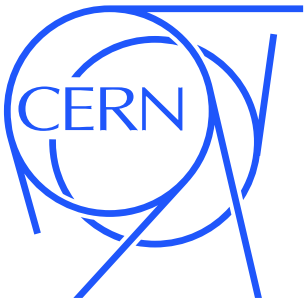

CernVM Release Testing Walkthrough

CernVM Release Testing Walkthrough Developer Manual

GNU USER

Technical Report
Version: 1.6 June 2011

Abstract

The CernVM Release Testing project is a testing infrastructure for CernVM images, the usecase for the project is to provide an automated testing environment, which will install and configure CernVM images, run the set of tests and report the results on a web interface.

Contents

1 Overview

CernVM currently supports images for VirtualBox, VMware, Xen, KVM and Microsoft Hyper-V hypervisors, each new release of a CernVM image needs to be thoroughly tested on each supported platform and hypervisor. The CernVM Release Testing project is designed to meet this requirement by providing an automated testing environment for CernVM images, which will install and configure CernVM images, run the set of tests and report the results on a web interface.

The intent of this document is to provided a step-by-step guide on setting up an entire CernVM Release Testing infrastructure, including instructions on how to set up and configure test clients, the main server running the web interface and database, as well as writing and executing tests. If you are new to release testing and want a document to guide you through the entire process of setting up a working CernVM Release Testing infrastructure, then this guide for you.

All the code needed to setup the entire Release Testing infrastructure for CernVM image testing, is located at the CernVM Release Testing Google Code project page[1] including this document and all other documentation.

While this document is not intended to be a replacement for the reference manual, the following is a brief description of the Release Testing infrastructure including an introduction to the core component, AMD Tapper [2]. Figure 1.1 consists of a diagram outlining the Tapper Architecture, which consists of test clients and a server, the server is what controls the test clients, gathers results, and then displays the results through a web interface.

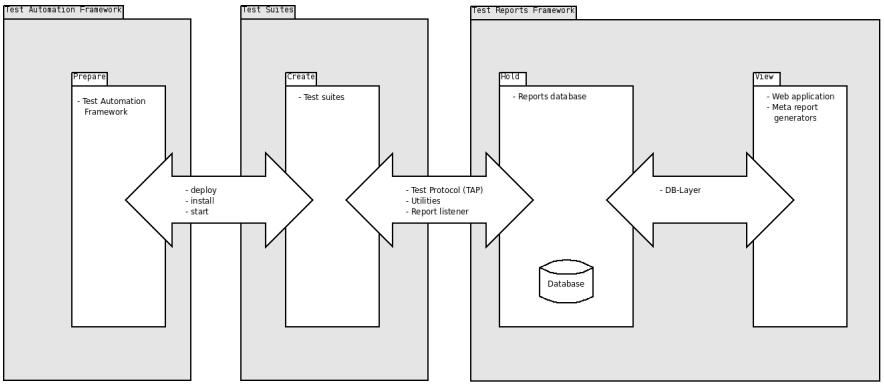

Figure 1.1: Overview of the Tapper architecture

2 CernVM Release Testing Test Client Platform Setup

2.1 Introduction

The intent of this document is to provided a step-by-step guide on setting up an entire CernVM Release Testing infrastructure, including instructions on how to set up and configure test clients, the main server running the web interface and database, as well as writing and executing tests. If you are new to release testing and want a document to guide you through the entire process of setting up a working CernVM Release Testing infrastructure, then this guide for you.

This section provides complete step by step instructions on how to setup and configure the test clients which are part of a basic working Release Testing environment by outlining the procedure for setting up test clients on numerous platforms, hence why this is called a *walkthrough* document. This guide is intended for users familiar enough with computers and desktop environments to enter basic commands in a terminal and install various operating systems.

As this guide is directed towards users who are new to CernVM Release Testing and Tapper and are interested in quickly getting a CernVM testing infrastructure quickly set up, many assumptions regarding the requirements necessary are made.As is the case, these instructions are provided for a generalized audience based on our own experience and the requirements that we feel most users will have, *so feel free to deviate from the instructions.*

2.2 Red Hat Based Test Client Setup

2.2.1 Configuring the system

1. After the system has booted remove the follow unnecessary startup applications by selecting from the menu `System -> Preferences -> Startup Applications`
   - bluetooth
   - evolution alarm
   - Gnome Login Sound
   - PackageKit Update Applet
   - print queue
   - screensaver
   - visual assistance/aid
   - volume control
   - any others you think are unnecessary based on your own discretion
2. Next enable and configure the remote desktop from the menu `System -> Preferences -> Remote Desktop` and ensure that the following options are configured
   - Enable the option "Allow others to view your desktop"
   - Enable the option "Allow other users to control your desktop"
   - Disable the option "You must confirm access to this machine"
3. Next enable SSH access to the machine, in order for SSH and VNC access to work the firewall will have to be disabled
   a. First disable the firewall from the menu `System -> Adminisration -> Firewall` and click the "Disable" button and then click "Apply" to apply the changes! **This is a quick solution for now because it's too much work to configure the firewall for VNC, SSH, Apache, MySQL, PHPMyAdmin, MCP, and all the other network daemons and should not be a problem if this is just being accessed internally**.
   b. Now that the firewall is disabled, configure sshd, the ssh daemon, to run on startup

      ```
      $ su -c "chkconfig --level 345 sshd on"
      ```

4. Next, configure the system to login automatically at boot
   a. Edit the login screen configuration file for gdm using the following command

      ```
      $ su -c "gedit /etc/gdm/custom.conf"
      ```

   b. Then in the custom.conf file, put the following under the heading [daemon], which will automatically log the system in as the user you created, *make sure you replace the user cernvm with the user that you created.*

Listing 2.1: Configure Automatic Login

```
AutomaticLoginEnable=true
AutomaticLogin=cernvm
```

5. Next, configure the screen saver from the menu `System -> Preferences -> Screensaver` and ensure that the following options are configured
   - Disable the option "Lock screen when screensaver active"
6. Now, reboot the machine, and ensure that the following work
   - It automatically boots up into the full desktop environment without having to login
   - You have access to the machine using SSH and can login on the root account
   - You have VNC access to the machine and can control the system using VNC
7. Finally, update the system from the menu `System -> Administration -> Software Update` and after it has completed the updates reboot the system

2.2.2 Installing libvirt and virsh

1. The virtualization API libvirt and the command line tool virsh [3] are the essential components required for setting up a test client and must be installed and properly configured before any testing can begin. Ensure that you follow the proceeding directions carefully and validate that virsh is working properly before proceeding to install and configure the various hypervisors.
2. First, begin by reviewing the release news listed on the libvirt website, http://libvirt.org/news.htm and read through the release notes for the latest version released to make sure that there are no regressions or deprecated support for the platforms you wish to support. If you intend to set up an entire infrastructure and support all of the CernVM virtualization platforms, which would include *Xen, KVM, VirtualBox, and VMware*, then you must download a version later than *0.8.7* as there was no support for VMware prior to that release.
3. Next, download the latest release that is a **tar.gz** file from the libvirt release server, http://libvirt.org/sources/ based on the latest release which does not have any regressions or deprecations for the virtualization platforms you wish to support [1]. As of this date, the latest release of libvirt is version 0.9.2, this is the release that will be used for the following instructions and examples.
4. Next, install the following dependencies which are required to build libvirt from source, *from now on execute all commands as root.*

[1] This shouldn't be an issue but just in case there is a newer version in which Xen support is deprecated, then you would need to use the last release which has Xen support

Listing 2.2: Install Dependencies

```
# Change to root account, enter password if prompted
$ su

# Install dependencies for building from source
$ yum install gnutls-devel numactl numactl-devel python-devel libnl-devel \
libxml2-devel device-mapper-libs device-mapper-devel

# Install GCC
$ yum install gcc make
```

5. Next, install software for managing and viewing virtual machines, *it is imperative that these applications are installed before compiling and installing libvirt otherwise there will be conflicts.*

Listing 2.3: Install Virtual Machine Management Software

```
# Change to root account, enter password if prompted
$ su

# Install management software
$ yum install virt-manager python-virtinst virt-viewer
```

6. Next extract the files and execute configure with the following options, then finally compile and install libvirt.

Listing 2.4: Compile and Install libvirt

```
# Change to root account, enter password if prompted
$ su

# Extract and execute configure
$ tar -xvvzf libvirt-*.tar.gz
$ cd libvirt-*/
$ ./configure --prefix=/usr --disable-silent-rules --disable-shared \
--enable-static --enable-dependency-tracking --with-qemu \
--with-vmware --with-libssh2 --with-vbox --with-test --with-remote \
--with-libvirtd --with-numactl --with-network --with-storage-dir

# Compile and install libvirt
$ make
$ make install
```

7. Finally, verify that the the service libvirtd can start, because it is required for libvirt to function, and then configure libvirtd to start automatically at boot. As well, ensure that virsh installed correctly and is running by connecting to

the test hypervisor and ensuring that the test virtual machine, named "test" is running.

