Building, deploying, securing and managing your own crypto-currency mining

Web Front End

an elaborate introduction

A Step by Step approach utilizing the Ubuntu Linux Operating System

Illustrated with MPOS, Mining Portal Open Source

Scripted, from the Ground Up

Edition 1

April, 2014

Steven J. Martin

M.S. MIS, MBA, B.S. MIS

Whether your interested in building a successful shared mining pool, or interested in the technology, or maybe your interested in keeping all of the coins you actually mine; I think you can learn something from this book. This is a step by step approach, from a bare-metal Linux installation to an operational web based shared mining pool

using the same approach, and consequently … likely, the same software professional mining sites use. Every step in this book is explained in detail, organized by chapter. Each step of each chapter details the commands necessary for success, and each chapter additionally includes a complete setup script for automation, and reuse.

With the contents of this book, you will be able to create a web based portal under the Apache SSL, MySQL, and MPOS application suites, and be mining “your own” coins and keep 100% of them in your wallet. The first coin we mine is Litecoin. Then it's on to Dogecoin. From there we set up for infinitecoin, and finally, we go international with eMark. Each of these coin bases can run on a separate, or the same server. Each is configured with their own database, and each uses it's own Stratum. Having the scripts necessary to accomplish this, and of course, the proper procedures, makes these tasks easy and repeatable. By establishing one site, then “switching it up” to an alternate coin, the book reinforces the small configuration changes necessary, and builds the knowledge necessary to be a successful Internet entrepreneur, and owner of you very own public, or if you prefer, private pooled coin mining web site.

If you are interested in solo mining, then this is the way to accomplish the task. Solo mining without variable difference is near impossible because of the high network difficulty. This is why so many recommend pooled mining. A pooled mining site includes the very software we are speaking about here. A mining site with workers, variable differences, pools, rounds, shares, and found blocks. You may be able to find all of the information detailed within these pages somewhere amongst the words strewn about the Internet. But I am unsure that you will find a more concise, step by step approach as within this book you are now perusing … your go to reference.

Have a look at the table of contents, then decide; do I want to be the purveyor of an Internet based mining site? Do I want to setup a shared mining pool for my close associates? Or do I simply wish to keep 100% of what I mine in my own wallet. Chances are, based on your answer, some, if not all of the steps necessary are right here. If for nothing other that personal mining, to actually keep the coins you mine, then this book,will pay for itself likely on the first operational day. If you decide that you would like to pursue becoming an Internet entrepreneur, then so be it. Using this book as a reference will certainly get you well on your way. If your desire is to establish a small private site for friends and family, then great!

From scrypt mining to sha256d, pow and pos, vardiff, stratum's, coin daemons, and clients, cpu and gpu, asics mining, twistd and config.py; It's in here waiting to be read and put to good use. Who knows what will be the future of crypt coins. Which will succeed, and which will fail. Yes it is true that it is nearly impossible to find a Bitcoin block these days, however there exists a plethora of alternatives, domestic and international ripe for the picking. In a 12 hour period, I personally solved an eMark (dem) block and banked 50 coins. In a 24 hour period, I found 9 infinitecoin (ifc) blocks that netted 1,200 coins for each block found. If I were mining on a pooled web site owned by another entrepreneur, I would have shared these with any number of other individuals, and would have been paid my shares, and maybe a block find bonus, plus I would have forked over 7.5 % of these to the purveyor of the site. Instead they now exist in my wallet.

So read on Internet entrepreneur, or solo miner, or private pool purveyor, or simply an interested reader. Read on and get your learn on. I encourage you to simply read on my friend.

An excerpt from Appendix “A”

Proof is sometimes in the pudding.

This next example is the infinitecoin. I was pleasantly surprised to find out that the infinitecoin block solution pays 1,200 coins per block found. This is a very easy coin to mine, yet has about the same potential return as the Dogecoin, which is rather difficult to mine. Below is a chart from MPOS showing expected normal PPLNS, (say from a normal Internet mining site,) and the actual return. The actual return is staggeringly higher than the expected.

Prologue

An Introduction

In my previous book also available as a Kindle download "*Automating the Installation, Configuration and Execution of Crypto Currency Mining Applications using Linux,*" I discuss the rudimentary aspects of establishing via automation, the mining rigs. Conversely, this book details the opposing end of that technology which is the mining server. The "mining server" provides the services of a pooled banking resource , much like a central bank, or the "banker in a Monopoly game." The public pooled mining web site is an application providing a shared pooling resource for individual miners. Much like the banker in Monopoly, your the banker, and as such, you are responsible for making payments based on your particular mining distribution topology.

Establishing a crypto-currency mining pool, or server if you will, can also be setup specifically for solo mining. It offers the flexibility of a public pooled mining resource such as variable difficulties, multiple workers, password protection, etc. right in your own back yard. Set it up and utilize it specifically as your own pooled mining server, or set it up to share with a small group. It's your mining site, so basically it is up to you. There are no transaction fees to pay so every share mined is your 100% to keep. That's good enough for me.

This book is a reference guide which provides a step by step, and scripted approach to establishing a pooled cryoto-currency mining site utilizing the Ubuntu Linux Operating System. It is illustrated using MPOS, "Mining Portal Open Source," a PHP based consortium and the most widely used pooled mining server software solution in service today. Scripting is embedded within this book which can also be downloaded. Links are provided within. We start with the base Linux Operating System platform, then work our way up the chain delving into technologies including MySQL, Apache Secure Server, PHP, the MPOS application, SendMail, Security Considerations and more. This is a no nonsense, "*soup to nuts,*" step by step and scripted approach. Yes, all of this information is obtainable somewhere on the Internet, possibly contained in several dozen or even hundreds of "posts," be them right or wrong. However here it is presented to you in a format that you can follow. Completely scripted, documented in a field proven topology.

This book starts with the fundamentals of installing the Ubuntu Debian Linux Server and journeys up the ladder, providing a ton of information. It's organized by chapter, and each chapter includes proven scripting for automating the task at hand. I like the MPOS documentation available on-line so very much that I wish to share a paragraph here:

" Before following this guide this warning is given as advice and a word of warning. Running a live pool is a job which requires in depth knowledge of pool code as well as the ability to debug and fix the pool. By NO means is this an in depth guide to running a pool and as such should only be used for private/testnet pools! As running a pool is a long and tiresome process where many different things can affect the stability and usability of the pool, it is guaranteed that problems will occur. "

This is very true, however there is so very much to learn simply by traversing the technology here within, and who knows, together we may actually start a lucrative and legitimate pooled mining web site.

As a personal note, if you are running a Mac or Windows OS, the information presented here may or may not be applicable to you. The instructions within this book are intended specifically for Linux, and to be even more specific, the Ubuntu/Debian distribution, and may or may not be applicable to other platforms and architectures.

A “bit” about the Author

The author of this book, and associated programs and executable automation scripts is currently game fully employed within the IT industry while his principle role is IT automation. He first started programming in the 1980's on a TI-994a a monochrome monitor and an Advanced Basic Cartridge.

After graduating college with a B.S. in MIS, he was employed working in IT on Large IBM mainframes coding RPG, Fortran, Cobol, and other so called “Mainframe” languages while working on his Masters. He graduated with a M.S. MIS, and and MBA in 1986 and has been consistently employed in some form of Computer Science even since.

Around 1990 he picked up UNIX for the first time and realized there is just no going back!

He is VERY much interested in educating younger people with an attitude and aptitude for computer science, and he believes it is the duty of “our generation” to help guide those young and eager individuals. His own thirst for knowledge and his willingness to share, is what this book is all about. He hopes you enjoy and profit from it.

The author would like to thank all of the individual contributors to the ever growing technical base available as open source for their hard work and diligence and in continuing to make this stuff fun to explore.

A “coin” of truth about Fonts

`currier font` indicates an executable script. This is actual executable code which can be used to accomplish the designated task.

