Installation and Maintenance of Navigator KeyTrustee

*The purpose of this article is to fully document the installation, configuration, and maintenance procedures that go into Navigator KeyTrustee (formerly zTrustee Server).*

**Overview**

Navigator KeyTrustee (formerly zTrustee Server) is a virtualized security module used to store and maintain sensitive objects. Navigator Encrypt (formerly zNcrypt) uses KeyTrustee to store its own encryption keys by default. In addition, KeyTrustee can also be integrated with other Hardware Security Modules (HSMs) in order to integrate more tightly with legacy systems, as well as to provide an extra layer of security.

For the remainder of this document zTrustee and KeyTrustee are interchangeable terms and refer to the same body of software.

**Installation**

**Prerequisites**

To install zTrustee Server, the first step is to make sure the necessary prerequisites are met:

**Hardware Requirements**

The following hardware requirements must be met:

* Processor: >= 1 GHz, >= 4 cores, x86-64
* Memory: >= 8 GB
* Storage: >= 20 GB

**Operating System Requirements**

Only the following Linux distributions and version are supported at this time:

* CentOS/RedHat Enterprise Linux 6.2 - 6.5
* Ubuntu 12.04

**Network Requirements**

The following ports are required to be open:

* HTTP - 80
* HTTPS - 443
* SMTP - 25

**Package Requirements**

The following packages will be installed as part of the zTrustee Server installation.

**Ubuntu**

* apache2
* apache2-mpm-worker
* apache2-utils
* apache2.2-bin
* apache2.2-common
* gnutls-bin
* libapache2-mod-wsgi
* libopendkim7
* libopts25
* libpq5
* libunbound2
* libvbr2
* ntp
* opendkim
* opendkim-tools
* openssl-blacklist
* pgbouncer
* postfix
* postgresql
* postgresql-9.1
* postgresql-client-9.1
* postgresql-client-common
* postgresql-common
* python-bcrypt
* python-egenix-mxdatetime
* python-egenix-mxtools
* python-flask
* python-jinja2
* python-magic
* python-markupsafe
* python-psycopg2
* python-sqlalchemy
* python-sqlalchemy-ext
* python-support
* python-tz
* python-werkzeug
* run-one
* sasl2-bin
* ssl-cert

**RedHat/CentOS**

* pytz
* pyOpenSSL
* apr-util-ldap
* m2crypto
* cyrus-sasl-plain
* python-psycopg2
* mod\_proxy\_html
* py-bcrypt
* python-argparse
* gnutls
* gnutls-utils
* httpd
* httpd-tools
* mod\_ssl
* mod\_wsgi
* openssl
* postgresql-server
* postgresql-libs
* postgresql

You do not need to explicitly install these packages (unless performing an offline installation), as they will be resolved by the package manager at a later step.

**Adding the Gazzang Repository**

As long as the system prerequisites have been met, then it's time to install the zTrustee Server packages. The first step is letting the package manager know where to find the correct dependencies. Depending on what Linux distribution you are using, these commands can vary slightly.

***Note!****The credentials used below for the 'cloudera' user are just for illustrative purposes. Please use your credentials.*

**Ubuntu**

Add the Gazzang repository to the apt configuration by running:

# Add the repo echo "deb https://cloudera:password@archive.gazzang.com/ubuntu/stable precise main" | sudo tee -a /etc/apt/sources.list   # Add the GPG signing key curl -sL http://archive.gazzang.com/gpg\_gazzang.asc | sudo apt-key add -

**RedHat/CentOS**

Add the Gazzang repository to the yum configuration by running:

sudo cat << EOF > /etc/yum.repos.d/gazzang.repo [gazzang] name=RHEL \$releasever - Gazzang baseurl=https://cloudera:password@archive.gazzang.com/redhat/stable/\$releasever enabled=1 gpgcheck=1 gpgkey=https://archive.gazzang.com/gpg\_gazzang.asc EOF

***Note!*** On some RHEL/CentOS systems, it might be necessary to also configure yum to use the EPEL repositories. For more information on EPEL (and how to configure yum), please see [here](https://fedoraproject.org/wiki/EPEL/FAQ#How_can_I_install_the_packages_from_the_EPEL_software_repository.3F). You can add the EPEL repository to your configuration by running:

sudo rpm -Uvh http://download.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm

**Installing the Server**

Once the repository has been appropriately configured for your package manager, we can install the server itself.

**Ubuntu**

sudo apt-get update && sudo apt-get install haveged -y

sudo service haveged start # this is to make sure the system entropy is high when generating GPG keys

sudo apt-get install ztrustee-server -y

**RedHat/CentOS**

sudo yum install ztrustee-server haveged -y

sudo service haveged start # this is to make sure the system entropy is high when generating GPG keys

# Add the necessary services to the start order

sudo chkconfig --level 2345 httpd on

sudo chkconfig --level 2345 postgresql on

sudo chkconfig --level 2345 postfix on

sudo chkconfig --level 2345 haveged on

Assuming that finished successfully, you're fully installed!