Listing 2.5: Verify virsh was Installed Properly

```
# Change to root account, enter password if prompted
$ su

# Verify libvirtd starts, set to start automatically at boot
$ service libvirtd start
$ chkconfig --level 2345 libvirtd on

# Verify vitual machine "test" running
$ virsh -c test:///default list --all
```

2.2.3 Installing and configuring KVM

1. The first step is to start by installing the KVM package, KVM may have alrready been installed by default, If you receive a message that a package is already installed then simply continue.

   Listing 2.6: Install KVM

   ```
   $ yum install qemu-kvm
   ```

2. Next, verify the location of the qemu-kvm binary, at the moment virsh still expects that the binary is located in `/usr/bin`, but since Red Hat 6, the binary has been moved into the directory `/usr/libexec`. To resolve the issue simply create a symbolic link to the binary if qemu-kvm does not exist.

   Listing 2.7: Symbolic Link to qemu-kvm

   ```
   # If qemu-kvm does not exist create symbolic link
   $ ln -s /usr/libexec/qemu-kvm /usr/bin/qemu-kvm
   ```

3. Next, verify that KVM has been installed properly and that virsh can connect to the KVM hypervisor using the following commands, if you are able to connect to the virsh console without any errors then virsh is able to connect to the KVM hypervisor.

   Listing 2.8: Verify that virsh can Access KVM

   ```
   $ su
   $ virsh -c qemu:///session
   ```

4. Next, to ensure that KVM is properly configured and installed, follow this guide provided on the CernVM website http://cernvm.cern.ch/portal/kvm **except, do not create a kvm definition file as the xml template file is provided by the test suite scripts** and verify that you are able to connect to the libvirtd kvm system session.

Listing 2.9: Verify that KVM is Properly Configured

```
$ su
$ virsh -c qemu:///system
```

5. Finally, ensure that you are able to connect to the QEMU/KVM hypervisor using the virtual machine manager, as it is necessary to view the status of the CernVM images and must be installed to troubleshoot and view the CernVM images. Simply launch the virtual machine manager application from `Applications -> System Tools -> Virtual Machine Manager` and **if you are prompted to install libvirt, select "No"** as a custom version of libvirt was installed previously.

2.2.4 Installing and configuring VirtualBox

1. First, begin by downloading and installing a version of VirtualBox supported by libvirt from the VirtualBox download page, it is best to download the latest version within the series *that has been available for at least a month prior to the release of the version of libvirt installed.* VirtualBox can be downloaded from the following location, http://www.virtualbox.org/wiki/Downloads ensure that you select the appropriate Red Hat based distribution, version and architecture for your system. The following instructions for this section of the guide uses VirtualBox 4.0.12 for Red Hat 6, AMD64.

2. Before installing VirtualBox, install the dependencies required to build the VirtualBox kernel modules.

Listing 2.10: Install VirtualBox Dependencies

```
$ su
$ yum install kernel-headers kernel-devel
```

3. Next, after downloading the latest version of VirtualBox for your distribution and installing the dependencies install VirtualBox as the root account using the following command.

Listing 2.11: Install VirtualBox Dependencies

```
# Enter the root password when prompted
$ su
$ rpm -iv VirtualBox-*.rpm
```

4. Next, in order to use VirtualBox and have full access to the drivers needed, ensure that the root account belongs to the group "vboxusers". Add the root account to the group "vboxusers" using the following command.

Listing 2.12: Add root to vboxusers

```
$ su
$ usermod -a -G vboxusers root
```

5. Due to an issue with VirtualBox[2], in order for it to work with virsh the virtual machine(s) must be created and configured as the root account, otherwise when you try to connect or start a VirtualBox virtual machine with virsh you will get an "unknown error", which is obviously very vague and difficult to resolve. **Therefore ALWAYS start VirtualBox as the root account using the following procedure**.

Listing 2.13: Always Start VirtualBox as Root

```
# Switch to the root account, enter root password
$ su

# Start VirtualBox as root
$ virtualbox
```

6. Finally, verify that VirtualBox has been installed properly and that virsh can connect to the VirtualBox hypervisor, verify that the VirtualBox module, *vboxdrv* has been loaded and that you are able to connect to the virsh console without any errors.

Listing 2.14: Verify that virsh can Access VirtualBox

```
$ su

# Verify that the vboxdrv module is loaded
$ lsmod | grep -i vboxdrv

# Verify that virsh can connect to virtualbox
$ virsh -c vbox:///session
```

2.2.5 Installing and configuring VMware

1. First, begin by downloading the latest version of VMware Workstation or VMware Player from the VMware product page, http://www.vmware.com/products/, VMware Player is free, whereas VMware Workstation requires a license. So if you decide to use VMware Workstation instead of VMware Player you will have to purchase a license for it in order to continue.

2. Before installing VMware, install the dependencies required to build the VMware kernel modules.

Listing 2.15: Install VMware Dependencies

```
$ su
$ yum install kernel-headers kernel-devel
```

[2]The issues is that VirtualBox looks for virtual machine configuration files (*.vbox) in the "VirtualBox VMs" folder of the user that launched VirtualBox. The issue is worsened by the fact that there can only be one "VirtualBox VMs" folder which causes conflicts with multiple users.

3. Next, to install VMware simply set the bundle file as executable and execute the file as root.

Listing 2.16: Install VMware

```
$ su
$ chmod +x VMware*.bundle
$ ./VMware*.bundle
```

4. Next, launch VMware as root and wait for it to compile the kernel modules, then verify that the following VMware kernel modules are loaded, currently virsh has support to connect to the VMware hypervisor, but there are some minor issues such as a lack of support for VMware network configurations, currently only "bridged" mode is supported.

   - vmnet
   - vmblock
   - vmci
   - vmmon

Listing 2.17: Verify VMware Kernel Modules Loaded

```
# Launch VMware as root, this will build kernel modules
$ su
$ vmware

# Verify that the kernel extentsions are loaded
$ lsmod | grep -i vm
```

.

5. Finally, verify that VMware has been configured properly and that virsh supports the current version of VMware installed by connecting to the virsh console for the VMware hypervisor.

Listing 2.18: Verify VMware Works with Virsh

```
# Verify that virsh can connect to vmware
$ virsh --connect vmwarews:///session
```

6. If everything so far has worked, then libvirt, virsh, and the hypervisors have been installed and configured properly, if you have any outstanding issues solve them before proceeding further, or go to the section "Server Platform Setup" 3 as the Tapper server does not require libvirt, virsh, or hypervisor configuration.

2.2.6 Setting up the Tapper Test Suite

1. **Before proceeding any further ensure that you have all other test clients set up this far, and then proceed to follow the instructions for setting up and configuring the Tapper server in the section "Server Platform Setup"** 3.

2. Now that the Tapper server has been installed and configured and the Tapper web interface and database have proven to be working, the next step is to verify that the test client can actually send a report to the Tapper server in the form of a TAP file. After sending the TAP report to the server, ensure that the test client is working by viewing the tapper reports in your browser at the following url: http:/<tapper_server>/tapper/reports. You should now see a report from the test client, there should be a report from a system named whatever the "Tapper-Machine-Name" in demo_report.tap was set as. *For the example demo_report.tap provided below it would be cernvm-rhtestclient.* [3].

