Whitewood netRandom Client
Installation and Configuration Guide

Table of Contents

Introduction

The netRandom Client requests entropy in the form of true random numbers from the netRandom Free service at https://www.getnetrandom.com. netRandom provides entropy directly into the Operating System's entropy pool. On Linux, the client mixes in streamed entropy into the main entropy pool which is then automatically used to re-seed the Pseudo Random Number Generators (PRNGs) that are present within Linux and that output through the well know device files /dev/random and /dev/urandom/. On Windows, a user-space module receives the raw streamed entropy, which it then sends to the Kernel boot driver. The netRandom sourced entropy is additive to the existing local sources of entropy in Windows or Linux such as disk interrupts, systems interrupts, keyboard and mouse activity etc.

Document number: NRF10006

Why do I need more Entropy?

The primary purpose of using the netRandom Client as a supplementary source of entropy is to ensure that the operating system always has access to a reliable source of high-quality entropy while not being dependent on the trustworthiness of any single source of entropy. It is important to take this preventative step of providing supplementary entropy because there is no reliable method to otherwise track the availability and quality of exiting entropy sources in an operating system and there are many situations where the existing supply of entropy is inadequate.

The threat of 'entropy starvation' can become a critical issue in situations where there is minimal user level activity (for example in the case of data center servers), where system level activity can be manipulated or is shared across multiple instances (for example in virtualized environments) or where other specialist sources or entropy such as RdRand are either not present, have been disabled or are not trusted.

Furthermore, as VMs are instantiated there is a risk that the internal state of their PRNGs is either compromised through an attacker's access to the 'golden image' or is identical to other VM instances that were replicated from the same source. To overcome these issues, the use of netRandom entropy ensures that each VM instance is rapidly, frequently and independently re-seeded to ensure that for each instance the local Windows or Linux PRNGs quickly becomes unique and stays unpredictable to an external attacker. With netRandom, this level of re-seeding is achieved independently of the hardware platform in use, the local physical environment and type and configuration of the hypervisor.

On Linux, as a result of the rapid re-seeding capabilities of the netRandom Client, the normally blocking device /dev/random, which blocks requests for random numbers if insufficient entropy is present, will rarely block even when under heavy load. Similarly, the non-blocking device, /dev/urandom, will continue to operate as normal but can now be regarded as a true random number generator due to the assurance that it is adequately seeded.

Linux netRandom Client Architecture Overview

The netRandom Client is designed to be lightweight, easy to install and secure while providing flexibility in terms of network bandwidth requirements and re-seeding rates. The Linux Client is made up of three components:

- C library that communicates with the netRandom Server to establish a local source of entropy
- Daemon to inject the entropy provided by netRandom into the Linux entropy pool
- Stunnel to provide SSL encryption for the client to server connection

The netRandom Client employs multi-layered security to protect against eavesdropping and man in the middle attacks. All communications between Client and Server are protected using the Transport Layer Security (TLS) protocol to encrypt all traffic and to guard against substitution and replay attacks. Additionally, the data delivered to the Client is authenticated and integrity checked using a standard HMAC operation with a dedicated pre-shared key.

The primary components of the netRandom Client are illustrated in Figure 1 and are briefly described below:

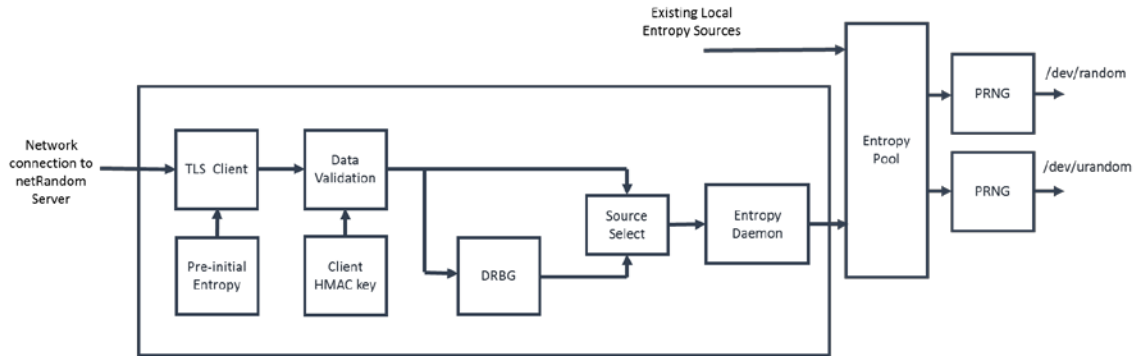


Figure 1- Client Entropy Input and Output

- TLS Client: Terminates SSL/TLS connection with netRandom Server. Decrypts received entropy from the encrypted channel, verifies server identity and checks message integrity. The Linux version of the netRandom Client uses Stunnel to create the SSL connection with the server.
- Pre-initial entropy: Pre-stored file of entropy to support initial TLS connection establishment. File initially provided out of band as part of enrollment package from netRandom Server and subsequently updated with fresh entropy.
- Data Validation: Primary integrity check on received entropy using HMAC and dedicated Client HMAC key.