**Configuration**

**Post-Installation Scripts**

Once zTrustee Server is installed, we will need to run the post-installation scripts to setup the necessary services and schema.

***Note!****This step is performed automatically for Ubuntu users, so this can be skipped unless you are running RHEL/CentOS.*

**RedHat/CentOS**

sudo /usr/lib/ztrustee-server/postinst/setup-rh

Once completed, the zTrustee Server is fully functioning and operational.

**Verifying the Installation**

Once the post-installation scripts have been run, then you should be all set to start adding organizations and registering clients. To make sure that everything is up and running, you can run:

curl -k https://localhost/?a=fingerprint

Which should return the GPG public fingerprint of the new installation. This should look similar to:

4096R/643DE0D3F2C2B850C25997DE5C0C07848A386196

**Adding an Organization**

Before registering clients against the server, you will need to create an 'organization' to attribute clients and objects against. An organization is a means to logically separate clients and objects between different users and use-cases. At Gazzang, we would create an organization for each customer using our SaaS server.

As an example, we can create a simple test group using the orgtool command:

sudo /usr/lib/ztrustee-server/orgtool add -n test-group -c root@$(hostname -f)

Where 'test-group' is the name of the organization, and 'root@$(hostname -f)' is the organization contact. The organization contact will be alerted of each new client registration against that organization.

Once the organization has been created, you can list out more information by running:

sudo /usr/lib/ztrustee-server/orgtool list

Which should display more metadata around the organization. As an example:

Dropped privileges to ztrustee { "test-group": { "auth\_secret": "4KlTwFDMem/icvP+IQ0VRQ==", "contacts": [ "root@vagrant-centos64.vagrantup.com" ], "creation": "2014-07-08T19:35:50", "expiration": "9999-12-31T23:59:59", "key\_info": null, "name": "test-group", "state": 0, "uuid": "pyayLHhGsE8h7JN7VIB4fpiqV3VBVsKaNlO0NvJLqww" } }

Where the important pieces are:

* The organization **name** ('test-group' above), which will be required for all client registrations
* The **auth\_secret**parameter, which is the authorization secret that all clients will use to register against this organization. This parameter was randomly generated during the organization creation. This can also be set to a custom parameter by using the command:

sudo /usr/lib/ztrustee-server/orgtool set-auth -n test-group -s MYNEWSECRET

Once an organization is created, then clients can start registering and storing objects.

**Installing Custom SSL Certificates**

As part of the post-installation steps, self-signed SSL certificates were generated to enable immediate security of network communication. By default, these scripts will not be trusted by any client that is trying to register against your server, since they have not been signed by a trusted Certificate Authority (or CA). If you would like to install a custom set of SSL certificates, then you will need to replace the self-signed certificates with a set of CA-signed certificates.

To use a custom set of certificates, you will need to replace these three files with the ones provided by your CA or IT organization:

* /var/lib/ztrustee/.ssl/ssl-cert-ztrustee.pem – The SSL certificate, signed by a valid and trusted authority.
* /var/lib/ztrustee/.ssl/ssl-cert-ztrustee-pk.pem – The private key used to generate and sign the SSL certificate above.
* /var/lib/ztrustee/.ssl/ssl-cert-ztrustee-ca.pem – The chain (or intermediate) file, which should have been provided by your CA upon signing.

Once the necessary files are in place, you will need to restart the Apache process in order for your changes to take effect:

**Ubuntu**

service apache2 restart

**RedHat/CentOS**

service httpd restart

**Backups**

The only currently supported backup workflow for zTrustee Server is a "hot-warm" active failover method. This configuration involves two servers:

* One "hot" (primary) server, which actively services requests from clients. This server syncs file system data to the secondary server at predefined schedule (using the system cron).
* One "warm" (secondary) server, which is a passive replica of the primary server. This server sits idle until a failover occurs.

In its current state, this work flow requires manual intervention to failover between servers (this will not happen automatically). Also, it is useful to keep in mind that data only flows from the primary server to the secondary server. In the case of a failover scenario, you will need to set the old primary as the new secondary in order to keep the systems in sync.

The backup workflow typically performs the following series of steps (based on a cron):

* All zTrustee Server processes are stopped, which includes apache2/httpd, postgresql, postfix, and (on Ubuntu) pgbouncer.
* A *local* backup is taken of all of the necessary directories (defined in /usr/lib/ztrustee-server/backup-functions).
* All zTrustee Server processes are started again to resume accepting incoming requests from clients.
* The local backup is rsync'd to the secondary server, creating the replica version.