Listing 2.19: Send a Basic Report to the Tapper Server

```
# Save the following in a file named demo_report.tap

        1..2
        # Tapper-Suite-Name: Tapper-Deployment
        # Tapper-Suite-Version: 1.001
        # Tapper-Machine-Name: cernvm-rhtestclient
        ok - Hello World
        ok - Just another description

# Send the report to the tapper server using netcat
$ cat demo_report.tap | nc -w10 cernvm-server 7357
```

3. Next, download a copy of the CernVM Test Suite and the CernVM Test Cases from the Google Code svn repository [1] and install the the following dependencies.

Listing 2.20: Install CernVM Test Suite and Dependencies

```
# Install subversion, required to checkout auto-tapper
$ yum install subversion

# Checkout a copy of cernvm testsuite and cernvm testcases
$ svn checkout http://cernvm-release-testing.googlecode.com/svn/\
trunk/tapper/tapper-autoreport/ cernvm-testsuite

# Install the missing dependencies
```

[3] This is why a consistent hostname convention was emphasized earlier, as reports are often sorted and organized based on hostnames

```
$ yum install prove
$ yum install uuid
$ yum install spawn
$ yum install expect
$ yum install expectk
$ yum install nmh

# Run install-mh and accept the defaults
$ install-mh
```

4. Now that cernvm-testsuite has been installed, configure the variables in the configuration file for the hypervisors you want to test and according to your Tapper infrastructure setup. Sample configuration files are provided in the **config** folder, all of the settings provided in the configuration files by default are the **mandatory, minimum configuration options** and in most cases the defaults should be sufficient for testing, the only mandatory settings that are not provided by default **and must be configured manually are CVM_TS_REPORT_SERVER and CVM_VM_IMAGE_VERSION** . Please refer to the developer manual for a more complete detailed list of the mandatory and optional configuration settings.

   CVM_TS_SUITENAME Must ALWAYS be set, the default suite name in the configuration file should be suitable

   CVM_TS_SUITEVERSION Must ALWAYS be set, reflects the release version number of the test suite framework, the default version given in the configuration should only be changed if you customize/update the scripts

   CVM_TS_REPORT_SERVER Must ALWAYS be set, this is the ip address or hostname of the Tapper report server which the reports from the test results are sent to

   CVM_TS_DOWNLOAD_PAGE Must ALWAYS be set, normally the default url provided in the configuration file is accurate, but in the event that the internal CernVM image release page is relocated then this url must be changed.

   CVM_VM_HYPERVISOR Must ALWAYS be set, should not have to change the defaul hypervisor in the configuration files, valid values (case sensitive) are **kvm,vbox,vmware**

   CVM_VM_TEMPLATE Must ALWAYS be set, normally the default template provided in the configuration file should not be changed, only change this to use a custom template file. The custom template file *must be placed within the templates folder*

   CVM_VM_NET_TEMPLATE Must ALWAYS be set, normally the default network template provided in the configuration file should not be changed, only change this to use a custom network template file for the CernVM image,

only applies to kvm and virtualbox. The custom network template file *must be placed within the templates folder*

CVM_VM_IMAGE_VERSION Must ALWAYS be set, specifies the version of the CernVM image to use from the release page

CVM_VM_IMAGE_TYPE Must ALWAYS be set, specifies the type of CernVM image, valid image types supported, (case sensitive) are **basic and desktop**

CVM_VM_ARCH Must ALWAYS be set, specifies the architecture of the CernVM image, valid architectures (case sensitive) are **x86 and x86_64**

5. Finally, now that cernvm-testsuite has been installed and configured on the test client and the test client and Tapper Server have proven to be working, the next step is to verify that cernvm-testsuite works correctly and can actually send a report to the Tapper server in the form of a TAP file. To execute the CernVM Test Cases script, "cernvm-tests.sh" simply source the configuration file you wish to use and execute the script. Once the script has completed and sent a TAP report to the server, ensure that the test client is working by viewing the tapper reports in your browser at the following url: `http:/<tapper_server>/tapper/reports`. You should now see a new report from the test client, there should be a report from a system with the same hostname.[4].

Listing 2.21: Execute CernVM Test Cases Script

```
# Simply source the configuration file, and execute the script
$ . ./config/<configuration_file>
$ ./cernvm-tests.sh
```

[4]This is why a consistent hostname convention was emphasized earlier, as reports are often sorted and organized based on hostnames

2.3 Debian Based Test Client Setup

2.3.1 Installing the system

1. Install the system as you would for any other Linux distribution, except pay attention to the following instructions on configuring Debian to use ext4 (if available) instead of ext3 as the performance gains are noticeable.

2. Now, when prompted by the installer to configure the partitioning layout, if there are other operating systems installed on the system select the "Guided - use entire disk" option, and if available select the option "Use Remaining Free Space". Otherwise, if there are no other operating system installed on the hard drive select the "Manual" option, *beware that doing so will risk erasing everything on the hard drive if you create a new partition table.* Using the manual option, create two primary partitions, with the first taking up the size of the hard drive minus twice the size of the amount of RAM installed, and the second primary partition as a SWAP file using the remaining free space. The following is an example of what the partiton layout would look like for a 40.0 GB hard drive with 2GB of ram.

Listing 2.22: Manual Partition Layout Example

```
#1      PRIMARY  36.0  GB      B  f      EXT4   /
#2      PRIMARY  4.0   GB         f      SWAP   SWAP
```

3. Finally, the last important installation setting, when prompted to choose software to install, select the following
   - Graphical desktop environment
   - SSH Server
   - Standard system utilities

2.3.2 Configuring the system

1. After the system has booted remove the follow unnecessary startup applications by selecting from the menu
   `System -> Preferences -> Startup Applications`
   - bluetooth
   - evolution alarm
   - Gnome Login Sound
   - print queue
   - screensaver
   - update notifier
   - visual assistance/aid

- volume control
- any others you think are unnecessary based on your own discretion

2. Remove the follow unnecessary services by selecting from the menu `System -> Administration -> Services`
   - alsa utils
   - bluetooth
   - CUPS
   - exim4
   - any others you think are unnecessary based on your own discretion
3. Next enable and configure remote desktop from the menu `System -> Preferences -> Remote Desktop` and ensure that the following options are configured
   - Enable the option "Allow others to view your desktop"
   - Enable the option "Allow other users to control your desktop"
   - Disable the option "You must confirm access to this machine"
4. Next configure the system to login automatically at boot from the menu select `System -> Administration -> Login Screen` and then set it to login to the user account you created previously (such as cernvm) automatically.
5. Next, remove cd-rom support from sources.list, which is used by Debian for updates [5], execute the following command with root privileges and comment out any lines that start with "deb cdrom" by using a #

   Listing 2.23: Removing CD-ROM Requirement for Updates

   ```
   $ su -c "gedit /etc/apt/sources.list"
   ```

6. Again, continue to edit /etc/apt/sources.list still with root privileges and ensure that each line ends with "main contrib non-free", then save the file and do the following command with root privileges.

   Listing 2.24: Updating the System

   ```
   $ su -c "apt-get update"
   ```

7. Next, configure the screen saver from the menu `System -> Preferences -> Screensaver` and ensure that the following options are configured
   - Disable the option "Lock screen when screensaver active"

[5] And is a nuisance for any new user as it forces you to find the CD and put it in the computer for the update to continue

8. The following instructions involve enabling headless support so that you can remote desktop to the machine without having a monitor connected to the computer
   a. Edit the xorg.conf file and put the following in it

      Listing 2.25: Configuring Xorg for Headless Support

```
$ su -c "gedit /etc/X11/xorg.conf"

Section "Device"
Identifier "VNC Device"
Driver "vesa"
EndSection

Section "Screen"
Identifier "VNC Screen"
Device "VNC Device"
Monitor "VNC Monitor"
SubSection "Display"
Modes "1280x1024"
EndSubSection
EndSection

Section "Monitor"
Identifier "VNC Monitor"
HorizSync 30-70
VertRefresh 50-75
EndSection
```

   b. Then edit grub and set the option "nomodeset", and proceed to update grub and reboot

      Listing 2.26: Configuring Grub for Headless Support

```
$ su -c "gedit /etc/default/grub"

GRUB_CMDLINE_LINUX="nomodeset"

$ su -c "update-grub"
```

9. Now, reboot the machine, and ensure that the following work
   - It automatically boots up into the full desktop environment without having to login
   - You have access to the machine using SSH and can login on the root account

- You have VNC access to the machine and can control the system using VNC

10. Finally, update the system from the menu `System -> Administration -> Update Manager` and after it has completed the updates reboot the system

2.3.3 Installing libvirt and virsh

1. The virtualization API libvirt and the command line tool virsh [3] are the essential components required for setting up a test client and must be installed and properly configured before any testing can begin. Ensure that you follow the proceeding directions carefully and validate that virsh is working properly before proceeding to install and configure the various hypervisors.