Table of Contents

Table of Contents goes here.

Table of Contents

Chapter 1 The Base Operating System

Step One is installing the base Linux Operating System

Enter Linux

Prior to installing and or configuring anything one must have the Linux operating system installed on some form of a computer. Be it a laptop, a cabinet, a so called “pizza box,” really is of no consequence. To get started all we need is a computer that is applicable to the installation of the Linux operating system.

I have encountered absolutely NO problems with Ubuntu or Debian in performance and longevity, thus I personally prefer these distributions. Any Linux distribution is simple to get up and running and most will recognize other operating systems you may have currently installed and will setup a Dual Boot system for you, and as such, will allow you to first repartition your hard drive to make room for their install. This is a great way to prototype your environment. Or in other words, take it for a test drive, in a small, safe environment learning the ropes. Then when ready to go production, however a heftier “production” environment would be preferable with a stand alone server

platform running a hardened OS. This is fairly easily obtainable, and is within the confines of this book. Please do not get me wrong, setting up a production ready, hardened pooled mining server is a lot of work and requires a rather large degree of over site. However having said this, I assume you're already “down the rabbit hole,” so to speak, and ready for the challenge. So let's explore. There is nothing to lose, and all this technology waiting and wanting to be learned.

Installing the host Linux operating system is your responsibility and the deciding which distribution is yours alone to make. However this book makes the assumption that the base Linux distribution is Debian based, and the instructions and scripts herein are ONLY applicable to and or have been tested on Ubuntu Server or Desktop 13.10. The Ubuntu Server offers a fairly robust and secure base platform that is hardened to be utilized as an industrial server. This distribution is not a desktop, and is void of GUI, but offers a very suitable platform upon which to build a secure and reliable web hosting application and is the Authors recommendation.

One word of caution if you do prototype using a personal computer is to make sure you have a backup prior to installing anything on your. Installing Linux most definitely has the potential of overwriting your hard drive and eliminating everything that is currently written there. If you can, you should consider using a separate computer, or at least purchase another hard drive. Hard drives are so inexpensive and having a separate hard drive to install Linux on is simply the best choice given all of the available options.

Once you have determined the where and what you will install Linux on, not to mention the distribution, it is a relatively painless operation to actually install it. Several options are available. Most popular is to boot a Live CD from a compact disk you burn, or burn this very same Live CD onto a USB stick that is 1GB or greater in size. Either if these approaches are simple. The only problem with USB is I have found that not all computers offer to boot from USB, and even some of those that do have problems booting from USB. Another problem is that some of the Live CD distributions exceed the maximum allowable 800 MB of record able space of a standard CD. One can also use a DVD to burn a Live DVD. This eliminates both of the above two problems if you have a DVD burner and some extra writable DVDs.

Next determine if you are running a 32 bit or 64 bit compatible CPU. If you are running a 64 bit compatible CPU then I suggest you install the 64 bit Linux distribution. Most new CPU's are not only 64 bit compatible, they often are multi-processing cores. A quad core AMD CPU is perfect and around 100 dollars US. It fits nicely into a mini or micro motherboard combined with a moderately decent supply of RAM, say around 8 GB and a standard SATA hard drive of 500 GB to 1TB, and you have a perfect platform. A graphics card is on no consequence. Money would be better spent on memory or a nice Ethernet card as this equipment is designated to be a powerful, secure server, not a desktop PC.

This is the official Ubuntu server download URL for the latest 64 bit distribution. You can adjust it accordingly for desktop and or 32 bit distributions as you see fit by altering distro=desktop, or bits=32 as is demonstrated by the second URL.

Ubuntu 64 bit Sever URL

http://www.ubuntu.com/download/server/thank-you?distro=server&bits=64&release=latest

Ubuntu 32 bit Desktop URL

http://www.ubuntu.com/download/server/thank-you?distro=desktop&bits=32&release=latest

This step is perhaps the easiest of all steps within the book as it has been proven and improved upon so many times it is nearly flawless. One note worth mentioning is that the 64 and 32 bit versions of the Ubuntu downloads fit rather nicely on a single 800 MB CD. Using the tool “cdrecord” from a Linux command line passing in the ISO image downloaded file name creates an easy and very fast mechanism for installation. If you do not yet have a Linux environment, simply burn the ISO IMAGE to a CD or conversely write it to a USB stick using one of a plethora of available tools like *unetbootin*.

Here is an excellent resource for installing Linux from a Live CD. It is really a very simple process that can be repeated as many time as you wish. https://help.ubuntu.com/community/Live CD

These are the basic steps to most Live CD installations:

- •The Welcome Screen
- •Choose the Keyboard
- •Set the Locale and System Time
- •Set up and Partition the Hard Drive
- •Create the Primary Users and Passwords
- •Installation Overview
- •Install the Operating System
- •Install the BOOT Loader
- •Finish the Installation
- •Reboot the Computer

Following are instructions in detail for the patient participants.

If the computer currently has important files, then backup the files to a separate storage unit like a USB external hard-drive. Once you feel the computer's hard-drive can be erased without regrets, place the Ubuntu installation disc in the disc tray. The live disc will boot-up and a desktop will be seen. Soon, the system installer will appear with the option to "Try Ubuntu" or "Install Ubuntu". On the side, the language can be chosen. Trying Ubuntu allows users to try out the system or perform recovery on the system already on this computer. Please choose Install Ubuntu.

Next, click the timezone of your location or type it in the box.

The following window allows users to setup their keyboard type and layout. A text box is provided for testing.

In the next window, the user will type their name, the hostname, username, and password. The user can also choose

to encrypt their home folder. Also, they can setup the system to automatically log in using that username. The installer will then install the system. This can take some time, so read more cool articles on Linux.org while this installs.

Once the installation is complete, remove the installation disc and click "Restart Now" on the new window.

We continue by erasing the contents of the disk, and partitioning the hard drive for the operating system. A partition is like cutting up a pizza into slices with one exception. Some slices are very much larger than others. Normal “slices” or partitions are as follows:

/ is the slash or root partition. This is where most of the OS will reside.

/boot is the boot partition and within this slice the kernel or the Linux OS and all necessary boot strap data and executable files will reside.

SWAP is memory on disk and is used as an exhaustive measure of last resort. RAM is swapped to SWAP, and is very undesirable as disk is millions, of not billions of times slower that memory.

/home is where the heart is at. This is where our own personal files reside, like for example our downloaded data files.

/var is a partition that is particularly important to us as this is where the Apache www-ssl files for our application will reside.

It is a best practice to over partition the “/” root and /var partitions when setting up a server environment. The /home is not as relevant, so to speak, as compared to a Desktop installation.

Select the desired storage device and click "New Partition Table". Then, click some free space and press the "+" button. In the new window, make the needed . If there is still more free space, repeat the process. This window will allow users to choose a and the mount point. The mount point is the directory that will be attached to this partition. For example, users may want /home/ and /var/ each on a separate partition from /. Unless the user has a specific need, the partitions should be made EXT4. Make sure some space is left for the swap space. To make swap space, set the file system type to swap. Generally, swap space should be twice the size of the current available memory. For better performance, make the swap space two partitions, each the size of the current memory.

Once finished, click "Install Now".

For those of you that do not understand partition encryption, it is a privacy measure. Assume this computer contains private data for a business or hospital. If the computer is ever stolen and the hard-drive is placed in another computer, the data cannot be seen. However, this does not protect the data from being deleted or formatted.

LVM stands for Logical Volume Manager. LVM allows a set of hard-drives to be divided into logical volumes instead of physical partitions. A user could have three 2-terabyte hard-drives. With LVM, the user could make two logical volumes each three TB terabytes. The logical volumes allow partitions to be made that span multiple physical hard-drives.

Boot-up the system and make sure all of your devices work properly. You can then configure the system to your liking and install your needed applications.

