NAME

haveged - Generate random numbers and feed linux random device.

SYNOPSIS

haveged [options]

DESCRIPTION

haveged generates an unpredictable stream of random numbers harvested from the indirect effects of hardware events on hidden processor state (caches, branch predictors, memory translation tables, etc) using the HAVEGE (HArdware Volatile Entropy Gathering and Expansion) algorithm. The algorithm operates in user space, no special privilege is required for file system access to the output stream.

Linux pools randomness for distribution by the /dev/random and /dev/urandom device interfaces. The standard mechanisms of filling the /dev/random pool may not be sufficient to meet demand on systems with high needs or limited user interaction. In those circumstances, **haveged** may be run as a privileged daemon to fill the /dev/random pool whenever the supply of random bits in /dev/random falls below the low water mark of the device.

haveged tunes itself to its environment and provides the same built-in test suite for the output stream as used on certified hardware security devices. See **NOTES** below for further information.

OPTIONS

-b nnn, --buffer=nnn
: Set collection buffer size to nnn KW. Default is 128KW (or 512KB).

-d nnn, --data=nnn
: Set data cache size to nnn KB. Default is 16 or as determined dynamically.

-f file, --file=file
: Set output file path for non-daemon use. Default is sample, use - for stdout.

-F , --Foreground
: Run daemon in foreground. Do not fork and detach.

-i nnn, --inst=nnn
: Set instruction cache size to nnn KB. Default is 16 or as determined dynamically.

-n nnn, --number=nnn
: Set number of bytes written to the output file. The value may be specified using one of the suffixes k, m, g, or t. The upper bound of this value is 16t (2^44 Bytes = 16TB). A value of 0 indicates unbounded output and forces output to stdout. This argument is required if the daemon interface is not present. If the daemon interface is present, this setting takes precedence over any --run value.

-o <spec>, --onlinetest=<spec>
: Specify online tests to run. The <spec> consists of optional tot and continuous groups, each group indicates the procedures to be run, using a<n> to indicate a AIS-31 procedure A variant, and b to indicate AIS procedure B. The specifications are order independent (procedure B always runs first in each group) and case insensitive. The a<n> variations exist to mitigate the a slow autocorrelation test (test5). Normally all procedure A tests, except the first are iterated 257 times. An a<n> option indicates test5 should only be executed every modulo <n> times during the procedure's 257 repetitions. The effect is so noticable that A8 is the usual choice.

The tot tests run only at initialization - there are no negative performance consequences except for a slight increase in the time required to initialize. The tot tests guarantee haveged has initialized properly. The use of both test procedures in the tot test is highly recommended because the two test emphasize different aspects of RNG quality.

In continuous testing, the test sequence is cycled repeatedly. For example, the string tbca8b (suitable for an AIS NTG.1 device) would run procedure B for the tot test, then cycle between procedure A8 and procedure B continuously for all further output. Continuous testing does not come for free, impacting both throughput and resource consumption. Continual testing also opens up the possibility of a test failure. A strict retry procedure recovers from spurious failure in all but the most extreme circumstances. When the retry fails, operation will terminate unless a w has been appended to the test token to make the test advisory only. In our example above, the string tbca8wbw would make all continuous tests advisory. For more detailed information on AIS retries see **NOTES** below.

Complete control over the test configuration is provided for flexibility. The defaults (ta8bcb if run as a daemon and ta8b otherwise) are suitable for most circumstances.

-p file, --pidfile=file
: Set file path for the daemon pid file. Default is /var/run/haveged.pid,

-r n, --run=n
: Set run level for daemon interface:

  n = 0 Run as daemon - must be root. Fills /dev/random when the supply of random bits falls below the low water mark of the device.

  n = 1 Display configuration info and terminate.

  n > 1 Write <n> kb of output. Deprecated (use --number instead), only provided for backward compatibility.

  If --number is specified, values other than 0,1 are ignored. Default is 0.

-v n, --verbose=n
: Set diagnostic bitmap as sum of following options:

  1=Show build/tuning summary on termination, summary for online test retries.

  2=Show online test retry details

  4=Show timing for collections

  8=Show collection loop layout

  16=Show collection loop code offsets

  32=Show all online test completion detail

  Default is 0. Use -1 for all diagnostics.

-w nnn, --write=nnn
: Set write_wakeup_threshold of daemon interface to nnn bits. Applies only to run level 0.

-?, --help
: This summary of program options.

NOTES

haveged tunes the HAVEGE algorithm for maximum effectiveness using a hierarchy of defaults, command line options, virtual file system information, and cpuid information where available. Under most circumstances, user input is not required for excellent results.

Run-time testing provides assurance of correct haveged operation. The run-time test suite is modeled upon the AIS-31 specification of the German Common Criteria body, BIS. This specification is typically applied to hardware devices, requiring formal certification and mandated start-up and continuous operational testing. Because haveged runs on many different hardware platforms, certification cannot be a goal, but the AIS-31 test suite provides the means to assess haveged output with the same operational tests applied to certified hardware devices.