2. First, begin by reviewing the release news listed on the libvirt website, http://libvirt.org/news.htm and read through the release notes for the latest version released to make sure that there are no regressions or deprecated support for the platforms you wish to support. If you intend to set up an entire infrastructure and support all of the CernVM virtualization platforms, which would include *Xen, KVM, VirtualBox, and VMware*, then you must download a version later than *0.8.7* as there was no support for VMware prior to that release.

3. Next, download the latest release that is a **tar.gz** file from the libvirt release server, http://libvirt.org/sources/ based on the latest release which does not have any regressions or deprecations for the virtualization platforms you wish to support [6]. As of this date, the latest release of libvirt is version 0.9.2, this is the release that will be used for the following instructions and examples.

4. Next, install the following dependencies which are required to install the libvirt files from the source files that were downloaded, *from now on execute all commands as root.*

Listing 2.27: Install Dependencies

```
$ su
$ apt-get install libxen-dev gnutls-dev libnuma-dev \
libdevmapper-dev python-dev libnl-dev libxml2 \
libxml2-dev libgnutls-dev

# Install GCC
$ apt-get install gcc make build-essential
```

5. Next, install software for managing and viewing virtual machines.

[6] This shouldn't be an issue but just in case there is a newer version in which Xen support is deprecated, then you would need to use the last release which has Xen support

Listing 2.28: Install Virtual Machine Management Software

```
# Change to root account, enter password if prompted
$ su
$ apt-get install virt-manager virtinst virt-viewer
```

6. Next extract the files and execute configure with the following options, then finally compile and install libvirt.

Listing 2.29: Compile and Install libvirt

```
# Extract and execute configure
$ tar -xvvzf libvirt-*.tar.gz
$ cd libvirt-*/
$ ./configure --prefix=/usr --disable-silent-rules --disable-shared \
--enable-static --enable-dependency-tracking --with-xen \
--with-xen-inotify --with-qemu --with-vmware --with-libssh2 \
--with-vbox --with-test --with-remote --with-libvirtd --with-numactl \
--with-network --with-storage-dir

# Compile and install libvirt
$ make
$ make install
```

7. Finally ensure that virsh installed correctly and is running by connecting to the test hypervisor and ensuring that the test virtual machine, named "test" is running.

Listing 2.30: Verify virsh was Installed Properly

```
# Change to root account, enter password if prompted
$ su

# Verify virsh is working, test should be running
$ virsh -c test:///default list --all
```

2.3.4 Installing and configuring KVM

1. The first step is to install the KVM hypervisor, start by installing KVM and the other additional packages such as virt-manager, which is a graphical management tool and virt-install, which is a command line interface (CLI) virtual machine creation/installation/configuration tool using the following commands with root privileges. If you receive a message that a package is already installed then simply continue.

Listing 2.31: Installing KVM and Other Related Programs

```
$  su
$ apt-get install qemu-kvm
```

2. Next, verify that KVM has been installed properly and that virsh can connect to the KVM hypervisor using the following commands, if you are able to connect to the virsh console without any errors then virsh is able to connect to the KVM hypervisor.

Listing 2.32: Verify that virsh can Access KVM

```
$ su
$ virsh -c qemu:///session
```

3. Next, to ensure that KVM is properly configured and installed, follow this guide provided on the CernVM website http://cernvm.cern.ch/portal/kvm **except, do not create a kvm definition file as the xml template file is provided by the test suite scripts** and verify that you are able to connect to the libvirtd kvm system session.

Listing 2.33: Verify that KVM is Properly Configured

```
$ su
$ virsh -c qemu:///system
```

4. Finally, ensure that you are able to connect to the QEMU/KVM hypervisor using the virtual machine manager, as it is necessary to view the status of the CernVM images and must be installed to troubleshoot and view the CernVM images. Simply launch the virtual machine manager application from `Applications -> System Tools -> Virtual Machine Manager` and**if you are prompted to install libvirt, select "No"** as a custom version of libvirt was installed previously.

2.3.5 Installing and configuring VirtualBox

1. First, begin by downloading and installing a version of VirtualBox supported by libvirt from the VirtualBox download page, it is best to download the latest version within the series *that has been available for at least a month prior to the release of the version of libvirt installed.* VirtualBox can be downloaded from the following location, http://www.virtualbox.org/wiki/Downloads ensure that you select the appropriate Debian based distribution, version and architecture for your system. The following instructions for this section of the guide uses VirtualBox 4.0.12 for Debian Squeeze, AMD64.

2. Before installing VirtualBox, install the dependencies for executing VirtualBox as well as the dependencies required to build the VirtualBox kernel modules.

Listing 2.34: Install VirtualBox Dependencies

```
$ su
$ apt-get install libqt4-network libqt4-opengl libqtcore4 libqtgui4
$ apt-get install linux-headers-$(uname -r)
```

3. Next, after downloading the latest version of VirtualBox for your distribution and installing the dependencies install VirtualBox as the root account using the following command.

4. Next, after downloading the latest version of VirtualBox for your distribution install VirtualBox as the root account using the following command.

Listing 2.35: Install VirtualBox Dependencies

```
# Enter the root password when prompted
$ su
$ dpkg -i virtualbox-*.deb
```

5. Next, in order to use VirtualBox and have full access to the drivers needed for USB support, ensure that the root account belongs to the group "vboxusers" using the following command.

Listing 2.36: Add root to VirtualBox Group

```
$ su -c "usermod -a -G vboxusers root"
```

6. Due to an issue with VirtualBox[7], in order for it to work with virsh the virtual machine(s) must be created and configured as the root account, otherwise when you try to connect or start a VirtualBox virtual machine with virsh you will get an "unknown error", which is obviously very vague and difficult to resolve. **Therefore ALWAYS start VirtualBox as the root account using the following procedure**.

Listing 2.37: Always Start VirtualBox as Root

```
# Switch to the root account, enter root password
$ su

# Start VirtualBox as root
$ virtualbox
```

7. Finally, verify that VirtualBox has been installed properly and that virsh can connect to the VirtualBox hypervisor, verify that the VirtualBox module, *vboxdrv* has been loaded and that you are able to connect to the virsh console without any errors.

Listing 2.38: Verify that virsh can Access VirtualBox

```
$ su

# Verify that the vboxdrv module is loaded
```

[7] The issues is that VirtualBox looks for virtual machine configuration files (*.vbox) in the "VirtualBox VMs" folder of the user that launched VirtualBox. The issue is worsened by the fact that there can only be one "VirtualBox VMs" folder which causes conflicts with multiple users.

```
$ lsmod | grep -i vboxdrv

# Verify that virsh can connect to virtualbox
$ virsh -c vbox:///session
```

2.3.6 Installing and configuring VMware

1. First, begin by downloading the latest version of VMware Workstation or VMware Player from the VMware product page, http://www.vmware.com/products/, VMware Player is free, whereas VMware Workstation requires a license. So if you decide to use VMware Workstation instead of VMware Player you will have to purchase a license for it in order to continue.