Chapter Summary

If you do not know anything about UNIX this is a good starting point:
http://www.dummies.com/store/Computers-Internet/Operating-Systems/Linux.html

When on a Linux server to record a CD from an ISO image simply use cdrecord from the command line as "`cdrecord isofile.iso`."

Creating a USB or a boot-able CD using a GUI from Windows or Linux use: unetbootin.

Look on the Internet for on line information like this: http://www.youtube.com/watch?v=GhnLk3gviWY

If you install the Ubuntu Server and would like to convert it to a Desktop.

```
sudo apt-get install tasksel
sudo tasksel
```

Here is an excellent resource for delving deeper into logical and physical volumes while planning your disk layout.

1. http://www.geekpeek.net/lvm-physical-volume-management/

2. http://www.geekpeek.net/lvm-volume-group-management/

3. http://www.geekpeek.net/lvm-logical-volume-management

Additionally, there is a LVM shell script written specifically for LVM management. It is necessary to boot into single user mode in order to manage volumes while it is best practice to work directly from the console of the computer. To boot into single user mode issue "init 1."

http://geekpeek.net/download/lvm-management-v01.sh

Most important IMHO would be the disk layout. Certainly utilizing LVM is the way to go. Partitioning separate LVMs for root, boot, usr, home, var, tmp, and of course SWAP space is preferable. More space is better than not enough. Though LVM. MySQL, and Linux in general provide adequate resources for resizing "partitions," it is better to over think your needs. MySQL's datadir defaults to /var. This is going to be our main partition. On a 1 TB

disk, I would consider allocating 250 – 300 GB, ¼ or more of your resources to this partition alone. For the "/" partition, I would recommend 100 GB. The remaining partitions are not as important, and with a 2 GB swap which we hope to never use, the remainder can be divided rather equally.

When installing the operating system, choose defaults avoiding any 3rd party software. Do not install MySQL, Apache, or other sub categories, as these will be downloaded and installed separately in later sections. For example to harden Apache it really should be downloaded and compiled locally removing unwanted and unnecessary code. So it is wisest to select most defaults while avoiding 3rd party software add-ons. It is also very likely that there will be more than one installation of this application, as indicated previously. It may be best to prototype on a smaller machine, then install into production once all of the learning curves are overcome.

Chapter 2 The Prerequisites

Step Two prepare the base Linux Operating System

Conflicts and controversy

Updating the operating system and applying the required packages for the array of applications that encompass this technology is generally a controversy, while most definitely introduces operating system conflicts. These conflicts are often resolved by the OS automatically, however installing the packages necessary to fulfill all of the necessary prerequisites is sometimes a juggling act at best. This is why we start with a known operating system base and work our way up from there. What works on a particular Ubuntu distribution may or may not work in a similar fashion on, say Debian, while it most likely will not be the same on CentOS, and so on. It is important to start out on a solid foundation. This is why we have chosen the latest distribution of the Ubuntu Server, or the Ubuntu Desktop.

"In the beginning there was the .tar.gz. Users had to compile each program that they wanted to use on their GNU/Linux systems. When Debian was created, it was deemed necessary that the system include a method of managing the packages installed on the machine. The name dpkg was given to this system. Thus the famous 'package' first came into being on GNU/Linux, a while before Red Hat decided to create their own 'rpm' system.

A new dilemma quickly took hold of the minds of the makers of GNU/Linux. They needed a rapid, practical, and efficient way to install packages that would manage dependencies automatically and take care of their configuration files while upgrading. Here again, Debian led the way and gave birth to APT, the Advanced Packaging Tool, which has since been ported by Conectiva for use with rpm and has been adopted by some other distributions."

-- From Debian APT HOWTO

All prerequisite packages will be installed within this step alleviating the pain of having to upgrade and install additionally required packages as we progress through the steps. This step may take some time to complete. However it is best to get this done prior to continuing. It is a necessary component. In an attempt to upgrade and install all the necessary prerequisite components, not only for MPOS, but for ALL of the applications, the list of

packages may appear to be rather daunting. On a typical server, though it should be no more than half an hour to complete this step.

We will be using the Debian package manager, and the counter part “apt-get”. All commands will be performed with sudo, Super User Do. Please note that if you are running on a Debian platform, and not Ubuntu, then you may need to `su` as the root user and issue this command substituting your user id for *youruserid*: `echo 'youruserid ALL=(ALL) ALL' >> /etc/sudoers`.

Scripted approach

The script for this step is “`step_1_os_prerequisites.sh`”. As stated this is a prerequisite step that requires root access in order to make the necessary and appropriate changes to the operating system. Her we will break the script down into its components.

Update the Operating System to grab the latest OS patches, security updates, etc. form the manufacturer.

```
sudo apt-get -y update
```

Upgrade the Operating System to upgrade the package versions where applicable from the manufacturer, and their recommendations.

```
apt-get -y upgrade
```

Install the general packages. These are the packages that most every other software installation relies on.

```
sudo apt-get -y install git wget build-essential
```

Install the wide prerequisite packages. These packages span multiple applications. That is to say that very many of the application we will be installing depend on these as a base.

```
sudo apt-get -y install libboost-all-dev libcurl4-openssl-dev libdb5.1-dev libdb5.1++-dev mysql-server
```

It is during this installation phase that the MySQL server is installed. This is an interactive installation which asks us for our MySQL root access password. Please respond with your desired password, however, jot this down as it will be very important in future tasks. Changing the root password in MySQL can be a rather tedious task, so please keep this in mind. When presented with the following screen, or similar facsimile, enter your MySQL root password.

A validation screen will appear for the password validation. Validate the password, and make note of it for later steps.

For the Stratum server, the meat so to speak, we require python, and all of its prerequisite packages. The Stratum is the pooled interface to the coin daemon. Its main components are python, though there is a middle layer structure in C.

```
sudo apt-get -y install python-twisted python-mysqldb python-dev python-setuptools python-memcache python-simplejson python-pylibmc
```

The easy_install python utility provides a standard format for distributing Python programs and libraries based on the **Python Eggs** wrapper. Eggs are analogous to jar files in Java.

```
easy_install -U distribute
```

If after the initial easy_install distribute your easy_install ceases to operate, it is very likely that more than one version exists. The correct easy_install should be in your path as the following commands show.

```
steven@steven-M275:~/mpos$ which easy_install
/usr/bin/easy_install
steven@steven-M275:~/mpos$ sudo /usr/bin/easy_install -U distribute
```

Executing other versions of easy install is not recommended. Please use the /usr/bin/easy_install from this point forward if and when appropriate.

MPOS specific requirements include the PHP application, and Apache 2.

```
sudo apt-get -y install memcached apache2 php5 php5-memcached php5-mysqlnd php5-curl
php5-common php5-json libapache2-mod-php5
```

It should now be possible to open a browser to your IP address, or to 127.0.0.1 via the non secure http port 80. Opening a browser and navigating to http://127.0.0.1 should present the following result.

