ORACLE
Linux

Oracle Userspace CPU Time Jitter RNG Entropy Source version 2.2.0

SP 800-90B Non-Proprietary Public Use Document

Document Version: 1.1

Document Date: 2024-03-12

Prepared for:

Oracle Corporation
World Headquarters
2300 Oracle Way
Austin, TX 78741
U.S.A.

Prepared by:

atsec information security Corp.
4516 Seton Center Parkway, Suite 250
Austin, TX 78759
USA

ORACLE
Linux

Title: Oracle Userspace CPU Time Jitter RNG Entropy Source

Date: December 17th, 2024

Author: atsec information security

Contributing Authors:

Oracle Security Evaluations – Global Product Security

Oracle Corporation

World Headquarters

2300 Oracle Way

Austin, TX 78741

U.S.A.

Worldwide Inquiries:

Phone: +1.650.506.7000

Fax: +1.650.506.7200

www.oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together

ORACLE
Linux

Table of Contents

ORACLE
Linux

1 Description

The Oracle Userspace CPU Time Jitter RNG version 2.2.0 is a non-physical entropy source, part of the kernel binary that feeds the DRBG in the userspace crypto libraries. 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 using both possible timers. The noise source was tested under the assumption that its output is non-IID.

Table 1: Operational environment and version.

Model	Operational Environment and Version	Processor
ORACLE SERVER X9-2c	Oracle Linux 9 on KVM on Oracle Linux 8	Intel® Xeon® Platinum 8358
ORACLE SERVER E4-2c		AMD EPYC™ 7J13
ORACLE SERVER A1-2c		Ampere® Altra® Q80-30
Marvell Liquid IO II	Oracle Linux 9	Marvell OCTEON III

2 Security Boundary

The boundary for this non-physical, software-based entropy source is the executable code compiled from the C code that implements it. The noise source, LFSR, and HMAC-SHA-512 DRBG conditioning component are implemented as part of the kernel executable.