2. Before installing VMware, install the dependencies required to build the VMware kernel modules.

   Listing 2.39: Install VMware Dependencies

   ```
   $ su
   $ apt-get install linux-headers-$(uname -r)
   ```

3. Next, to install VMware simply set the bundle file as executable and execute the file as root.

   Listing 2.40: Install VMware

   ```
   $ su
   $ chmod +x VMware*.bundle
   $ ./VMware*.bundle
   ```

4. Next, launch VMware as root and wait for it to compile the kernel modules, then verify that the following VMware kernel modules are loaded, currently virsh has support to connect to the VMware hypervisor, but there are some minor issues such as a lack of support for VMware network configurations, currently only "bridged" mode is supported.

   - vmnet
   - vmblock
   - vmci
   - vmmon

   Listing 2.41: Verify VMware Kernel Extensions Loaded

   ```
   # Launch VMware as root, this will build kernel modules
   $ su
   $ vmware
   ```

```
# Verify that the kernel extentsions are loaded
$ lsmod | grep -i vm
```

5. Finally, verify that VMware has been configured properly and that virsh supports the current version of VMware installed by connecting to the virsh console for the VMware hypervisor.

Listing 2.42: Verify VMware Works with Virsh

```
# Verify that virsh can connect to vmware
$ virsh --connect vmwarews:///session
```

6. If everything so far has worked, then libvirt, virsh, and the hypervisors have been installed and configured properly, if you have any outstanding issues solve them before proceeding further, or go to the section "Server Platform Setup" 3 as the Tapper server does not require libvirt, virsh, or hypervisor configuration.

2.3.7 Installing and configuring Xen

1. At the moment Xen is not natively supported on Red Hat based platforms, therefore the instructions provided are specific to Debian Squeeze and currently Debian provides the only option for testing CernVM Xen images. The following instructions are specific to Debian Squeeze, Xen support on Lenny is much more difficult and only support for Xen 3.x, which is much older than the latest version of Xen which is provided by default on Debian Squeeze. The following instructions are for Xen 4.0, which is currently the latest version of Xen.

2. **Before proceeding any further ensure that Xen is not installed on a system which has any of the other hypervisors installed as it will cause severe instability**. It is best to create a new installation for the Xen hypervisor only, with libvirt installed as installing Xen on a system with any other hypervisor installed causes severe instability and either makes the system kernel panic or causes other failures.

3. Now that you have a dedicated system for Xen, begin by installing the Xen package and Xen Linux kernel with dom0 support.

Listing 2.43: Installing Xen Package and Kernel

```
$ su
$ apt-get install xen-linux-system
```

4. Next, install the packages required by Xen for Hardware Virtual Machine (HVM) which is required to run the CernVM image.[8]

[8] This is because the CernVM image for Xen currently requires a fully virtualized environment, (HVM) rather than a paravirtualized environment

Listing 2.44: Installing Packages for HVM Support

```
$ su
$ apt-get install xen-qemu-dm-4.0
```

5. Next, install the following tools for Xen, which make managing virtual machine and interfacting with the Xen dom0 hypervisor much easier.

Listing 2.45: Installing Xen Tools

```
$ apt-get install xen-watch xen-tools
```

6. Next, configure GRUB to support booting from the Xen kernel by default, otherwise you may reboot and determine the Xen entry from the boot menu and then modify the value of GRUB_DEFAULT in the file /etc/default/grub. The easiest method is provided, which is to simply boot the Xen kernel by default, **please note that you may not be able to run or test CernVM images on any hypervisor other than Xen if you boot the Xen kernel**. If you need to use KVM/VMware/VirtualBox then you will have to use a separate system or installation and boot the regular Linux kernel instead of the Xen kernel.

Listing 2.46: Configure Booting the Xen Kernel

```
# Set Xen kernel to boot by default
$ su
$ mv /etc/grub.d/10_linux /etc/grub.d/50_linux
```

7. Next, before rebooting the system, configure Xen to shutdown the virtual machines when the host system shuts down instead of attempting to save the state of the virtual machine. Edit the file `/etc/default/xendomains` and configure the two settings XENDOMAINS_RESTORE and XENDOMAINS_SAVE as follows.

Listing 2.47: Configure Xen Virtual Machine Shutdown

```
$ su -c "gedit /etc/default/xendomains"

# Configure the settings in the file as follows
XENDOMAINS_RESTORE=false
XENDOMAINS_SAVE=""
```

8. Next, enable a bridged network to the Xen virtual machine by editting the file `/etc/xen/xend-config.sxp` and uncommenting the lines **(network-script network-bridge)** and **(vif-script vif-bridge)**.

9. Next, configure the memory for the Xen dom0 and disable dom0 memory ballooning, the minimal memory set for this guide is 1024, but you may increase or lower the amount, 512 should still be an acceptable amount.

Listing 2.48: Configure Xen Memory Use

```
$ su -c "gedit /etc/xen/xend-config.sxp"

# Uncomment and configure the following options
(dom0-min-mem 1024)
(enable-dom0-ballooning no)
```

10. In addition to configuring the the memory for the Xen dom0 in the xend configuration file, the Xen dom0 minimal memory must also be set as a GRUB boot paramater, *otherwise dom0 will use all available memory making it impossible to boot virtual machines.* The memory set for this guide is 1024, but you may increase or lower the amount, 512 should still be an acceptable amount.

Listing 2.49: Configure Xen Memory Use as GRUB Boot Paramater

```
# Edit /etc/default/grub
$ su
$ gedit /etc/default/grub
# Add the following line to the grub configuration file
# Xen boot parameters for all Xen boots
GRUB_CMDLINE_XEN="dom0_mem=1024M"

# update grub with new configurations
$ update-grub
```

11. Finally, enable the Xen unix daemon, which is a mandatory requirement for executing tests and using libvirt with Xen. Simply uncomment the following line, **(xend-unix-server no)** and change the value to **yes**.

12. Now that Xen has been properly configured reboot the system and wait for it to boot the Xen kernel with dom0 support, it may take several minutes for the system to boot Xen. **Please note, that if you intend on using any of the other hypervisors you must use a completely separate system or installation**.[9].

13. After the system has booted, first verify that the Xen kernel has loaded properly and that the Xen kernel has booted correctly with Domain-0 (dom0) support, which is required in order for the Xen hypervisor to function. [10] **If you receive any errors for the follow procedure try waiting 5 minutes for the xen daemon (xend) to start, it can often take several minutes to start**.

[9] This is because the Xen dom0 and kernel conflicts with the other hypervisors
[10] Booting the Xen kernel does nothing unless the kernel has booted with Xen dom0 support

Listing 2.50: Verify Xen Booted Correctly

```
# Verify the kernel, it should contain "xen"
$ su
$ uname -r

# Verify that Xen has Domain-0 running
$ xm list
```

14. Finally, if Xen kernel has booted correctly and the system is also running with Xen Domain-0 support, verify that it is possible to connect to the Xen hypervisor with virsh. If virsh is able to connect to the Xen hypervisor then the Xen dom0, Domain-0 should be seen running by executing"list –all".

Listing 2.51: Verify virsh can Connect to Xen Hypervisor

```
# Connect to Xen hypervisor
$ su
$ virsh -c xen:///

# Domain-0 must be running for Xen support
$ list --all
```

2.3.8 Setting up the Tapper Test Suite

1. **Before proceeding any further ensure that you have all other test clients set up this far, and then proceed to follow the instructions for setting up and configuring the Tapper server in the section "Server Platform Setup"** 3.

2. Now that the Tapper server has been installed and configured and the Tapper web interface and database have proven to be working, the next step is to verify that the test client can actually send a report to the Tapper server in the form of a TAP file. After sending the TAP report to the server, ensure that the test client is working by viewing the tapper reports in your browser at the following url: `http:/<tapper_server>/tapper/reports`. You should now see a report from the test client, there should be a report from a system named whatever the "Tapper-Machine-Name" in demo_report.tap was set as. *For the example demo_report.tap provided below it would be cernvm-rhtestclient.* [11].