It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

The complete script: step_1_os_prerequisites.sh

```
#!/bin/bash -
#title          :dogon_--_mpos_automation.sh
#description    :Step 01 -- Install the prereqs
#author         :SJmariogolf
#date           :20140401
#version        :0.0-1
#notes          :
#bash_version   :4.2.45(1)-release
#==============================================================================

H=`hostname -i`
```

```
runyesorno(){

if [ "${1}" ] ; then
        yesorno="n"
        read -p "Inform: Would you like to run this step? [${2}/${1}] Enter y/n (n)? "
yesorno
                case "$yesorno" in
        y*|Y*)  ${1};;
                n*|N*)  echo "Skipped";;
        esac
fi

return 0
}

#-- general prereqs
runyesorno "sudo apt-get -y update" "Update the base OS."
runyesorno "sudo apt-get -y upgrade" "Upgrade the base OS."
runyesorno "sudo apt-get -y install git wget build-essential openssh-server" "General
prereqs."
#-- coin daemon prereqs
runyesorno "sudo apt-get -y install libboost-all-dev libcurl4-openssl-dev libdb5.1-dev
libdb5.1++-dev qt4-qmake libqt4-dev autoconf automake mysql-server" "Wide prereqs."
#-- stratum prereqs
runyesorno "sudo apt-get -y install uuid-runtime python-twisted python-mysqldb python-dev
python-setuptools python-memcache python-simplejson python-pylibmc" "Python specific
prereqs."
runyesorno "sudo easy_install -U distribute" "Distribute the python eggs."
#-- mpos and apache
runyesorno "sudo apt-get -y install memcached apache2 php5 php5-memcached php5-mysqlnd
php5-curl php5-common php5-json libapache2-mod-php5" "MPOS and apache 2."
```

Chapter 3 The Coin Daemon and QT Client

Step Three download, compile and execute the coin daemon

Daemons and clients

In multitasking computer operating systems, a daemon (/ˈdeɪmən/) is a computer program that runs as a background process, rather than being under the direct control of an interactive user. Traditionally daemon names end with the letter d: for example, syslogd is the daemon that implements the system logging facility and sshd is a daemon that services incoming SSH connections.

Two compilation processes are involved with each of the coin methods chosen. One compile is for the daemon process, and the other is for the client. The daemon process, and the client process cannot execute simultaneously. Under normal circumstances we will be executing the daemon process, devoid of the client. The client process can execute as a server, or daemon process by passing the -server flag as an executable option, however we will not be using it in that manner. It is the daemon process we are mainly interested in, but will be compiling both the daemon and client processes.

The daemon process performs two IMPORTANT functions.

ONE It downloads the block chain, and for Litecoin this is approximately ½ million records. This of course takes some time to accomplish. Perhaps one half of a day. With Bitcoin it may take 2 and a half days to normalize. We cannot proceed on without the daemon process up to date and normalized. There is a test network block chain that is quite a bit smaller and we can use it to test with, but for mining of actual coins we require a normalized coin daemon. Much more information exists specific to the daemon process and commands to determine its processing state. The purveyor of a production mining site becomes very familiar with the daemon and its interaction.

TWO The daemon creates DEFAULT wallet information and selects a DEFAULT coin address. This address is LOCAL to this establishment and this wallet. No other addresses are relevant at this juncture. We will locate this default address and take special note of it throughout this book. However, this is very IMPORTANT as this is “the address,” and it is OUR address. It is the definitive address we will be using from this time forward with this

particular installation. We will make note of this on several occasions. For now it is enough just to keep this in mind.

For the sake of consistency we will begin with a proven technology. That is Litecoin. The Litecoin daemon and client processes are known to compile and execute correctly and as such make for a perfect first installation. Later in this book we will switch to an alternative coin, and at this juncture it is noteworthy to mention that the Bitcoin is also an excellent coin. Alternative coins are alternative for a reason, and may or may provide support or interoperability. Litecoin works consistently and for this reason alone, let us begin with Litecoin.

Let us begin by ensuring we are not root, rather we are logged in to our server using a normal login. Mine is steven. All users have a $HOME environment variable that is normally /home/*username*, where username is in my case steven. We should change directory to $HOME prior to downloading or installing any software. Generally I work out of /home/steven/workspace, but in any event we should relocate to $HOME, and it would be a good idea to create a folder, or in UNIX terms, a directory entitled mpos.

Make the directory mpos.

```
mkdir $HOME/mpos
```

Using the git process now download the Litecoin software.

```
git clone git://github.com/litecoin-project/litecoin.git
```

Using qmake, make the make file for the QT-Client.

```
cd $HOME/mpos/litecoin
qmake bitcoin-qt.pro
make
sudo cp litecoin-qt /usr/bin
#-- create the $HOME/.litecoin/litecoin.conf
mkdir -p $HOME/.${COIN}
COIN=”litecoin”
MyCUser=${COIN}rpc
MyCPasswd=`uuidgen -r`
cat << EOF > ${HOME}/.${COIN}/${COIN}.conf
server=1
rpcuser=$MyCUser
rpcpassword=$MyCPasswd
rpcallowip=127.0.0.1
EOF
#-- start the daemon
litecoind -daemon
```

Now we wait and watch the daemon process the block chain. We can see the progress by tailing the debug.log file

within the $HOME/.litecoin directory. Another way to check the progress is by issuing these commands.

```
litecoind getinfo
litecoind help
```

Managing the coin daemon

This is a good opportunity to discuss the coin daemon process in general, the directory, debug.log file, commands, wallet, etc. as the process to populate the coin data base can be rather time consuming. Performing a google search, one can determine the normalized block count of most any coin, and by performing the correlative coin command. In our case "`litecoind getinfo`". We look for the correct block count using the Internet, then correlate this count against the results of our get info command. In general I have found that the Bitcoin, for example may require up to two days to normalize. It is also possible to backup, or simply securely copy the block chain from another server as long as the coin daemon is stopped on the sending server.

The directory that contains the daemon and client data is generally, and by default contained within your $HOME directory as a "." dot notated, hidden directory of the same, or similar name as the coin. For example /home/*spiderman*/.litecoin. That is, of course if your login to the Linux server happens to be spiderman. It is also possible to redirect this data directory using a command line argument passed to the daemon at startup. "-datadir=<dir>." This argument will specify a new target data directory that you must adhere to if and whenever it is altered.

Within the directory structure these directories and files are created and maintained by the daemon. For example.

```
steven@steven-M275:~/mpos$ cd ~/.litecoin/
steven@steven-M275:~/.litecoin$ ls
blocks  chainstate  database  db.log  debug.log  litecoin.conf  litecoind.pid  peers.dat
 wallet.dat
```

A couple of file worthy of pointing out here are the debug.log, and of course our wallet. The debug. The debug.log file is nice to tail, as it will indicate progress in our block chain download, errors, peer addresses and the like.

The wallet.dat is fundamental. The first and perhaps foremost command is to determine the default wallet account for this installation. To do this we enter the command to display the default address as: "`litecoind getaddressesbyaccount "" `". This command, combined with another display, as in our case: "`litecoind validateaddress LUZ8btg3Sp00f3d3p4p6bgAKGbv7KCunSh6`" shows us as indicated below that this is our address and that it is local, and ismine is true. This indicates that this address is in our wallet, and is validly our address. Other addresses, even existing ones are not valid to this particular wallet. That is to say, we can copy and replace our new wallet.dat file with an existing one from another server, but we cannot simply add existing addresses into this wallet and expect renumeration. Old and existing addresses belong to an altogether different wallet. This is a rather important concept, because as we mine, we may even solve a block. When this happens, we also would like the rewards to be placed into an account whereby we actually have a valid address and this valid address is actually ours. I hope this is clear.

```
steven@steven-M275:~/.litecoin$ litecoind getaddressesbyaccount ""
[
    "LUZ8btg3Sp00f3d3p4p6bgAKGbv7KCunSh6"
]
steven@steven-M275:~/.litecoin$ litecoind validateaddress
LUZ8btg3Sp00f3d3p4p6bgAKGbv7KCunSh6
{
    "isvalid" : true,
    "address" : "LUZ8btg3Sp00f3d3p4p6bgAKGbv7KCunSh6",
    "ismine" : true,
    "isscript" : false,
    "pubkey" : "037ac449e3bb196fdd7c31e94Sp00f3da2a75778cf98eb51415367e7b9b7e915c9",
    "iscompressed" : true,
    "account" : ""
}
```

This is a more complete description of the key, value, pairs returned from the *getinfo* coind daemon command. An explanation of the meaning of the fields given by 'getinfo' :

version - The version number of this bitcoin-qt or bitcoind program itself. Both of are equivalent. -qt is simply the graphical user interface version

protocolversion: The version of the bitcoin network protocol supported by this client (user agent software).