**Setting the Configuration**

The first step to setting up backup replication for zTrustee is to set the appropriate configuration. The configuration file that controls where data is synced can be found in '/etc/ztrustee/ztrustee.conf', which should look similar to:

**/etc/ztrustee/ztrustee.conf**

PKG="ztrustee" ZTRUSTEE\_SERVER="https://ztrustee.gazzang.com" ZTRUSTEE\_USER="$USER@$(hostname -f)" ZTRUSTEE\_GPG\_OPTS="--no-default-keyring --keyring $PKG" ZTRUSTEE\_SERVER\_USER="ztrustee" ZTRUSTEE\_SERVER\_GROUP="ztrustee" ZTRUSTEE\_SERVER\_HOME="/var/lib/$PKG" ZTRUSTEE\_SERVER\_CONFDIR="$ZTRUSTEE\_SERVER\_HOME/.$PKG" ZTRUSTEE\_SERVER\_CONF="$ZTRUSTEE\_SERVER\_CONFDIR/$PKG.conf" ZTRUSTEE\_SERVER\_LOGDIR="/var/log/$PKG" ZTRUSTEE\_SERVER\_LOG="$ZTRUSTEE\_SERVER\_LOGDIR/$PKG.log" ZTRUSTEE\_SERVER\_CERT="$ZTRUSTEE\_SERVER\_HOME/.ssl/ssl-cert-$PKG.pem" ZTRUSTEE\_SERVER\_PUB\_CERT="/etc/ssl/certs/ssl-cert-$PKG.pem" ZTRUSTEE\_SERVER\_PK="$ZTRUSTEE\_SERVER\_HOME/.ssl/ssl-cert-$PKG-pk.pem" ZTRUSTEE\_SERVER\_CA="$ZTRUSTEE\_SERVER\_HOME/.ssl/ssl-cert-$PKG-ca.pem" #ZTRUSTEE\_SERVER\_HOSTNAME= #ZTRUSTEE\_SERVER\_MAIL= #ZTRUSTEE\_WARM\_BACKUP\_DIR= #ZTRUSTEE\_WARM\_SPARE= #ZTRUSTEE\_WARM\_SPARE\_DIR=/ #ZTRUSTEE\_WARM\_SPARE\_DIR="/srv/ztrustee/$(date +%Y-%m-%d\_%H-%M-%S)" ZTRUSTEE\_SERVER\_HOSTNAME="vagrant-centos64.vagrantup.com" ZTRUSTEE\_SERVER\_MAIL="ztrustee@vagrant-centos64.vagrantup.com"

The variables we are most interested are:

* ZTRUSTEE\_WARM\_SPARE (line 19) - The hostname of the machine we want to set as secondary (ex: ZTRUSTEE\_WARM\_SPARE="mysecondary.mydomain.com")
* ZTRUSTEE\_WARM\_SPARE\_DIR (line 20) – The directory on the secondary server, where you would like to replicate data to. For a full replica, this will be the "/" (or root) directory, but can also be set to other directories to fit your use-case.

As an example, let's say that our secondary server's internal hostname is "ip-10-542-12-19.internal.aws.com", and we would like to make this server a full working replica of our primary. This will make our new configuration look like:

**/etc/ztrustee/ztrustee.conf**

PKG="ztrustee" ZTRUSTEE\_SERVER="https://ztrustee.gazzang.com" ZTRUSTEE\_USER="$USER@$(hostname -f)" ZTRUSTEE\_GPG\_OPTS="--no-default-keyring --keyring $PKG" ZTRUSTEE\_SERVER\_USER="ztrustee" ZTRUSTEE\_SERVER\_GROUP="ztrustee" ZTRUSTEE\_SERVER\_HOME="/var/lib/$PKG" ZTRUSTEE\_SERVER\_CONFDIR="$ZTRUSTEE\_SERVER\_HOME/.$PKG" ZTRUSTEE\_SERVER\_CONF="$ZTRUSTEE\_SERVER\_CONFDIR/$PKG.conf" ZTRUSTEE\_SERVER\_LOGDIR="/var/log/$PKG" ZTRUSTEE\_SERVER\_LOG="$ZTRUSTEE\_SERVER\_LOGDIR/$PKG.log" ZTRUSTEE\_SERVER\_CERT="$ZTRUSTEE\_SERVER\_HOME/.ssl/ssl-cert-$PKG.pem" ZTRUSTEE\_SERVER\_PUB\_CERT="/etc/ssl/certs/ssl-cert-$PKG.pem" ZTRUSTEE\_SERVER\_PK="$ZTRUSTEE\_SERVER\_HOME/.ssl/ssl-cert-$PKG-pk.pem" ZTRUSTEE\_SERVER\_CA="$ZTRUSTEE\_SERVER\_HOME/.ssl/ssl-cert-$PKG-ca.pem" #ZTRUSTEE\_SERVER\_HOSTNAME= #ZTRUSTEE\_SERVER\_MAIL= #ZTRUSTEE\_WARM\_BACKUP\_DIR= ZTRUSTEE\_WARM\_SPARE="ip-10-542-12-19.internal.aws.com" ZTRUSTEE\_WARM\_SPARE\_DIR=/ #ZTRUSTEE\_WARM\_SPARE\_DIR="/srv/ztrustee/$(date +%Y-%m-%d\_%H-%M-%S)" ZTRUSTEE\_SERVER\_HOSTNAME="vagrant-centos64.vagrantup.com" ZTRUSTEE\_SERVER\_MAIL="ztrustee@vagrant-centos64.vagrantup.com"