Listing 2.52: Send a Basic Report to the Tapper Server

```
# Save the following in a file named demo_report.tap

1..2
```

[11] This is why a consistent hostname convention was emphasized earlier, as reports are often sorted and organized based on hostnames

```
    # Tapper-Suite-Name: Tapper-Deployment
    # Tapper-Suite-Version: 1.001
    # Tapper-Machine-Name: cernvm-rhtestclient
    ok - Hello World
    ok - Just another description

# Send the report to the tapper server using netcat
$ cat demo_report.tap | nc -w10 cernvm-server 7357
```

3. Next, download a copy of the CernVM Test Suite and the CernVM Test Cases from the Google Code svn repository [1] and install the the following dependencies.

Listing 2.53: Install CernVM Test Suite and Dependencies

```
# Install subversion, required to checkout auto-tapper
$ yum install subversion

# Checkout a copy of cernvm testsuite and cernvm testcases
$ svn checkout http://cernvm-release-testing.googlecode.com/svn/\
trunk/tapper/tapper-autoreport/ cernvm-testsuite

# Install the missing dependencies
$ apt-get install curl
$ apt-get install wget
$ apt-get install uuid
$ apt-get install spawn-fcgi
$ apt-get install expect
$ apt-get install expectk
$ apt-get install nmh
```

4. Now that cernvm-testsuite has been installed, configure the variables in the configuration file for the hypervisors you want to test and according to your Tapper infrastructure setup. Sample configuration files are provided in the **config** folder, all of the settings provided in the configuration files by default are the **mandatory, minimum configuration options** and in most cases the defaults should be sufficient for testing, the only mandatory settings that are not provided by default **and must be configured manually are CVM_TS_REPORT_SERVER and CVM_VM_IMAGE_VERSION** . Please refer to the developer manual for a more complete detailed list of the mandatory and optional configuration settings.

   CVM_TS_SUITENAME Must ALWAYS be set, the default suite name in the configuration file should be suitable

   CVM_TS_SUITEVERSION Must ALWAYS be set, reflects the release version number of the test suite framework, the default version given in the configuration should only be changed if you customize/update the scripts

CVM_TS_REPORT_SERVER Must ALWAYS be set, this is the ip address or hostname of the Tapper report server which the reports from the test results are sent to

CVM_TS_DOWNLOAD_PAGE Must ALWAYS be set, normally the default url provided in the configuration file is accurate, but in the event that the internal CernVM image release page is relocated then this url must be changed.

CVM_VM_HYPERVISOR Must ALWAYS be set, should not have to change the defaul hypervisor in the configuration files, valid values (case sensitive) are **kvm,vbox,vmware**

CVM_VM_TEMPLATE Must ALWAYS be set, normally the default template provided in the configuration file should not be changed, only change this to use a custom template file. The custom template file *must be placed within the templates folder*

CVM_VM_NET_TEMPLATE Must ALWAYS be set, normally the default network template provided in the configuration file should not be changed, only change this to use a custom network template file for the CernVM image, **only applies to kvm and virtualbox**. The custom network template file *must be placed within the templates folder*

CVM_VM_IMAGE_VERSION Must ALWAYS be set, specifies the version of the CernVM image to use from the release page

CVM_VM_IMAGE_TYPE Must ALWAYS be set, specifies the type of CernVM image, valid image types supported, (case sensitive) are **basic and desktop**

CVM_VM_ARCH Must ALWAYS be set, specifies the architecture of the CernVM image, valid architectures (case sensitive) are **x86 and x86_64**

5. Finally, now that cernvm-testsuite has been installed and configured on the test client and the test client and Tapper Server have proven to be working, the next step is to verify that cernvm-testsuite works correctly and can actually send a report to the Tapper server in the form of a TAP file. To execute the CernVM Test Cases script, "cernvm-tests.sh" simply source the configuration file you wish to use and execute the script. Once the script has completed and sent a TAP report to the server, ensure that the test client is working by viewing the tapper reports in your browser at the following url: `http:/<tapper_server>/tapper/reports`. You should now see a new report from the test client, there should be a report from a system with the same hostname.[12].

Listing 2.54: Execute CernVM Test Cases Script

```
# Simply source the configuration file, and execute the script
```

[12] This is why a consistent hostname convention was emphasized earlier, as reports are often sorted and organized based on hostnames

```
$ . ./config/<configuration_file>
$ ./cernvm-tests.sh
```

2.4 OS X Test Client Setup

2.4.1 Configuring the system

1. After the system has booted, the first thing to configure are the power management settings and other preferences, as this system will be running as a test client, sleep and other automatic energy saving features must be disabled. Begin by navigating to the power options,
   `Apple logo -> System Preferences -> Energy Saver` for the option "Computer sleep" slide the bar to the far right so that it is set to "Never" and ensure that the following options are all disabled.
   - Put the hard disk(s) to sleep when possible
   - Wake for Ethernet network access
   - Allow power button to put the computer to sleep
2. Next, set a hostname for the system from the menu
   `Apple logo -> System Preferences -> Sharing` beside the "Computer Name:" option at the top, click the "Edit..." button. Then enter a relevant hostname for the system based on the hardware or operating system it is running; the hostname should be relevant and unique to better identify the system. A good naming convention should refer to the hardware or operating system and call it a host to differentiate from the virtual machine that will be running as a guest, for example a hostname such as *cernvm-osx-host* could be used, **whatever convention you use make sure it is consistent**.
3. Next, enable SSH access to the system by navigating to
   `Apple logo -> System Preferences -> Sharing` and from the list of services that can be shared, enable "Remote Login", which is SSH.
4. Now, to enable VNC access to the system, select from the same list of services that can be shared, "Remote Management" and for local access options window that appears, enable all of the options listed such as "Observe" and "Change settings". Then to enable VNC compatibilty so that the OS X system can be accessed by other non-Apple computers, click the "Computer Settings..." button and enable the following options and set a password for the "VNC viewers..." option.
   - Show Remote Management status in menu bar
   - Anyone may request permission to control screen
   - VNC viewers may control screen with password
5. Now, to ensure that your user logs in automatically, navigate to
   `Apple logo -> System Preferences -> Accounts` and click "Login Options", you may have to click on the lock icon and enter your password in order to make changes to the login options. Then for the option "Automatic login:" select your user from the list of accounts to enable automatic login.

6. Finally, to ensure that the settings were configured properly, reboot the machine and ensure that the following work.
   - It automatically boots up into the full desktop environment without having to login
   - You have access to the machine using SSH and can login
   - You have VNC access to the machine and can control the system using VNC

2.4.2 Installing libvirt and virsh

1. The virtualization API libvirt and the command line tool virsh [3] are the essential components required for setting up a test client and must be installed and properly configured before any testing can begin. Ensure that you follow the proceeding directions carefully and validate that virsh is working properly before proceeding to install and configure the various hypervisors.

2. First, to install libvirt, begin by installing Homebrew[13] using the following command, for a more detailed installation guide refer to the Homebrew wiki https://github.com/mxcl/homebrew/wiki/Installation.

   Listing 2.55: Install Homebrew

   ```
   # Install Homebrew using the following command
   /usr/bin/ruby -e "$(curl -fsSL https://raw.github.com/gist/323731)"
   ```

3. Next, download and install Xcode, which is freely available at http://developer.apple.com/xcode/. At the moment Xcode 3 can be downloaded for free, whereas Xcode 4 costs money and must be purchased from the Apple App store[14], this guide uses the freely available Xcode 3, which at the time of writing is version 3.2.6.

4. Next, to install libvirt simply use the following command, please note, in the unlikely event that there has not been an updated "formula" to install a newer version of libvirt, then you can attempt to either create one or update the current libvirt formula and benefit yourself and others who may install libvirt using Homebrew. The following are instruction for creating a formula https://github.com/mxcl/homebrew/wiki/Formula-Cookbook.

   Listing 2.56: Install libvirt

   ```
   # Install libvirt using the following command
   brew install libvirt
   ```

[13] Homebrew is a package manager for OS X, which is similar to apt, it will be used to install dependencies instead of manually building from source

[14] There is still one catch, in order to download Xcode 3, which is free, you must first make an Apple developer account

5. Finally, ensure that virsh installed correctly and is running by connecting to the test hypervisor and ensuring that the test virtual machine, named "test" is running.