- Client HMAC key: Persistent 256bit AES key provided out of band as part of enrollment package from netRandom Server.
- Deterministic Random Bit Generator (DRBG): NIST SP800-90A compliant process to generate a set of output bits that appear to be random but are deterministic based on the value of an initial seed – the function of a DRBG is to create a potentially large set of data from a relatively small seed value.
- Source Selector – configurable switch to determine where the output of the DRBG or the raw streamed entropy is used to supply the Entropy Daemon
- Entropy Daemon: Monitors the Linux Entropy Pool and requests fresh netRandom entropy whenever the Entropy Pool requires additional entropy
- Entropy Pool: Existing Linux cache of available entropy (maximum 4096 bits)
- Pseudo Random Number Generator (PRNG): Existing process to generate random numbers within Linux using entropy from the Entropy Pool act as data source for /dev/random and /dev/urandom.

Note – the terms DRNG and PRNG are used interchangeably, both terms are used in this document to enable the reader to distinguish between the DRBG within the netRandom Client and the PRNGs (DRNGs) within the Linux kernel.

The netRandom Client incorporates a DRBG for two important reasons. The first is to provide the option to reduce the amount of network traffic requested by the client. The second is to maintain a supply of entropy into the Linux Entropy Pool even in the event of a network failure to the netRandom Server and therefore reduce the potential for /dev/random to block

Windows netRandom Client Architecture Overview

The netRandom Client is designed to be lightweight, easy to install and secure while providing flexibility in terms of network bandwidth requirements and re-seeding rates. The Windows Client is made up of three components:

- **Entropy Source Connector:** a Windows DLL that handles the connection to the Whitewood Entropy Server
- **Entropy Router:** a C++ class which requests entropy data from the Entropy Source Connector and then routes it to the Entropy Consumer Agent registered to receive the entropy data
- **The Entropy Consumer**(s) – The getNetRandom free service supports only one Entropy Consumer: the CNG entropy pool.

The netRandom Client employs multi-layered security to protect against eavesdropping and man in the middle attacks. All communications between Client and Server are protected using the Transport Layer Security (TLS) protocol to encrypt all traffic and to guard against substitution and replay attacks. Additionally, the data delivered to the Client is authenticated and integrity checked using a standard HMAC operation with a dedicated pre-shared key.

The primary components of the netRandom Client are illustrated in Figure 1 and are briefly described below:

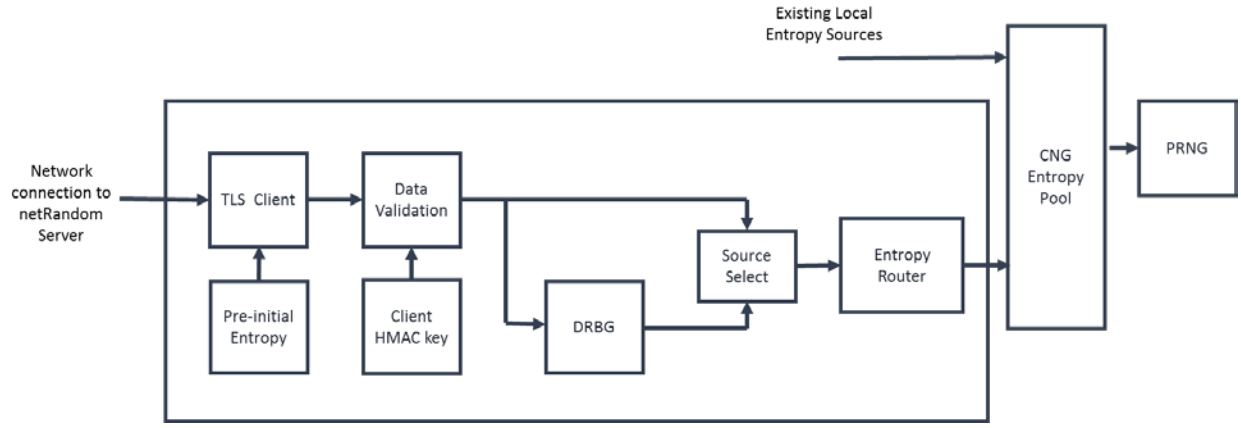


Figure 1- Client Entropy Input and Output

- TLS Client: Terminates SSL/TLS connection with netRandom Server. Decrypts received entropy from the encrypted channel, verifies server identity and checks message integrity. The Linux version of the netRandom Client uses Stunnel to create the SSL connection with the server.
- Pre-initial entropy: Pre-stored file of entropy to support initial TLS connection establishment. File initially provided out of band as part of enrollment package from netRandom Server and subsequently updated with fresh entropy.