Where we have updated (and uncommented) the 'ZTRUSTEE\_WARM\_SPARE' and 'ZTRUSTEE\_WARM\_SPARE\_DIR' variables accordingly.

**Updating the Backup Scripts**

If you are using a RHEL/CentOS system, then you will also need to update the backup scripts to match the corresponding services and directories.

***Note!****If you are using Ubuntu, then you can skip this step.*

**backup-functions**

The first file we will need to modify is the '/usr/lib/ztrustee-server/backup-functions' file, which should currently similar to:

**/usr/lib/ztrustee-server/backup-functions**

APACHE\_DIRS="/etc/apache2/ /var/lib/ztrustee/ /var/log/apache2/" POSTGRES\_DIRS="/etc/postgresql/ /etc/postgresql-common/ /var/lib/postgresql/ /var/log/postgresql/ /etc/pgbouncer" POSTFIX\_DIRS="/etc/postfix/ /var/spool/postfix/ /var/lib/postfix/"

Which we will need to update to point to the RHEL equivalents:

**/usr/lib/ztrustee-server/backup-functions**

APACHE\_DIRS="/etc/httpd/ /var/lib/ztrustee/ /var/log/httpd/" POSTGRES\_DIRS="/var/lib/pgsql /etc/sysconfig/pgsql" POSTFIX\_DIRS="/var/lib/postfix /var/spool/postfix /etc/postfix"

**push-warm-spare**

The second file we will need to modify is the '/usr/lib/ztrustee-server/push-warm-spare' file, where:

**/usr/lib/ztrustee-server/push-warm-spare**

ssh $DEST 'for s in postfix apache2 pgbouncer postgresql; do service $s stop; done'

Should be modified to match our RHEL equivalents:

**/usr/lib/ztrustee-server/push-warm-spare**

ssh $DEST 'for s in postfix httpd postgresql; do service $s stop; done'

**warm-backup**

Our last file to modify is the '/usr/lib/ztrustee-server/warm-backup' file, where:

**/usr/lib/ztrustee-server/warm-backup**

service apache2 graceful

Should be modified to match our RHEL equivalents:

**/usr/lib/ztrustee-server/warm-backup**

service httpd graceful

**Setting the Cron**

Once our backup configuration is properly set and all scripts have been updated accordingly, we will need to define when the zTrustee Server should backup its data. This is typically defined in the '/etc/cron.d/ztrustee-server' file. If it is not there, then you will need to create it.

***Note!****On Ubuntu, this file should have been created for you.*

This file defines on what schedule data should be synced between the primary and secondary servers. To set the sync to be every half hour (top and middle of the hour), you can use the following contents:

**/etc/cron.d/ztrustee-server**

# Synchronize all server configuration, state, and logs to warm spare # Perform a local cold backup of all server configuration, state, and logs to warm spare 1,31 \* \* \* \* root /usr/lib/ztrustee-server/warm-backup # Synchronize local backup of server configuration, state, and logs to warm spare 6,36 \* \* \* \* root /usr/lib/ztrustee-server/push-warm-spare

Where the corresponding 1,31 and 6,36 define at what minute of every hour these actions should take place.

**Verifying the Backup Process**

Once all of the previous steps have been completed, we can run a quick test to make sure everything is correctly configured. By running:

sudo /usr/lib/ztrustee-server/warm-backup sudo /usr/lib/ztrustee-server/push-warm-spare

To ensure that the data is correctly synced between servers.

**Failing Over**

If you ever need to fail over between servers, you will have to run the following commands on the secondary server to mark it as active:

**Ubuntu**

rm -f /var/lib/ztrustee/.ztrustee/maint for i in "postgresql" "pgbouncer" "postfix" "apache2"; do service $i start; done

**RedHat/CentOS**

rm -f /var/lib/ztrustee/.ztrustee/maint   for i in "postgresql" "postfix" "httpd"; do service $i start; done

***Note!****You will also need to modify your DNS configuration (or however you registered the clients) to point to the new primary (previously secondary) server.*

**Maintenance**

**Common Troubleshooting Issues**