Listing 2.57: Verify virsh was Installed Properly

```
# Verify virsh is working, test should be running
$ virsh -c test:///default list --all
```

2.4.3 Installing and configuring VirtualBox

1. First, begin by downloading and installing a version of VirtualBox supported by libvirt for OS X from the VirtualBox download page, it is best to download the latest version within the series *that has been available for at least a month prior to the release of the version of libvirt installed.* VirtualBox can be downloaded from the following location, http://www.virtualbox.org/wiki/Downloads ensure that you select the appropriate architecture for your system. The following instructions for this section of the guide uses VirtualBox 4.0.10 for AMD64.

2. Next, to verify that VirtualBox has been installed properly and that virsh can connect to the VirtualBox hypervisor, verify that the following VirtualBox kernel extensions are loaded.

   - org.virtualbox.kext.VBoxDrv
   - org.virtualbox.kext.VBoxUSB
   - org.virtualbox.kext.VBoxNetFlt
   - org.virtualbox.kext.VBoxNetAdp

Listing 2.58: Verify that VirtualBox Kernel Exentsions are Loaded

```
# Verify that the kernel extentsions are loaded
$ kextstat | grep -i virtualbox
```

3. Finally, verify that VirtualBox has been configured properly and that virsh supports the current version of VirtualBox installed by connecting to the virsh console for the VirtualBox hypervisor.

Listing 2.59: Verify VirtualBox Works with Virsh

```
# Verify that virsh can connect to virtualbox
$ virsh --connect vbox:///session
```

2.4.4 Installing and configuring VMware

1. First, begin by downloading and installing the latest version of VMware Fusion for OS X from the VMware product page, http://www.vmware.com/products/, VMware Fusion requires a license, so you will have to purchase it in order to continue.

2. Next, to verify that VMware Fusion has been installed properly, verify that the following VMware kernel extensions are loaded, currently virsh has support to connect to the VMware hypervisor and has only recently supported connecting to VMware Fusion since I added support in version 0.9.5 of libvirt.

   - com.vmware.kext.vmx86
   - com.vmware.kext.vmci
   - com.vmware.kext.vmioplug
   - com.vmware.kext.vmnet

   Listing 2.60: Verify VMware Kernel Extensions Loaded

   ```
   # Verify that the kernel extentsions are loaded
   $ kextstat | grep -i vmware
   ```

3. Next, add VMware Fusion to the system PATH variable permanently so that VMware Fusion can be executed from the command line as well as accessed by virsh.

   Listing 2.61: Add VMware Fusion to PATH

   ```
   # Add VMware Fusion to PATH
   $ echo 'export PATH=/Library/Application\ Support/VMware\ Fusion/:$PATH' >> ~
   $ echo 'export PATH=/Applications/VMware\ Fusion.app/Contents/MacOS:$PATH' >>

   # Source the file, set PATH variable
   $ . ~/.profile
   ```

4. Finally, verify that VMware Fusion has been configured properly and that virsh supports the current version of VMware Fusion installed by connecting to the virsh console for the VMware Fusion hypervisor.

   Listing 2.62: Verify VMware Fusion Works with Virsh

   ```
   # Verify that virsh can connect to vmware fusion
   $ virsh --connect vmwarews:///session
   ```

2.4.5 Setting up the Tapper Test Suite

1. **Before proceeding any further ensure that you have all other test clients set up this far, and then proceed to follow the instructions for setting up and configuring the Tapper server in the section "Server Platform Setup"** 3.

2. Now that the Tapper server has been installed and configured and the Tapper web interface and database have proven to be working, the next step is to verify that the test client can actually send a report to the Tapper server in the form of a TAP file. After sending the TAP report to the server, ensure that the test client is working by viewing the tapper reports in your browser at the following url: `http:/<tapper_server>/tapper/reports`. You should now see a report from the test client, there should be a report from a system named whatever the "Tapper-Machine-Name" in demo_report.tap was set as. *For the example demo_report.tap provided below it would be cernvm-osxtestclient.* [15].

Listing 2.63: Send a Basic Report to the Tapper Server

```
# Save the following in a file named demo_report.tap

        1..2
        # Tapper-Suite-Name: Tapper-Deployment
        # Tapper-Suite-Version: 1.001
        # Tapper-Machine-Name: cernvm-osxtestclient
        ok - Hello World
        ok - Just another description

# Send the report to the tapper server using netcat
$ cat demo_report.tap | nc -w10 cernvm-server 7357
```

3. Next, download a copy of the CernVM Test Suite and the CernVM Test Cases from the Google Code svn repository [1] and install the the following dependencies.

Listing 2.64: Install CernVM Test Suite

```
# Install subversion, required to checkout auto-tapper
$ brew install subversion

# Checkout a copy of auto-tapper and cernvm testcases
$ svn checkout http://cernvm-release-testing.googlecode.com/svn/\
trunk/tapper/tapper-autoreport/ cernvm-testsuite
```

[15] This is why a consistent hostname convention was emphasized earlier, as reports are often sorted and organized based on hostnames

4. Now, install the following dependencies using Homebrew[16].

Listing 2.65: Install Dependencies

```
# Update Homebrew
$ brew update

# Install the following dependencies
$ brew install bash
$ brew install bash-completion
$ brew install ssh-copy-id
$ brew install wget
$ brew install ossp-uuid
$ brew install spawn-fcgi
$ brew install md5sha1sum
$ brew install curl
```

5. Next, set the default terminal as the newer version of bash that was installed with Homebrew and then verify that the new version of bash has been properly configured as the default terminal.

Listing 2.66: Configure Bash

```
# Change to root account, enter password if prompted
$ sudo su

# Add the new bash shell to the list of acceptable shells
$ echo '/usr/local/bin/bash' >> /etc/shells

# For any account that the script will be executed on
# set the default shell as the new bash shell installed
$ chsh -s /usr/local/bin/bash

# Verify the new version of bash has been configured, check
# the version of bash installed matches "brew info bash" output
$ echo $BASH_VERSION
```

6. Now that cernvm-testsuite has been installed, configure the variables in the configuration file for the hypervisors you want to test and according to your Tapper infrastructure setup. Sample configuration files are provided in the **config** folder, all of the settings provided in the configuration files by default are the **mandatory, minimum configuration options** and in most cases the defaults should be sufficient for testing, the only mandatory settings that are not provided by default **and must be configured manually are**

[16] This section is extremely short in comparison to the previous instructions for this section which required building the dependencies from source

CVM_TS_REPORT_SERVER and CVM_VM_IMAGE_VERSION . Please refer to the developer manual for a more complete detailed list of the mandatory and optional configuration settings.

CVM_TS_SUITENAME Must ALWAYS be set, the default suite name in the configuration file should be suitable

CVM_TS_SUITEVERSION Must ALWAYS be set, reflects the release version number of the test suite framework, the default version given in the configuration should only be changed if you customize/update the scripts

CVM_TS_REPORT_SERVER Must ALWAYS be set, this is the ip address or hostname of the Tapper report server which the reports from the test results are sent to

CVM_TS_DOWNLOAD_PAGE Must ALWAYS be set, normally the default url provided in the configuration file is accurate, but in the event that the internal CernVM image release page is relocated then this url must be changed.

CVM_VM_HYPERVISOR Must ALWAYS be set, should not have to change the defaul hypervisor in the configuration files, valid values (case sensitive) are **kvm,vbox,vmware**

CVM_VM_TEMPLATE Must ALWAYS be set, normally the default template provided in the configuration file should not be changed, only change this to use a custom template file. The custom template file *must be placed within the templates folder*

CVM_VM_NET_TEMPLATE Must ALWAYS be set, normally the default network template provided in the configuration file should not be changed, only change this to use a custom network template file for the CernVM image, **only applies to kvm and virtualbox**. The custom network template file *must be placed within the templates folder*

CVM_VM_IMAGE_VERSION Must ALWAYS be set, specifies the version of the CernVM image to use from the release page

CVM_VM_IMAGE_TYPE Must ALWAYS be set, specifies the type of CernVM image, valid image types supported, (case sensitive) are **basic and desktop**

CVM_VM_ARCH Must ALWAYS be set, specifies the architecture of the CernVM image, valid architectures (case sensitive) are **x86 and x86_64**

7. Finally, now that cernvm-testsuite has been installed and configured on the test client and the test client and Tapper Server have proven to be working, the next step is to verify that cernvm-testsuite works correctly and can actually send a report to the Tapper server in the form of a TAP file. To execute the CernVM Test Cases script, "cernvm-tests.sh" simply source the configuration file you wish to use and execute the script. Once the script has completed and sent a TAP report to the server, ensure that the test client is working by

viewing the tapper reports in your browser at the following url: http:/<tapper_server>/tapper/reports. You should now see a new report from the test client, there should be a report from a system with the same hostname.[17].