walletversion: The version of the wallet.dat file. Wallet.dat contains bitcoin addresses and public & private key pairs for these addresses. There is additional data on the wallet. Care must be taken to not restore from an old wallet backup. New addresses generated in the wallet since the old backup was made will not exist in the old backup! Source: https://en.bitcoin.it/wiki/Wallet

balance: The total number of bitcoins held in the wallet.dat file.

blocks: The total number of blocks which constitute the shared block chain.

timeoffset: Seconds of difference between this node's "wall clock time" and the median reported by our network peers.

connections: the number of peers on the bitcoin P2P network that this node is connected to.

proxy: If using a proxy to connect to the network, listed here, otherwise blank.

difficulty: the current mining difficulty factor. Difficulty is increased as more miners and more hash compute power compete to be the next one to have a block of transactions added to the blockchain.

testnet: Boolean value (true OR false). There is a parallel bitcoin network, the testnet, where trials and experiments

may be carried out without impacting the official, live bitcoin P2P network

keypoololdest: timestamp (UNIX epoch) of the oldest key in the keypool

keypoolsize: A number of addresses are kept in reserve by the client. This is the size of that reserve.

paytxfee: Specifies what fee the client is willing to pay to expedite transactions. Miners may choose to ignore transactions that do not pay a fee, and these fee-less transactions will have low priority on queue of pending transaction and may linger there for hours.

To determine if your coin daemon is receiving up to date blocks, and is, if you will, fully normalized data. We can edit or otherwise, less, more, cat, etc. the $HOME/.coindaemon/debug.log, looking backwards from the bottom of the file seeking the word "block." When we find a block received, we can copy this block identifier, and issue the coin daemon command getblock. We then inspect the time which is a epoch tic. To show the date in your time zone in a human readable form, enter this Linux shell command: `date -d @1396530101 – Thu Apr 3 09:01:41 EDT 2014`, passing the time epoch derived from the getblock command. A recent date and time, say within the last half hour pretty much assures us that the daemon is operating properly and is up to date.

```
2014-04-03 13:03:51 received block 34068d985edc4a3e1d671ef1cd503e75945e6917bfc8f9eaee7924b61201ec7a
litecoind getblock 34068d985edc4a3e1d671ef1cd503e75945e6917bfc8f9eaee7924b61201ec7a
{
    "hash" : "34068d985edc4a3e1d671ef1cd503e75945e6917bfc8f9eaee7924b61201ec7a",
    "confirmations" : 1,
    "size" : 3474,
    "height" : 542979,
    "version" : 2,
    "merkleroot" : "360ae413cbe5f923ec29be77243d067926784f5fda50bc883f5f8be56cdab735",
    "tx" : [
        "7e15711460beb856bc814c3448b7811bf55bf02ad58353e01caba1a9826a6753",
        "c37a4c56644c0d9413b1cbca03d34b0d5e3a6f8715fe33f83bcf39ae3e988e95",
        "6c273f3624c259e42f1cebab6d9949634055f7417028c39b48f9658a8f5b6cd4",
        "1782d14e6a344f88f671ff63487a8c3581a323d31584490ab1ac7c335216451d"
    ],
    "time" : 1396530101,
    "nonce" : 1060738048,
    "bits" : "1b0b6796",
    "difficulty" : 5746.34909167,
    "previousblockhash" : "2f1accc500435e59ec56207ef4f46bd5ff98d4e3360b12b497b4340839ea4902"
}
date -d @1396530101
Thu Apr  3 09:01:41 EDT 2014
```

Two more, “fundamental commands” with coin daemons are: *coindaemon*d help, which will display all of the associative command parameters, and *coindaemon*d –help, to display run time parameters and options. The daemon name illustrated coindaemond in our case would be litecoind.

```
steven@steven-M275:~/.litecoin$ litecoind --help
Litecoin version v0.8.6.2-6-gf389e65-beta

Usage:
  litecoind [options]
  litecoind [options] <command> [params]  Send command to -server or litecoind
  litecoind [options] help                List commands
  litecoind [options] help <command>      Get help for a command

Options:
  -?                     This help message
  -conf=<file>           Specify configuration file (default: litecoin.conf)
  -pid=<file>            Specify pid file (default: litecoind.pid)
  -gen                   Generate coins (default: 0)
  -datadir=<dir>         Specify data directory
  -dbcache=<n>           Set database cache size in megabytes (default: 25)
  -timeout=<n>           Specify connection timeout in milliseconds (default: 5000)
  -proxy=<ip:port>       Connect through socks proxy
  -socks=<n>             Select the version of socks proxy to use (4-5, default: 5)
  -tor=<ip:port>         Use proxy to reach tor hidden services (default: same as -proxy)
  -dns                   Allow DNS lookups for -addnode, -seednode and -connect
  -port=<port>           Listen for connections on <port> (default: 9333 or testnet: 19333)
  -maxconnections=<n>    Maintain at most <n> connections to peers (default: 125)
  -addnode=<ip>          Add a node to connect to and attempt to keep the connection open
  -connect=<ip>          Connect only to the specified node(s)
  -seednode=<ip>         Connect to a node to retrieve peer addresses, and disconnect
  -externalip=<ip>       Specify your own public address
  -onlynet=<net>         Only connect to nodes in network <net> (IPv4, IPv6 or Tor)
  -discover              Discover own IP address (default: 1 when listening and no -externalip)
  -checkpoints           Only accept block chain matching built-in checkpoints (default: 1)
  -listen                Accept connections from outside (default: 1 if no -proxy or -connect)
  -bind=<addr>           Bind to given address and always listen on it. Use [host]:port notation for IPv6
  -dnsseed               Find peers using DNS lookup (default: 1 unless -connect)
  -banscore=<n>          Threshold for disconnecting misbehaving peers (default: 100)
  -bantime=<n>           Number of seconds to keep misbehaving peers from reconnecting (default: 86400)
  -maxreceivebuffer=<n>  Maximum per-connection receive buffer, <n>*1000 bytes (default: 5000)
```

```
  -maxsendbuffer=<n>     Maximum per-connection send buffer, <n>*1000 bytes (default: 1000)
  -bloomfilters          Allow peers to set bloom filters (default: 1)
  -paytxfee=<amt>        Fee per KB to add to transactions you send
  -mininput=<amt>        When creating transactions, ignore inputs with value less than this (default: 0.0001)
  -daemon                Run in the background as a daemon and accept commands
  -testnet               Use the test network
  -debug                 Output extra debugging information. Implies all other -debug* options
  -debugnet              Output extra network debugging information
  -logtimestamps         Prepend debug output with timestamp (default: 1)
  -shrinkdebugfile       Shrink debug.log file on client startup (default: 1 when no -debug)
  -printtoconsole        Send trace/debug info to console instead of debug.log file
  -rpcuser=<user>        Username for JSON-RPC connections
  -rpcpassword=<pw>      Password for JSON-RPC connections
  -rpcport=<port>        Listen for JSON-RPC connections on <port> (default: 9332 or testnet: 19332)
  -rpcallowip=<ip>       Allow JSON-RPC connections from specified IP address
  -rpcconnect=<ip>       Send commands to node running on <ip> (default: 127.0.0.1)
  -rpcthreads=<n>        Set the number of threads to service RPC calls (default: 4)
  -blocknotify=<cmd>     Execute command when the best block changes (%s in cmd is replaced by block hash)
  -walletnotify=<cmd>    Execute command when a wallet transaction changes (%s in cmd is replaced by TxID)
  -alertnotify=<cmd>     Execute command when a relevant alert is received (%s in cmd is replaced by message)
  -upgradewallet         Upgrade wallet to latest format
  -keypool=<n>           Set key pool size to <n> (default: 100)
  -rescan                Rescan the block chain for missing wallet transactions
  -salvagewallet         Attempt to recover private keys from a corrupt wallet.dat
  -checkblocks=<n>       How many blocks to check at startup (default: 288, 0 = all)
  -checklevel=<n>        How thorough the block verification is (0-4, default: 3)
  -txindex               Maintain a full transaction index (default: 0)
  -loadblock=<file>      Imports blocks from external blk000??.dat file
  -reindex               Rebuild block chain index from current blk000??.dat files
  -par=<n>               Set the number of script verification threads (up to 16, 0 = auto, <0 = leave that
many cores free, default: 0)

 Block creation options:
  -blockminsize=<n>      Set minimum block size in bytes (default: 0)
  -blockmaxsize=<n>      Set maximum block size in bytes (default: 250000)
  -blockprioritysize=<n> Set maximum size of high-priority/low-fee transactions in bytes (default: 27000)

 SSL options: (see the Litecoin Wiki for SSL setup instructions)
  -rpcssl                                  Use OpenSSL (https) for JSON-RPC connections
```

```
  -rpcsslcertificatechainfile=<file.cert>  Server certificate file (default: server.cert)
  -rpcsslprivatekeyfile=<file.pem>         Server private key (default: server.pem)
  -rpcsslciphers=<ciphers>                 Acceptable ciphers (default:
TLSv1+HIGH:!SSLv2:!aNULL:!eNULL:!AH:!3DES:@STRENGTH)
steven@steven-M275:~/.litecoin$ litecoind help
addmultisigaddress <nrequired> <'["key","key"]'> [account]
addnode <node> <add|remove|onetry>
backupwallet <destination>
createmultisig <nrequired> <'["key","key"]'>
createrawtransaction [{"txid":txid,"vout":n},...] {address:amount,...}
decoderawtransaction <hex string>
dumpprivkey <litecoinaddress>
encryptwallet <passphrase>
getaccount <litecoinaddress>
getaccountaddress <account>
getaddednodeinfo <dns> [node]
getaddressesbyaccount <account>
getbalance [account] [minconf=1]
getbestblockhash
getblock <hash> [verbose=true]
getblockcount
getblockhash <index>
getblocktemplate [params]
getconnectioncount
getdifficulty
getgenerate
gethashespersec
getinfo
getmininginfo
getnetworkhashps [blocks] [height]
getnewaddress [account]
getpeerinfo
getrawmempool
getrawtransaction <txid> [verbose=0]
getreceivedbyaccount <account> [minconf=1]
getreceivedbyaddress <litecoinaddress> [minconf=1]
gettransaction <txid>
gettxout <txid> <n> [includemempool=true]
gettxoutsetinfo
getwork [data]
```

```
getworkex [data, coinbase]
help [command]
importprivkey <litecoinprivkey> [label] [rescan=true]
keypoolrefill
listaccounts [minconf=1]
listaddressgroupings
listlockunspent
listreceivedbyaccount [minconf=1] [includeempty=false]
listreceivedbyaddress [minconf=1] [includeempty=false]
listsinceblock [blockhash] [target-confirmations]
listtransactions [account] [count=10] [from=0]
listunspent [minconf=1] [maxconf=9999999]  ["address",...]
lockunspent unlock? [array-of-Objects]
move <fromaccount> <toaccount> <amount> [minconf=1] [comment]
sendfrom <fromaccount> <tolitecoinaddress> <amount> [minconf=1] [comment] [comment-to]
sendmany <fromaccount> {address:amount,...} [minconf=1] [comment]
sendrawtransaction <hex string>
sendtoaddress <litecoinaddress> <amount> [comment] [comment-to]
setaccount <litecoinaddress> <account>
setgenerate <generate> [genproclimit]
setmininput <amount>
settxfee <amount>
signmessage <litecoinaddress> <message>
signrawtransaction <hex string> [{"txid":txid,"vout":n,"scriptPubKey":hex,"redeemScript":hex},...]
[<privatekey1>,...] [sighashtype="ALL"]
stop
submitblock <hex data> [optional-params-obj]
validateaddress <litecoinaddress>
verifychain [check level] [num blocks]
verifymessage <litecoinaddress> <signature> <message>
```