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

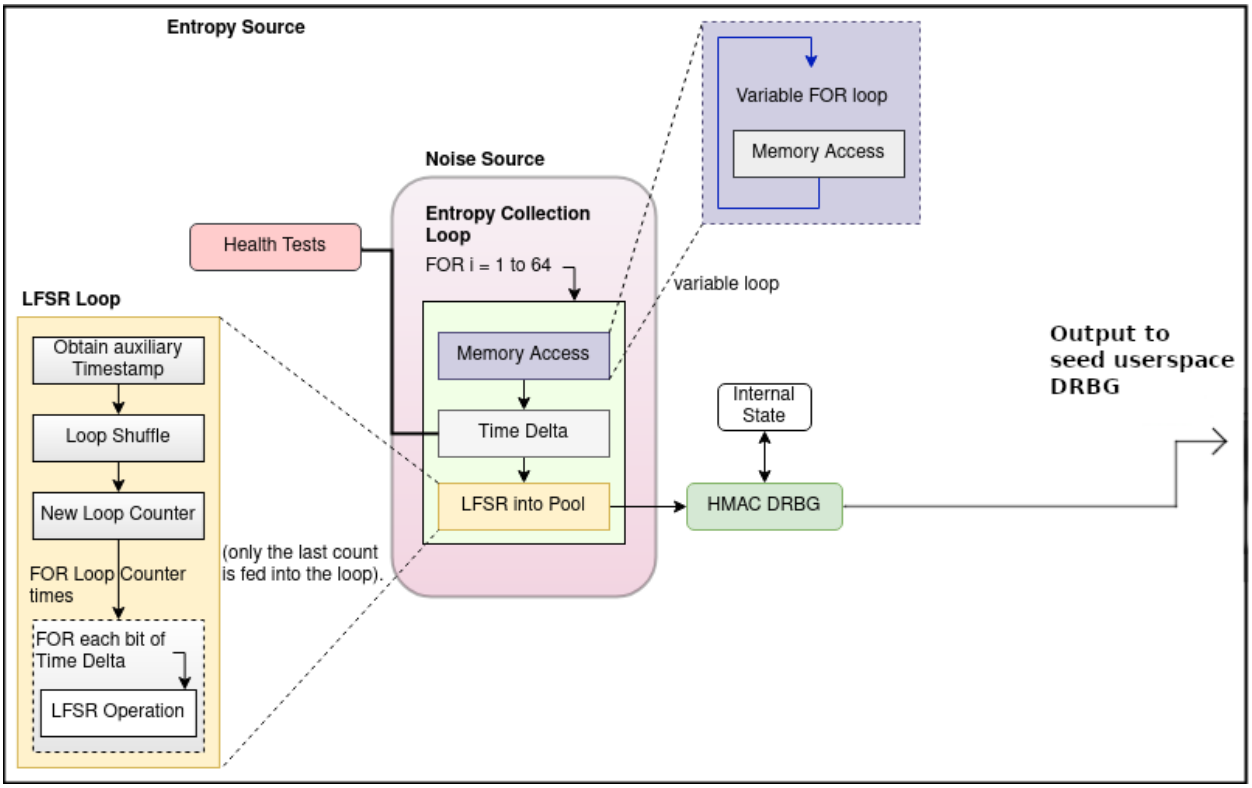

Figure 1: CPU Jitter 2.2.0 Design

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 first conditioning component of the entropy source, as well as an entropy pool that is used to accumulate the acquired entropy, then the NIST SP 800-90Ar1-compliant HMAC-SHA-512 DRBG (the second conditioning component). The time jitter of the execution time is measured over the processing logic of the LFSR and functions that support the implementation.

The noise source, the LFSR, and the HMAC-SHA-512 DRBG conditioning components are implemented as part of the kernel.

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, as shown in Table 2 below.

Table 2: Operating Conditions for each Operational Environment

Processor	**Temperature**	**Voltage**	**Clock Speed**	**Cache Sizes**
Intel® Xeon® Platinum	65C	0V - 2.15V	2.6 GHz	L1: 32x48 KB

Processor	Temperature	Voltage	Clock Speed	Cache Sizes
8358	Junction temperature			L2: 32x1280 KB L3: 49.15 MB
AMD EPYC™ 7J13	0C to 100C Processor temperature control value	0V - 3.3V	2.55 GHz	L1: 64x32 KB L2: 64x512 KB L3: 32.77 MB
Ampere® Altra® Q80-30	0C to 100C Junction temperature	0V - 1.1V	3.0 GHz	L1: 80x64 KB L2: 80x1024 KB
Marvell OCTEON III	0C to 105C Junction temperature	0.82V – 0.88V	1.5 GHz	L1: 32x16 KB L2: 4096 KB

4 Configuration Settings

The caller shall use the getrandom system call with the GRND_RANDOM 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.

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(): Obtains conditioned entropy. 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 SP 800-90B.

7 Min-Entropy Rate

The noise source provides an entropy rate for each time delta of $H_{submitter}$ = 1 bit of entropy per 64-bit sample.

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 0.926188 bits/bit.

Then, five 64-bit output blocks of the LFSR are used to seed the HMAC-SHA-512 DRBG conditioning component, which corresponds to 296 bits of entropy. This DRBG conditioning component outputs a 256 bit block, which is assessed to contain 256 bits of entropy.

As a result, the entropy source provides 1 bit/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 continuous-health tests are applied to each new sample obtained from the noise source.

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 have completed 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 cannot be reloaded without restarting the executable. Similarly, the data used for the on-demand health tests are discarded after successful completion.

9 Maintenance

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

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 samples consisting of at least 1,000,000 bytes 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.

11 Vendor Permissions and Relationship

The ESV certificate is “Reuse restricted to vendor”. Someone other than the vendor can only use the certificate with written and signed permission from the vendor’s point of contact (as indicated on the ESV certificate).