Listing 2.67: Execute CernVM Test Cases Script

```
# Simply source the configuration file, and execute the script
$ . ./config/<configuration_file>
$ ./cernvm-tests.sh
```

[17] This is why a consistent hostname convention was emphasized earlier, as reports are often sorted and organized based on hostnames

3 CernVM Release Testing Server Platform Setup

3.1 Introduction

This section provides complete step by step instructions on how to setup and configure the TAPPER server which is part of a basic working RELEASE TESTING environment by outlining the procedure for setting up the server, hence why this is called a *walkthrough* document. This guide is intended for developers, for a more complete reference guide to installing and configuring the TAPPER server please refer to the regular walkthrough document.

3.2 Debian Based Server Setup

For installing and configuring a Debian based server, follow the instructions outlined in the sections "Configuring the system" 2.3.2 for installation and configuration instructions with the only exception being that the hostname should be something unique such as *cernvm-debian6-server*, to indicate that it is running the Tapper server, **again keep the hostname convention consistent**.

3.2.1 Installing the Tapper Server

1. Next, execute the following commands to install necessary dependencies, *from now on all commands require root privileges.*

   Listing 3.1: Install Dependencies

   ```
   $ apt-get update
   $ apt-get install make
   $ apt-get install subversion
   ```

2. Now, download the latest copy of the Tapper-Deployment, which is an installer for Tapper from the CernVM Release Testing Google Code Project page

   Listing 3.2: Download Tapper-Deployment

   ```
   $ svn checkout http://cernvm-release-testing.googlecode.com/svn/trunk/\
   installer/tapper-deployment
   ```

3. Now edit the Makefile in the Tapper-Deployment installer folder and configure variable TAPPER_SERVER which is the hostname of the machine that is currently installing the starter-kit. For now disregard the TESTMACHINE variables, you should have something similar to this in the Makefile.

   Listing 3.3: Makefile Configuration

   ```
   # initial machine names
   TAPPER_SERVER=cernvm-server
   TESTMACHINE1=johnconnor
   TESTMACHINE2=sarahconnor
   TESTMACHINE3=bullock
   ```

4. After you have configured the Makefile in the installer folder, install Tapper-Deployment by executing the following command, for any prompts during the installation leave them as default and press enter. **During the installation, you will be prompted for the mysql password, DO NOT ENTER a password here UNLESS you already have an existing MySQL installation/database with a password set for the "root" account**. Finally, if you have any errors or other issues during the installation, please contact us with a summary of the problem and send us a copy of the installation log "install.log".

Listing 3.4: Install Tapper-Deployment

```
$ cd installer/
$ make localsetup 2>&1 | tee install.log
```

3.2.2 Setting up Tapper Web Interface and Database

1. Next you need to set a password for the root account of the mysql database[1]

   Listing 3.5: Set MySQL Root Password

   ```
   # Example: mysqladmin -u root password abc123
   $ mysqladmin -u root password <newpassword>
   ```

2. Now that the installion has completed and the security issue has been dealt with, ensure that you can access the tapper web interface and that it is working by viewing it in your browser using the url, http://localhost/tapper[2]

3. Next, install PHPMyAdmin so that it's easy to administrate and configure the Tapper databases, when prompted to select the "Web server to reconfigure automatically" select apache2 by pressing the space bar and press enter. *If you are prompted to "configure databases for phpmyadmin with dbconfig-common" select* **NO** .

   Listing 3.6: Install PHPMyAdmin

   ```
   $ apt-get update

   # When prompted for the server to reconfigure automatically select apache2
   # when prompted to configure the database with dbconfig-common select NO
   $ apt-get install phpmyadmin
   ```

4. Verify that PHPMyAdmin has been installed and configured correctly and that you can access the tapper web interface by viewing it in your browser using the url, http://localhost/phpmyadmin. Login to the PHPMyAdmin web interface using the *username root and the MySQL root password you set earlier using mysqladmin.*

5. Now, add all of the configured test machines created in the "Test Client Platform Setup" section to the Tapper database and set the test clients as active, then add the hardware specifications for each test client to the database. This example is just using a single generic test machine, you will have to repeat these commands for each test client and change the hostname *cernvm-host* and the values for mem, core,vendor, and has_ecc as needed; the vendor can be AMD or Intel.

[1] This will eventually be implemented in the makefile

[2] This can be accessed locally and remotely from other systems using the server hostname or IP address

Listing 3.7: Adding Test Clients to Tapper Database

```
# Add the hostname of the test client to database
$ tapper-testrun newhost --name cernvm-host --active

# Add the hardware specifications for the test client
$ mysql testrundb -utapper -ptapper
$ insert into host_feature(host_id, entry, value) values \
((select id from host where name = 'cernvm-host'), 'mem',
4096);
$ insert into host_feature(host_id, entry, value) values \
((select id from host where name = 'cernvm-host'), 'cores',
4);
$ insert into host_feature(host_id, entry, value) values \
((select id from host where name = 'cernvm-host'), 'vendor', 'AMD');
$ insert into host_feature(host_id, entry, value) values \
((select id from host where name = 'cernvm-host'), 'has_ecc',
0);
```

6. Next, send a sample test report to the tapper server, to ensure that the web interface, MCP, database, and reports framework are all working by viewing the tapper reports in your browser at the following url, http://localhost/tapper/reports You should now see a report from whatever the "Tapper-Machine-Name" in demo_report.tap was set as. *For the example demo_report.tap provided below it would be cernvm-server.*

Listing 3.8: Send a Report from the Tapper Server to Itself

```
# Save the following in a file named demo_report.tap
$ vi demo_report.tap

        1..2
        # Tapper-Suite-Name: Tapper-Deployment
        # Tapper-Suite-Version: 1.001
        # Tapper-Machine-Name: cernvm-server
        ok - Hello test world
        ok - Just another description

# Send the report to the tapper server using netcat
$ cat demo_report.tap | netcat -q7 -w1 cernvm-server 7357
```

7. Finally, ssh login to one of the test machine that was set up earlier, *in our examples, cernvm-host* and send another sample test report to the tapper server, to ensure that the web interface, MCP, database, and reports framework are all working by viewing the tapper reports in your browser at the following url: http://localhost/tapper/reports. You should now see a report from

whatever the "Tapper-Machine-Name" in demo_report.tap was set as. *For the example demo_report.tap provided below it would be cernvm-testclient.*

Listing 3.9: Send a Report to the Tapper Server from a Test Client

```
# Save the following in a file named demo_report.tap
$ vi demo_report.tap

        1..2
        # Tapper-Suite-Name: Tapper-Deployment
        # Tapper-Suite-Version: 1.001
        # Tapper-Machine-Name: cernvm-testclient
        ok - Hello test world
        ok - Just another description

# Send the report to the tapper server using netcat
$ cat demo_report.tap | netcat -q7 -w1 cernvm-server 7357
```

8. Now that it has been verified that the tapper server, including the web interface, MCP, database, and reports framework are all working; return to the sections titled "Setting up the Tapper Test Suite" for each of the test client, as there are unique instructions for each operating system.

Bibliography

[1] CernVM Release Testing Google Code Project. https://code.google.com/p/cernvm-release-testing/.

[2] Advanced Micro Devices Inc. AMD Tapper. http://developer.amd.com/zones/opensource/amdtapper/pages/default.aspx/, 2011.

[3] Red Hat Inc. libvirt Virtualization Api. http://libvirt.org/, 2011.

Index