haveged

Génération de nombres aléatoires

haveged génère un flux imprédictible de nombres aléatoires depuis les effets indirects des événements hardware sur l'état caché (caches, prédicteurs de branche, tables de traduction mémoire, etc) en utilisant l'argorithme HAVEGE (HArdware Volatile Entropy Gathering and Expansion). L'algorithme opère en userspace, sans privilèges spéciaux.

Linux utilise les interfaces /dev/random et /dev/urandom pour fournir des nombres aléatoire. Les mécanismes standard pour remplir /dev/random peuvent ne pas être suffisants pour répondre à la demande du système. Dans ces circonstances haveged peut être lancé comme services privilégié pour remplir /dev/random.

OPTIONS

-b nnn, –buffer=nnn Définis la taille du tampon en KW. (défaut : 128KW = 512Kio)

-d nnn, –data=nnn Définis la taille du cache de données en Ko. Défaut : 16 ou déterminé dynamiquement

-f file, –file=file Définis le fichier une utilisation non-service. Défaut : "sample"

-F, –foreground Ne lance pas en tâche de fond

-i nnn, –inst=nnn Taille du cache d'instruction en Ko. Défaut : 16 ou déterminé dynamiquement

nnn, –number=nnn Nombre d'octets écris dans le fichier de sortie.

-o <spec>, –onlinetest=<spec> Spécifie les tests online à lancer <spec> consiste de groupes optionnels "t"ot et "c"ontinuous, chaque groupe indiquant les procédures à lancer, utilisant "a<n>" pour indiques une procédure AIS-31 variante A, et "b" pour indiques la procédure B. Les spécification sont indépendante de l'ordre. Les tests "tot" sont lancés seulement à l'initialisation. En test continue, la séquence de test est cyclique.

-p file, –pidfile=file Définis le chemin du fichier pid. Défaut : /var/run/haveged.pid

-r n, –run=n Définis le niveau d'exécution pour le service. 0=lance en service, doit être root. 1=Affiche la configuration et se termine. >1, écris N Kio sur la sortie. déprécié, utiliser –number

-v n, –verbose=n Diagnostique. -1 signifie tous les diagnostiques (défaut0) :

- **1** Affiche le sommaire build/tunning en se terminant
- **2** Affiche les détails de retentatives de test online
- **4** Affice le timing pour les collection
- **8** Affiche a couche de boucle de collection
- **16** Affiche les offsets de code de boucle de collection
- **32** Affiche tous les détails de tests online terminés

-w nnn, –write=nnn Définis le write_wakeup_threshold de l'interface à nnn bits. uniquement en run level 0.

NOTES

haveged définis l'algorithme HAVEGE pour une efficacité maximum en utilisant un hiérarchie de défauts, options de ligne de commande, informations de système de fichier virtuel, et informations cpuid si disponible. Sous certaines circonstances, l'entrée utilisateur n'est pas requise pour d'excellents résultats.

Les tests run-time fournissent l'assurance des opérations de haveged. La suite de test est modélisée sur la spécification AIS-31 du German Common Criteria, BIS. Cette spécification est typiquement appliquée aux périphériques aléatoires, nécessitant une certification formelle.

Parce qu'haveged tourne sur différentes plateformes hardware, la certification ne peut pas être un but, mais la suite de test AIS-31 fournis un moyen de valider la sortie de haveged avec les même tests opérationnels appliqués aux périphériques hardwares certifiés.

Fichiers

Si haveged est lancé en tant que service, les fichiers suivants sont requis :

/dev/random

/proc/sys/kernel/osrelease

/proc/sys/kernel/random/poolsize

/proc/sys/kernel/random/write_wakeup_threshold

Exemples

Écrire 1,5Mo de données aléatoirs dans /tmp/random
haveged -n 1.5M -f /tmp/random
Générer un /tmp/keyfile pour le chiffrement de disque avec LUKS
haveged -n 2048 -f /tmp/keyfile
Écraser la partition /dev/sda1 avec des données aléatoires
haveged -n 0 | dd of=/dev/sda1
Générer des mots de passe ASCII aléatoires de 16 caractères
(haveged -n 1000 -f - 2>/dev/null | tr -cd '[:graph :]' | fold -w 16 && echo) | head
Écrire un flux d'octets dans un pipe. pv mesure la vitesse d'écriture dans le pipe
haveged -n 0 | pv > /dev/null
Évaluer la vitesse de génération de 1Go de données aléatoires
haveged -n 1g -f - | dd of=/dev/null
Créer un fichier clé aléatoire contenant 65 clé aléatoire pour le programme de chiffrement aespipe
haveged -n 3705 -f - 2>/dev/null | uuencode -m - | head -n 66 | tail -n 65
Tester la qualité des données générée avec la suite de test dieharder
haveged -n 0 | dieharder -g 200 -a
Générer 16k de données, tester avec la procédure A et B avec résultats détaillés.
haveged -n 16k -o tba8ca8 -v 33
Génèrer également 16k de données avec un buffer plus grand. Le test c peut être complété
haveged -n 16k -o tba8ca8 -v 33 -b 512