To stop and start the daemon: coindaemon `stop`; coindaemon `-daemon`

The complete script: step_2_coin_daemon_and_client.sh

```
#!/bin/bash -
#title           :dogon_--_mpos_automation.sh
#description     :Step 01 -- Install the coin daemon and client
#author          :SJmariogolf
```

```
#date           :20140401
#version        :0.0-1
#notes          :
#bash_version   :4.2.45(1)-release
#==============================================================================

H=`hostname -i`
COIN='litecoin'
GITREPO="http://github.com/${COIN}-project/${COIN}.git"
QTPRO="bitcoin-qt.pro"

runyesorno(){

if [ "${1}" ] ; then
        yesorno="n"
        read -p "Inform: Would you like to run this step? [${2}/${1}] Enter y/n (n)? "
yesorno
                case "$yesorno" in
        y*|Y*)  ${1};;
                n*|N*)  echo "Skipped";;
        esac
fi

return 0
}

sub_make_coin_conf(){

mkdir -p $HOME/.${COIN}

MyCUser=${COIN}rpc
MyCPasswd=`uuidgen -r`
cat << EOF > ${HOME}/.${COIN}/${COIN}.conf
```

```
server=1
rpcuser=$MyCUser
rpcpassword=$MyCPasswd
rpcallowip=127.0.0.1
EOF

return 0
}

#-- make the mpos directory

runyesorno "mkdir -p $HOME/mpos" "Make the $HOME/mpos directory."
runyesorno "cd $HOME/mpos" "Change directory to $HOME/mpos."
runyesorno "git clone ${GITREPO}" "Download ${COIN}."
runyesorno "cd $HOME/mpos/${COIN}" "Change directory to $HOME/mpos/${COIN}."
runyesorno "qmake ${QTPRO}" "Run qmake for the client."
runyesorno "make" "Compile."
runyesorno "sudo cp ${COIN}-qt /usr/bin" "Copy the client to /usr/bin."
runyesorno "cd $HOME/mpos/${COIN}/src" "Change directory into the daemon source."
runyesorno "make -f makefile.unix USE_UPNP=-" "Make the daemon process."
runyesorno "sudo cp ${COIN}d /usr/bin" "Copy the daemon to /usr/bin."
runyesorno "cd $HOME/mpos" "Change directory to $HOME/mpos."
runyesorno "sub_make_coin_conf" "Make the coin conf file."
runyesorno "${COIN}d -daemon" "Start the daemon."
```

Chapter 4 taking a test drive with CPU Miner

At this juncture you can proceed to install a mining client or proceed on with the installation. However it takes some time for the coin daemon to download its block chain and to normalize, so this is a good time to install a mining client like cpuminer. The simplest way to install a cpuminer client miner is to use the Knary automation freely available from github at git clone https://github.com/sjmariogolf/knary.git.

To make use of the Knary mining setup tool you can follow these instructions, otherwise to manually install cpuminer skip to the end of this chapter.

Automated Installation procedure

Change directory to HOME, or otherwise the directory within which to install Knary.

Install the Knary suite using one of the available methods as is depicted below. YOU MUST be on an active network with access to the Internet.

git clone https://github.com/sjmariogolf/knary.git

Change into the Knary directory...

```
cd knary/
```

Execute the main script...

```
./knary_execution.sh
```

Prior to moving forward the application MUST install the “dialog” package. The dialog package is used by Knary as the CUI (Command User Interface,) presenting screens similar in nature to any Linux installation.

```
 ./knary_execution.sh
Knary::, Sat Mar 01 2014 15:02:28 ,determine-os,16,root,Inform: LITTLE Endian\!
Knary::, Sat Mar 01 2014 15:02:28 ,determine-os,32,root,Inform: [Linux steven-Studio-1440 2.6.32-431.el6.i686 #1 SMP Fri Nov 22 00:26:36 UTC 2013 i686 i686 i386 GNU/Linux] I am a RedHat or Clone.
dialog-1.1-9.20080819.1.el6.i686
Ok to proceed...
dialog-1.1-9.20080819.1.el6.i686
```

Disclaimer

This book and software is covered under a very open source license and is part of the GPL (Gnu public License) and as such, it is FREE of liens, liabilities and reprisals. It is OPEN SOURCE. You may change it as you see fit. Re distribution of the software however must include the name Knary and the Authors Name may NOT be removed from the redistributed code. The programs included within are free and open source; the exact distribution terms for each additional program are described in the individual files within their respective */copyright.