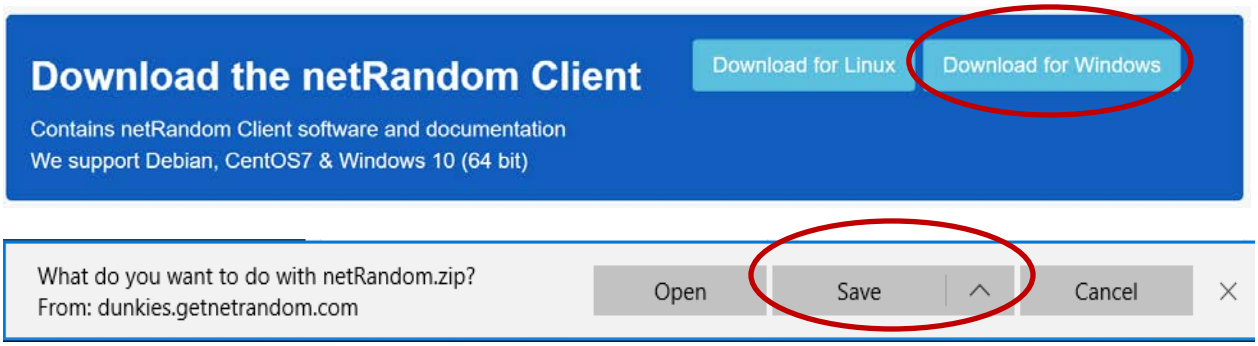
- Data Validation: Primary integrity check on received entropy using HMAC and dedicated Client HMAC key.
- Client HMAC key: Persistent 256bit AES key provided out of band as part of enrollment package from netRandom Server.
- Deterministic Random Bit Generator (DRBG): NIST SP800-90A compliant process to generate a set of output bits that appear to be random but are deterministic based on the value of an initial seed – the function of a DRBG is to create a potentially large set of data from a relatively small seed value.
- Source Selector – configurable switch to determine where the output of the DRBG or the raw streamed entropy is used to supply the Entropy Daemon
- Entropy Router: Requests fresh netRandom entropy and sends it to the CNG Entropy Pool subscriber
- Entropy Pool: Existing Windows cache of available entropy
- Pseudo Random Number Generator (PRNG): Existing process to generate random numbers within Windows using entropy from the Entropy Pool

Note – the terms DRNG and PRNG are used interchangeably, both terms are used in this document to enable the reader to distinguish between the DRBG within the netRandom Client and the PRNGs (DRNGs) within Windows CNG.

The netRandom Client incorporates a DRBG for two important reasons. The first is to provide the option to reduce the amount of network traffic requested by the client. The second is to maintain a supply of entropy into the Winows Entropy Pool even in the event of a network failure to the netRandom Server.

Windows Install procedure:

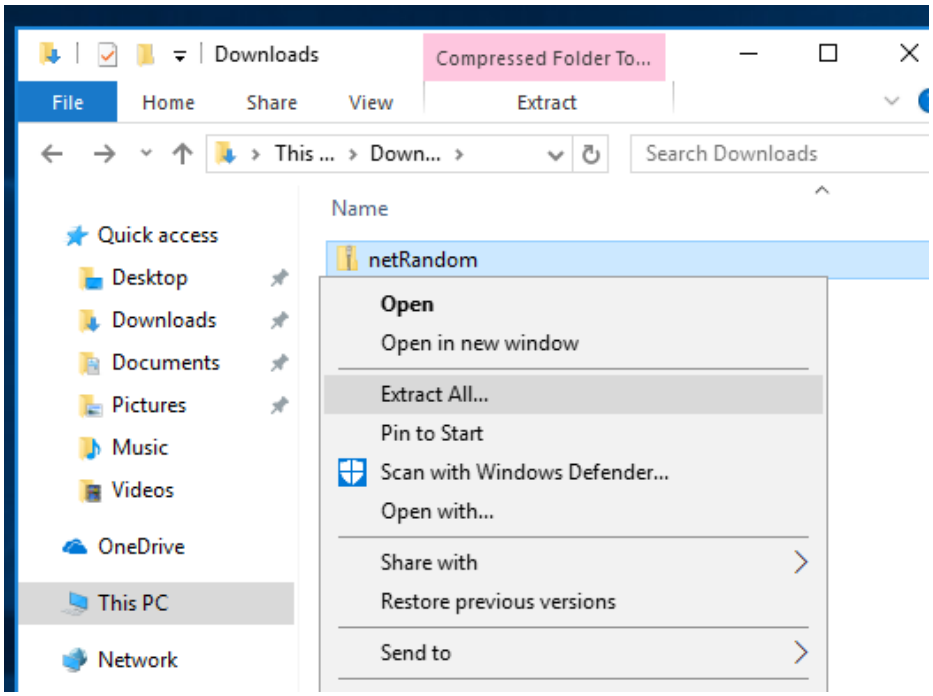
1. Download the Installation package from www.getnetrandom.com portal:



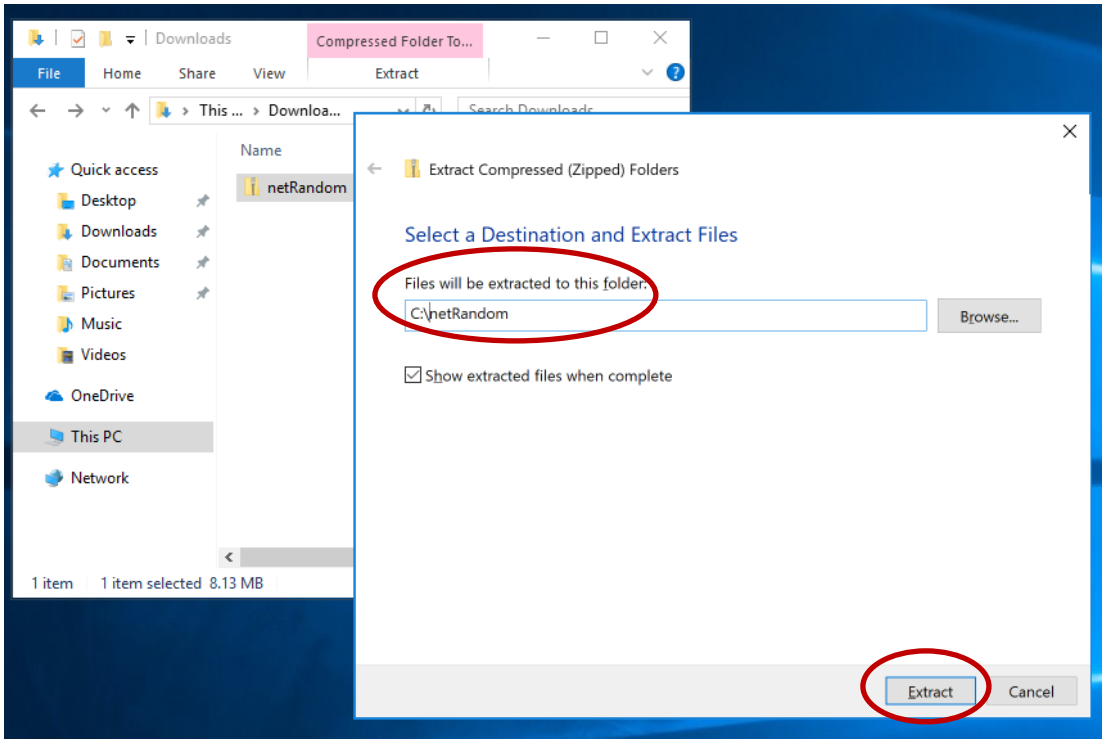
IMPORTANT: save the netRandom.zip file locally as it is a ZIP file.

2. Unzip the installation package using an unzip tool of your choice. If using the Windows Explorer's built in zip, the steps are as follows:

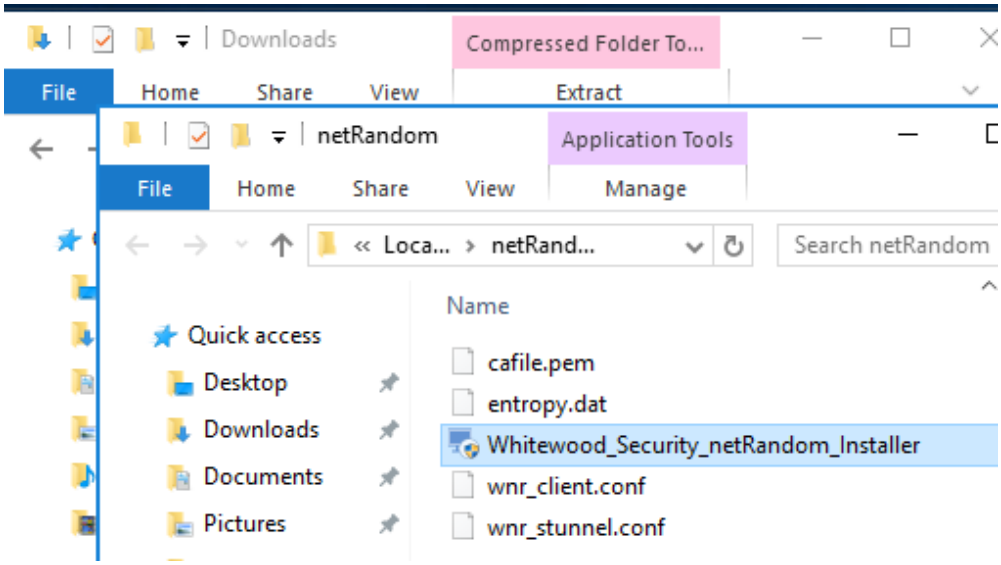
   a. Right click on the netRandom package and select "Extract All…"



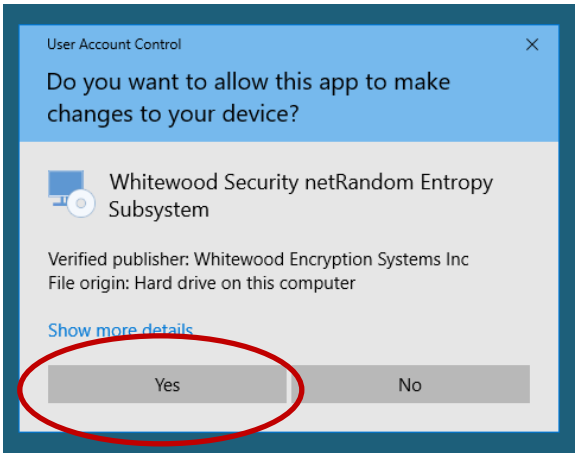
b. Select a destination and click the Extract button:

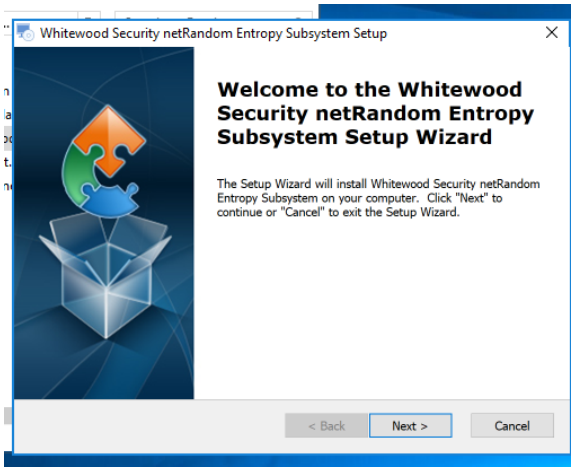


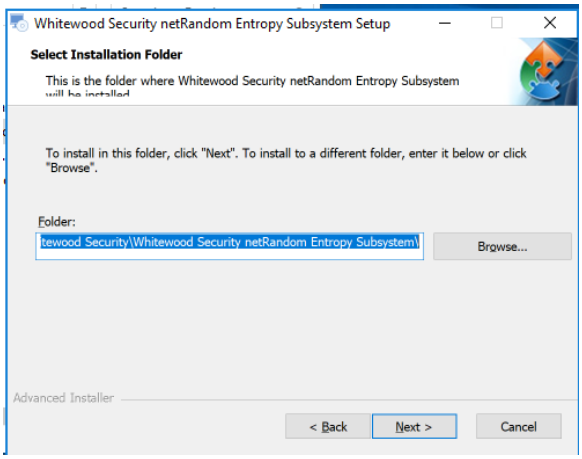
c. In the extracted location, double click on the installer Whitewood_Security_netRandom_Installer:

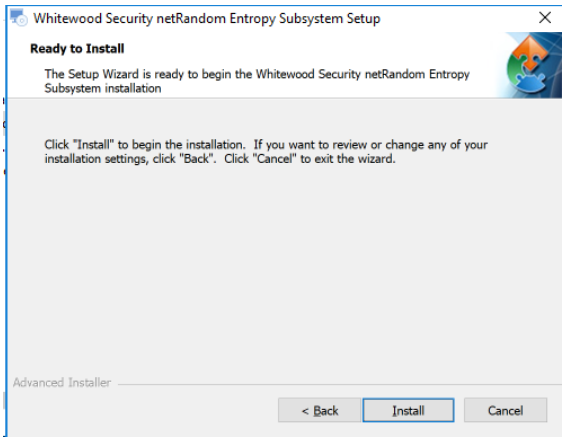


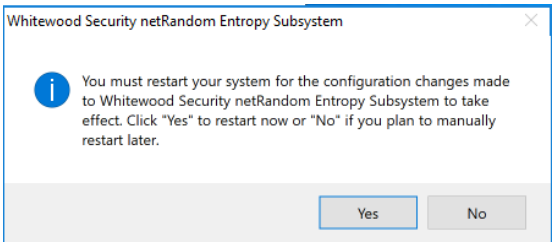
3. Run the Windows installer and follow the prompts in the installer:



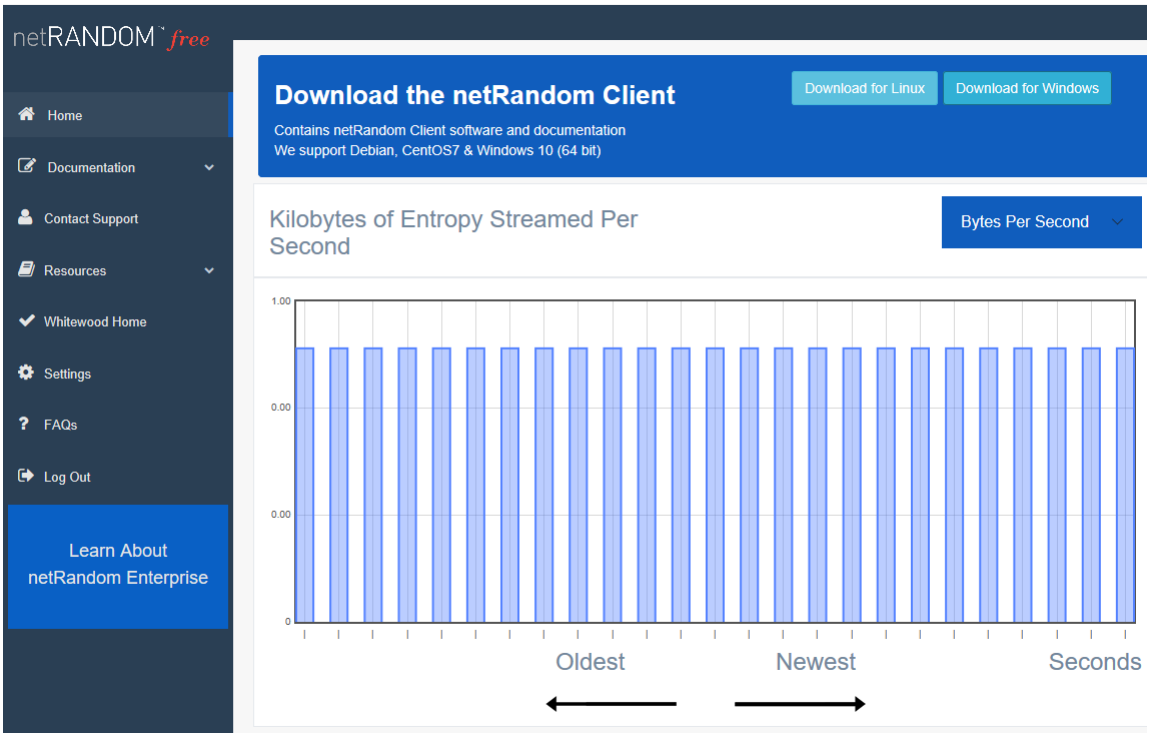








4. Log into the getNetRandom portal and confirm entropy is streaming to your machine:



If there is no streaming shown in the portal please contact support@getnetrandom.com or open the chat widget in the portal.