AIS test procedure A performs 6 tests to check for statistically inconspicuous behavior. AIS test procedure B performs more theoretical tests such as checking multi-step transition probabilities and making an empirical entropy estimate. Procedure A is the much more resource and compute intensive of the two but is still recommended for the haveged start-up tests. Procedure B is well suited to use of haveged as a daemon because the test entropy estimate confirms the entropy estimate haveged uses when adding entropy to the /dev/random device.

No test is perfect. There is a 10e-4 probability that a perfect generator will fail either of the test procedures. AIS-31 mandates a strict retry policy to filter out false alarms and haveged always logs test procedure failures. Retries are expected but rarely observed except when large data sets are generated with continuous testing. See the **libhavege(3)** notes for more detailed information.

FILES

If running as a daemon, access to the following files is required

/dev/random

/proc/sys/kernel/osrelease

/proc/sys/kernel/random/poolsize

/proc/sys/kernel/random/write_wakeup_threshold

DIAGNOSTICS

Haveged returns 0 for success and non-zero for failure. The failure return code is 1 general failure unless execution is terminated by signal <n>, in which case the return code will be 128 + <n>. The following diagnostics are issued to stderr upon non-zero termination:

Cannot fork into the background
: Call to daemon(3) failed.

Cannot open file <s> for writing.
: Could not open sample file <s> for writing.

Cannot write data in file:
: Could not write data to the sample file.

Couldn't get pool size.
: Unable to read /proc/sys/kernel/random/poolsize

Couldn't initialize HAVEGE rng
: Invalid data or instruction cache size.

Couldn't open PID file <s> for writing
: Unable to write daemon PID

Couldn't open random device
: Could not open /dev/random for read-write.

Couldn't query entropy-level from kernel: error
: Call to ioctl(2) failed.

Couldn't open PID file <path> for writing
: Error writing /var/run/haveged.pid

Fail:set_watermark()
: Unable to write to /proc/sys/kernel/random/write_wakeup_threshold

RNDADDENTROPY failed!
: Call to ioctl(2) to add entropy failed

RNG failed
: The random number generator failed self-test or encountered a fatal error.

Select error

Call to select(2) failed.

Stopping due to signal <n>
Signal <n> caught.

Unable to setup online tests
Memory unavailable for online test resources.

EXAMPLES

Write 1.5MB of random data to the file /tmp/random
```
haveged -n 1.5M -f /tmp/random
```

Generate a /tmp/keyfile for disk encryption with LUKS
```
haveged -n 2048 -f /tmp/keyfile
```

Overwrite partition /dev/sda1 with random data. Be careful, all data on the partition will be lost!
```
haveged -n 0 | dd of=/dev/sda1
```

Generate random ASCII passwords of the length 16 characters
```
(haveged -n 1000 -f - 2>/dev/null | tr -cd '[:graph:]' | fold -w 16 && echo ) | head
```

Write endless stream of random bytes to the pipe. Utility pv measures the speed by which data are written to the pipe.
```
haveged -n 0 | pv > /dev/null
```

Evaluate speed of haveged to generate 1GB of random data
```
haveged -n 1g -f - | dd of=/dev/null
```

Create a random key file containing 65 random keys for the encryption program aespipe.
```
haveged -n 3705 -f - 2>/dev/null | uuencode -m - | head -n 66 | tail -n 65
```

Test the randomness of the generated data with dieharder test suite
```
haveged -n 0 | dieharder -g 200 -a
```

Generate 16k of data, testing with procedure A and B with detailed test results. No c result seen because a single buffer fill did not contain enough data to complete the test.
```
haveged -n 16k -o tba8ca8 -v 33
```

Generate 16k of data as above with larger buffer. The c test now completes - enough data now generated to complete the test.
```
haveged -n 16k -o tba8ca8 -v 33 -b 512
```

Generate 16m of data as above, observe many c test completions with default buffer size.
```
haveged -n 16m -o tba8ca8 -v 33
```

Generate large amounts of data - in this case 16TB. Enable initialization test but made continuous tests advisory only to avoid a possible situation that program will terminate because of procedureB failing two times in a row. The probability of procedureB to fail two times in a row can be estimated as <TB to generate>/3000 which yields 0.5% for 16TB.
```
haveged -n 16T -o tba8cbw -f - | pv > /dev/null
```

Generate large amounts of data (16TB). Disable continuous tests for the maximum throughput but run the online tests at the startup to make sure that generator for properly initialized:
```
haveged -n 16T -o tba8c -f - | pv > /dev/null
```

SEE ALSO

libhavege(3),
cryptsetup(8), aespipe(1), pv(1), openssl(1), **uuencode(1)**

REFERENCES

HArdware Volatile Entropy Gathering and Expansion: generating unpredictable random numbers at user level by A. Seznec, N. Sendrier, INRIA Research Report, RR-4592, October 2002

A proposal for: Functionality classes for random number generators by W. Killmann and W. Schindler, version 2.0, Bundesamt fur Sicherheit in der Informationstechnik (BSI), September, 2011

A Statistical Test Suite for the Validation of Random NUmber Generators and Pseudorandom Number Generators for Cryptographic Applications, special publication SP800-22, National Institute of Standards and Technology, revised April, 2010

Additional information can also be found at http://www.issihosts.com/haveged/

AUTHORS

Gary Wuertz <gary@issiweb.com> and Jirka Hladky <hladky jiri AT gmail DOT com>