Knary comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

The disclaimer page

This page will be displayed once per Knary session execution.

The MAIN MENU

This is the main navigation menu.

1. 1.QUIT Return or Exit the application.
2. 2.CPU-MINING Navigate to the CPU Mining sub menu.

3. 3.GPU-MINING Navigate to the GPU Mining sub menu.
4. 4.ASICS-MINING Navigate to the ASICs Mining sub menu.
5. 5.BITCOIN Navigate to the BITCOIN sub menu.
6. 6.LITECOIN Navigate to the LITECOIN sub menu.
7. 7.STRATUM Navigate to the STRATUM sub menu.
8. 8.P2POOL Navigate to the P2POOL sub menu.
9. 9.MISC Navigate to the Miscellaneous sub menu.

Choose CPU-MINING to navigate to the CPU Mining sub menu.

Use the <Up/Down> arrow keys to highlight the desired line. Use the <Tab> key to move between <Ok> and <Cancel>.

Select CPU-MINING, and <Ok>.

Viewing the Sub Menu for CPU Mining. From this menu both bfgminer, and cpuminer installation and configuration can commence.

To INSTALL cpuminer, select CPU-INSTALL, and <Ok>.

Choose an installation path

Your actual installation path will depend on the user identification HOME environment variable. For instance if your name is steven, this directory might me relative to /home/steven/exec. Each installation Knary performs creates subordinate directories under this path. It is a good idea to leave this installation path as is. Not necessarily as it is shown below, rather YOUR home directory/knary/exec. All executable files will be placed into their own sub directories under exec. For cpuminer this will be HOME/knary/exec/cpuminer. Within this cpuminer sub directory you will find the executable and configuration files specific to cpuminer.

The installation defaults to wherever you're starting path is + exec/.

The resulting software will be placed into directories BELOW the exec/ as say for example exec/cpuminer. Later in this section we will be creating our own cpuminer configuration files. These files will also be placed into the cpuminer sub directory for current and later use.

<Create> to continue. <Rename> to “change” the path to install, or otherwise <Cancel>.

Install prerequisite packages

The installer can choose whether or not to install certain prerequisite, packages, etc. This is a function of re usability. To un-select, or select use the <Up/Down> arrow keys and the <Space> bar.

These have been determined to be the prerequisite packages necessary to boost the Operating System from “BARE BONES” to where it needs to be to install and operate the Mining Software. Depending on your expertise … Install

or rather Not Install as you see fit.

<OK> continues <Cancel> returns to the previous menu.

Monitoring the prerequisite updates and installs

If all goes well with an “Active Internet” the OS will be updated, the prerequisite packages will be installed, and the software for mining will be downloaded and installed. When software is installed (for example) cpuminer in it’s distinct directory name (cpuminer-2.3.2) Knary will always link a symbolic (cpuminer) for navigation ease.

```
[root@steven-Studio-1440 exec]# ls -la
drwxr-xr-x.  5 root root  4096 Mar  1 15:51 .
drwxr-xr-x. 13 root root  4096 Mar  1 15:43 ..
lrwxrwxrwx.  1 root root 31 Feb 28 10:36 cpuminer -> /root/knary/exec/cpuminer-2.3.2
drwxr-xr-x.  4 1000 1000  4096 Mar  1 15:36 cpuminer-2.3.2
drwxrwxr-x.  6 1000 1000  4096 Feb 28 20:20 jansson-2.4
[root@steven-Studio-1440 exec]# pwd
/root/knary/exec
[root@steven-Studio-1440 exec]#
```

Testing

Testing starts cpuminer with defaults which simply insures a connection and proper installation. This screen depicts proper installation and execution.

The test command used to invoke the cpuminer is as follows:

```
./minerd –url=http:freedom.wemineltc.com:3339 --userpass=sjmariogolf.1:peeb &
```

This command submits to the background a Test process, then when the enter key is pressed, the process is Terminated.

For the complete syntax refer to the help offered by the author of cpuminer.

```
./minerd –help
```

This concludes the Automated Installation.

Manual Installation procedure

The manual installation process is a pretty straight forward procedure. We start by downloading cpuminer and insuring we have all of the necessary prerequisite packages installed which we should have by now.

To build cpuminer from source code we will download its source code from its repository:

```
#!/bin/bash -
#title          :dogon_--_mpos_automation.sh
#description    :Step 03 -- Compile cpuminer
#author         :SJmariogolf
```

```
#date           :20140401
#version        :0.0-1
#notes          :
#bash_version   :4.2.45(1)-release
#==============================================================================
#-- make sure we have the prereqs
  sudo aptitude install git automake pkg-config gcc make
MINEREL="cpuminer-2.3.2"
SUBject="cpuminer"
myinstalloc=${HOME}/mpos
CFLAGS="-O3"
cd ${myinstalloc}
#-- wget the source code, it is the best way
wget http://sourceforge.net/projects/cpuminer/files/pooler-${MINEREL}.tar.gz
tar vxzf pooler-${MINEREL}.tar.gz
ln -s ${myinstalloc}/${MINEREL} ${myinstalloc}/${SUBject}
# compile it ourselves
cd ${myinstalloc}/${SUBject}
./configure CFLAGS="${CFLAGS}"
make clean
make
sudo make install
sudo ldconfig
```

Launch minerd program:

By default it mines using scrypt algorithm:

```
./minerd -o http://127.0.0.1:19333 -u rpcuser -p rpcpasswd
```

The rpc getwork user and password exist within the *coindaemon*.conf file under your $HOME/.litecoin directory, in our case this is the litecoin.conf. Within this conf file there is an rpcuser and an rpcpassword. For now, these will be used for testing suing the getwork ports of 9332.

Example:

```
cat ~/.litecoin/litecoin.conf
server=1
```

```
rpcuser=litecoinrpc
rpcpassword=0f2ab171-35de-4SP00f3d-0-9b582244b7dd
rpcallowip=127.0.0.1

./minerd -o 127.0.0.1:9332 -u litecoinrpc -p 0f2ab171-35de-4SP00f3d-0-9b582244b7dd
[2014-04-02 12:20:46] 1 miner threads started, using 'scrypt' algorithm.
[2014-04-02 12:20:49] thread 0: 4096 hashes, 1.37 khash/s
[2014-04-02 12:20:54] thread 0: 6856 hashes, 1.40 khash/s
...
```

Chapter 5 divide and conquer the Stratum Proxy

Enter Stratum

Stratum is an overlay network on the top of coin daemon P2P protocol, creating simplified facade for lightweight clients and hiding unnecessary complexity of decentralized protocol. However there's much bigger potential in this overlay network than just providing simplified API for accessing blockchain stored on Stratum servers. For utilization of such potential, we definitely need some robust protocol providing enough flexibility for various type of clients and their purposes.

In the previous chapter we established a mining client cpuminer. We also mined with this client solo, processing the 9332 TCP port. Using this approach it is unlikely that we will ever reach the target minimum network hash required for any given blockchain, conversely what we require is a proxy. The proxy is just that. It is a proxy between us and the network, varying the difficulty to be accepted into the "mining pool," so to speak. The variable difficulty presented by the proxy varies the network difficulty allowing a low enough difficulty that we are actually able to participate.

Having this process automated and scripted gives it the appearance of being a simple process. I assure you that it is not as simple and straight forward as it seems. For this reason I would like to pay homage to the many individuals and many more, countless hours spent developing this wonderful application suite we are delving into here now. So let us install the Stratum proxy.