RPM Install procedure:

1. Download the Installation and Configuration package from www.getnetrandom.com portal:

   Save the file to your device.

2. Uncompress the config files
   - tar xzf <location of downloaded file>/netRandomConfig.tar.gz

     For example, if the config package was downloaded to ~/Downloads folder, the command would be
     - tar xzf ~/Downloads/netRandomConfig.tar.gz

3. Uncompress the Installers file:
   - tar xzf ./installers.tar.gz

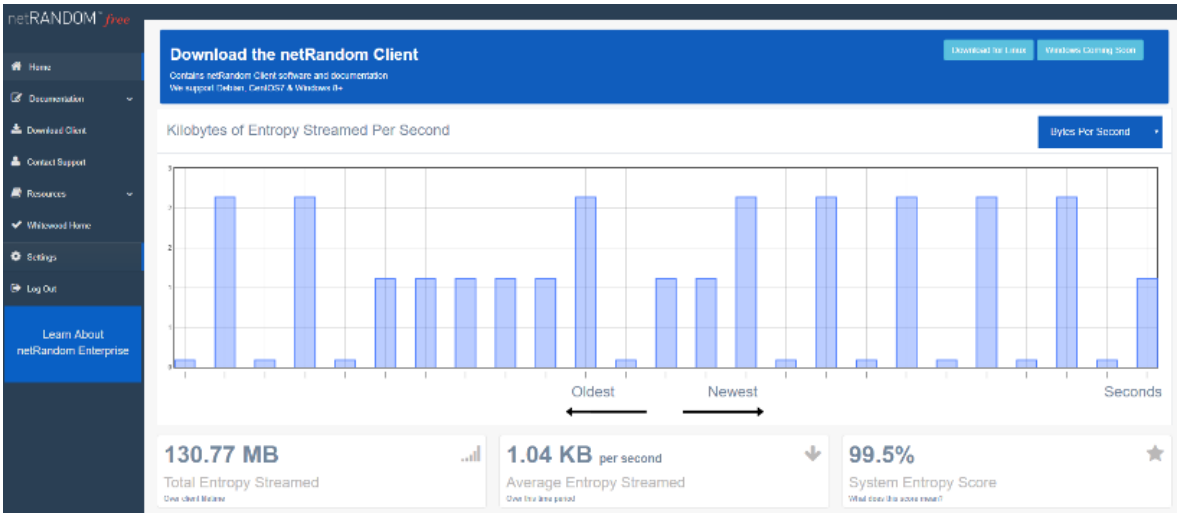
4. From the terminal shell, run local install:
   - sudo yum -y localinstall ./netrandom.rpm

5. Move the client configuration files into the /etc/netrandom directory
   - sudo mv ./etc/netrandom/* /etc/netrandom

6. Enable the netRandom client so it automatically starts when the machine boots:
   - sudo systemctl enable netrandomclient.service

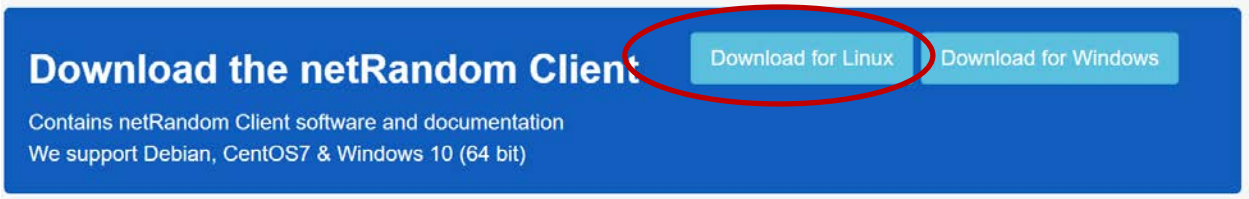
7. Reboot the machine, and the client will start automatically upon reboot

