the NIST SP 800-90B entropy tool. The minimum of the row-wise and column-wise entropy rate must be more than half that of the raw noise entropy rate.

If conditioned entropy data is to be tested, then the getrandom() interface shall be accessed instead of jent_lfsr_time() interface.