8. The client will now pull entropy from getNetRandom

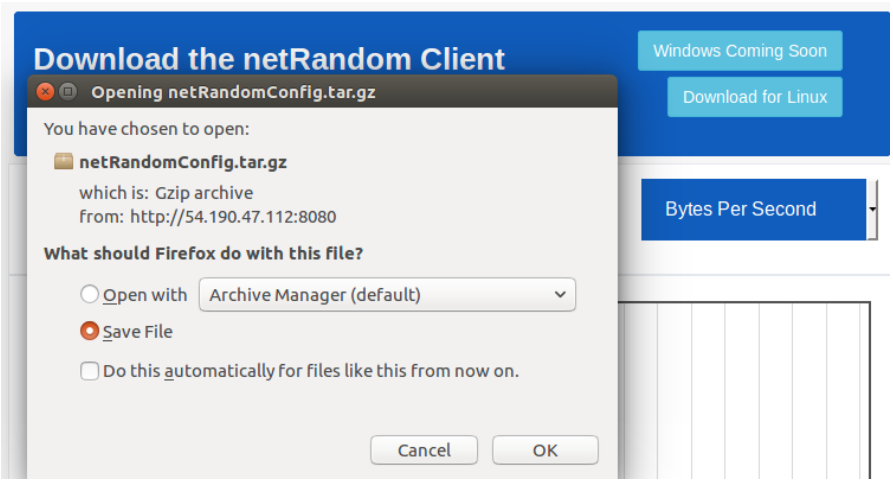


Ubuntu/Debian Installation Procedure

1. Download the Installation and Configuration package from www.getnetrandom.com portal:

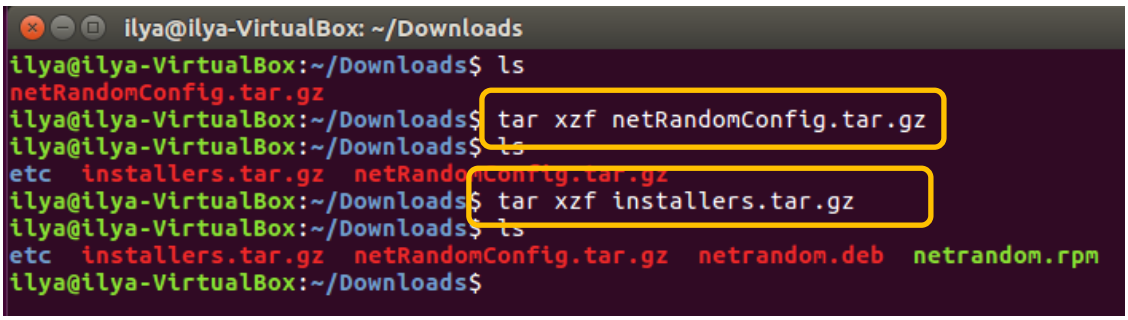


2. Save the file to the location of your choice (for example ~/Downloads)



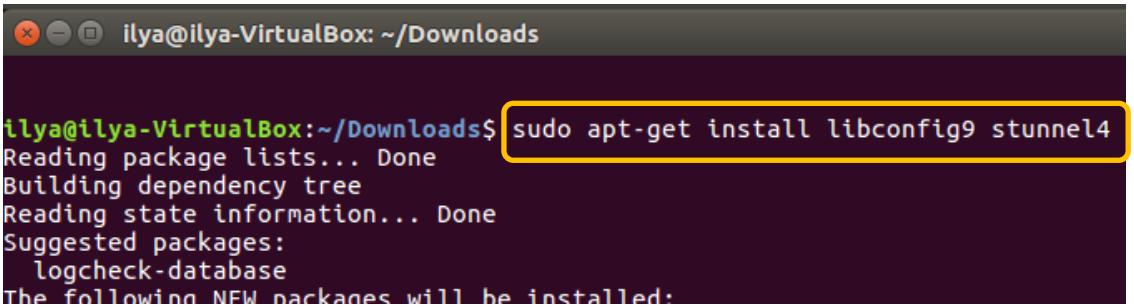
Note: In the examples below, the package has been downloaded to ~/Downloads, and all commands are run from inside that location

3. Unzip the netRandom configuration package and the installer package:
   using commands in the terminal/shell.
   - tar xzf netRandomConfig.tar.gz
   - tar xzf installers.tar.gz

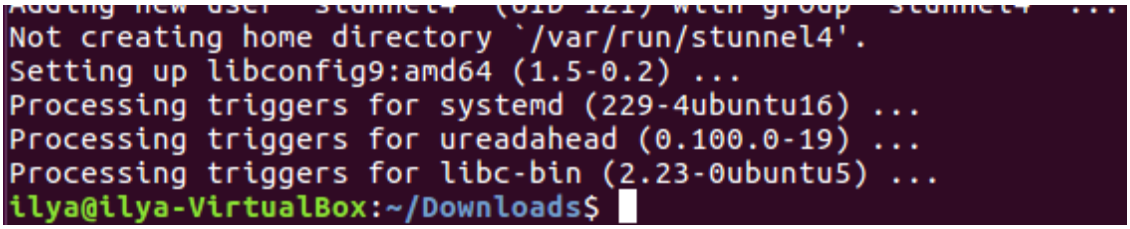


4. netRandom client has a dependency on *libconfig9* and *stunnel4* packages, which must be installed prior to installing the client:

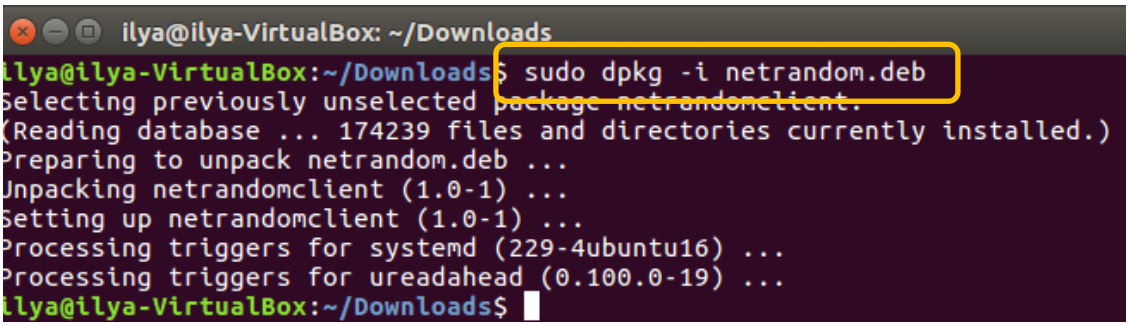
- sudo apt-get install libconfig9 stunnel4



. . .

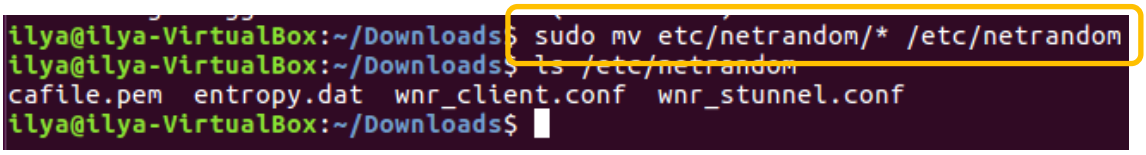


5. Run local install:
   - sudo dpkg -i netrandom.deb



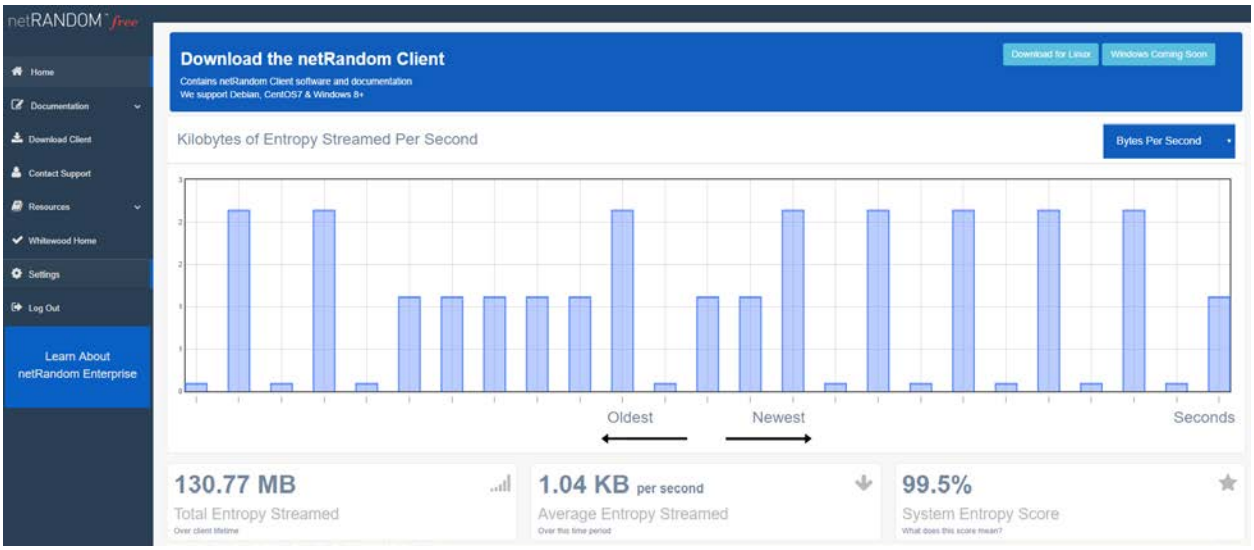
6. Move the client configuration files into the /etc/netrandom directory

   - sudo mv etc/netrandom/* /etc/netrandom



7. Reboot your machine at this point! (yes, a restart is required!)

8. After the machine restarts the netRandomclient will be pulling entropy from getNetRandom!



Linux Client configuration file

The netRandom Client (/etc/netrandom/wnr_client.conf and /etc/netrandom/wnr_stunnel.conf) have several pre-configured parameters. Contact support@getnetrandom.com or chat with us in the netRandom free app before making any changes to the config file.

Troubleshooting

SELinux

Depedning on the SELinux policy settings, SELinux may block stunnel from reading the configuration file. The symptoms are that after reboot there is no entropy being streamed from the network, and the netrandomtunnel.service service is not active.

Looking at the system logs the following error message will be present:

troubleshoot[3985]: SELinux is preventing /usr/bin/stunnel from read access on the file wnr_stunnel.conf. For co
thon[3985]: SELinux is preventing /usr/bin/stunnel from read access on the file wnr_stunnel.conf.

***** Plugin catchall (100. confidence) suggests **************************

If you believe that stunnel should be allowed read access on the wnr_stunnel.conf file by default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
ausearch -c 'stunnel' --raw | audit2allow -M my-stunnel
semodule -i my-stunnel.pp

The solution is to run the commands as suggested by SELinux error message to allow this access:

- ausearch -c 'stunnel' --raw | audit2allow -M my-stunnel
- semodule -i my-stunnel.pp