As we are currently working with Litecoin we will download the litecoin scrypt code as well as the stratum core and proxy. Prior to using github.com, we have to insure we have all of the necessary prerequisite packages. By this

point, we should be good to go, however it is best to include such prerequisite coding.

Install the prerequisites and update.

```
sudo apt-get install python-twisted python-mysqldb python-dev python-setuptools
python-memcache python-simplejson python-pylibmc
```

```
sudo /usr/bin/easy_install -U distribute
```

Now download the python scrypt library, the stratum core, and the stratum mining proxy into your $HOME/mpos directory. A note here, is that it is good to work out of a single directory like mpos. This consolidates the sub directories and files within one “folder.” The following code with download the necessary files into their proper sub directories.

```
cd $HOME/mpos
```

```
git clone https://github.com/Tydus/litecoin_scrypt.git
```

```
git clone https://github.com/ahmedbodi/stratum-mining.git
```

```
git clone https://github.com/ahmedbodi/stratum.git
```

The resulting sub directory structure should resemble something akin to the structure below.

```
cpuminer cpuminer-2.3.2 litecoin litecoin_scrypt pooler-cpuminer-2.3.2.tar.gz stratum
```

```
stratum-mining
```

Now continue with the initialization, setup and on to configuration.

```
cd $HOME/mpos/stratum-mining
```

```
git submodule init
```

```
git submodule update
```

```
cd $HOME/mpos/stratum-mining/externals/litecoin_scrypt
```

```
sudo python setup.py install
```

```
cd $HOME/mpos/stratum-mining/externals/stratum
```

```
sudo python setup.py install
```

Next configure the Stratum proxy by adjusting the $HOME/stratum-mining/config/config.py file. There is not a lot of information out there on this file. It really is kind of a guessing game, and learn by trial and error. However, by reading this book, your already one up because a lot of the leg work has already been done. A working configuration file in its entirety is below. There are a few parameters that require explanation.

First and foremost is the CENTRAL_WALLET. This is your address, and is the address of this installation of the litecoin daemon. Unless you replaced the wallet.dat, this is your address. We find the wallet address and validate it using these two commands: litecoind getaccountaddress "", and litecoind validateaddress. Make sure the isvalid is true, and the ismine is true. Otherwise you'll not be mining into your wallet. Maybe someone else's, but certainly not

yours.

```
CENTRAL_WALLET = 'This is YOUR wallet address'
```

Next is the TRUSTED_USER and TRUSTED_PASSWORD. These values we derive directly form the $HOME/.litecoin/litecoin.conf file. Make sure to place all values inside of single quotes.

```
LITECOIN_TRUSTED_USER = 'litecoinrpc'
LITECOIN_TRUSTED_PASSWORD = 'ed9e9c60-b7d1-408c-8200-sp00f3d18ad5'
```

This entire file after modifying the above parameters is at best a good starting point for the config.py file within the stratum-mining/conf directory. Additionally at the bottom of this file you may want to replace the email addresses with your email address. So copy this file and place it as the config.py file within the $HOME/mpos/stratum-mining/conf directory. Copy only the lines in currier font. A holistic version of the config.py exists at the end of this chapter. With the minor changes indicated above, it is ready for deployment into a test network.

The entire config.py file start ...

```
'''
This is example configuration for Stratum server.
Please rename it to config.py and fill correct values.
This is already setup with sane values for solomining.
You NEED to set the parameters in BASIC SETTINGS
'''
# ******************** BASIC SETTINGS ***************
# These are the MUST BE SET parameters!
LITECOIN_ALGO = 'scrypt'
LITECOIN_Reward = 'POW'
LITECOIN_TX = 'no'
CENTRAL_WALLET = 'LXwArgiagu9Sp00f3d4uaR7g4be7UZcyk' # local bitcoin address where money goes
LITECOIN_TRUSTED_HOST = 'localhost'
LITECOIN_TRUSTED_PORT = 9332
LITECOIN_TRUSTED_USER = 'litecoinrpc'
LITECOIN_TRUSTED_PASSWORD = 'ed9e9c60-sp00f-4u2c-8200-7e3b83818ad5'
# ******************** BASIC SETTINGS ***************
# Backup Litecoind connections (consider having at least 1 backup)
# You can have up to 99
#LITECOIN_TRUSTED_HOST_1 = 'localhost'
#LITECOIN_TRUSTED_PORT_1 = 8332
#LITECOIN_TRUSTED_USER_1 = 'user'
#LITECOIN_TRUSTED_PASSWORD_1 = 'somepassword'
#LITECOIN_TRUSTED_HOST_2 = 'localhost'
```

```
#LITECOIN_TRUSTED_PORT_2 = 8332
#LITECOIN_TRUSTED_USER_2 = 'user'
#LITECOIN_TRUSTED_PASSWORD_2 = 'somepassword'
# ******************** GENERAL SETTINGS ***************
# Enable some verbose debug (logging requests and responses).
DEBUG = False
# Destination for application logs, files rotated once per day.
LOGDIR = 'log/'
# Main application log file.
LOGFILE = "stratum.log"  # eg. 'stratum.log'
# Possible values: DEBUG, INFO, WARNING, ERROR, CRITICAL
LOGLEVEL = 'INFO'
# Logging Rotation can be enabled with the following settings
# It if not enabled here, you can set up logrotate to rotate the files.
# For built in log rotation set LOG_ROTATION = True and configure the variables
LOG_ROTATION = True
LOG_SIZE = 10485760 # Rotate every 10M
LOG_RETENTION = 10 # Keep 10 Logs
# How many threads use for synchronous methods (services).
# 30 is enough for small installation, for real usage
# it should be slightly more, say 100-300.
THREAD_POOL_SIZE = 300
# Disable the example service
ENABLE_EXAMPLE_SERVICE = False
GW_ENABLE = False
# ******************** TRANSPORTS *********************
# Hostname or external IP to expose
HOSTNAME = 'steven-Studio-1440'
# Port used for Socket transport. Use 'None' for disabling the transport.
LISTEN_SOCKET_TRANSPORT = 3333
# Port used for HTTP Poll transport. Use 'None' for disabling the transport
LISTEN_HTTP_TRANSPORT = 3334
# Port used for HTTPS Poll transport
LISTEN_HTTPS_TRANSPORT = None
# Port used for WebSocket transport, 'None' for disabling WS
LISTEN_WS_TRANSPORT = None
# Port used for secure WebSocket, 'None' for disabling WSS
LISTEN_WSS_TRANSPORT = None
# Salt used when hashing passwords
```

```
PASSWORD_SALT = '087c2894-db34-4ae7-a090-be566a926b2c'
# ******************** Database  ********************
# MySQL
DATABASE_DRIVER = 'mysql'
DATABASE_EXTEND = False
DB_MYSQL_HOST = 'localhost'
DB_MYSQL_DBNAME = 'mpos'
DB_MYSQL_USER = 'root'
DB_MYSQL_PASS = '6a46d66p'
# ******************** Adv. DB Settings *********************
#  Don't change these unless you know what you are doing
DB_LOADER_CHECKTIME = 15 # How often we check to see if we should run the loader
DB_LOADER_REC_MIN = 1  # Min Records before the bulk loader fires
DB_LOADER_REC_MAX = 50  # Max Records the bulk loader will commit at a time
DB_LOADER_FORCE_TIME = 300      # How often the cache should be flushed into the DB regardless of size.
DB_STATS_AVG_TIME = 300  # When using the DATABASE_EXTEND option, average speed over X sec
    # Note: this is also how often it updates
DB_USERCACHE_TIME = 600  # How long the usercache is good for before we refresh
# ******************** Pool Settings *********************
# User Auth Options
USERS_AUTOADD = False  # Automatically add users to db when they connect.
                           # This basically disables User Auth for the pool.
USERS_CHECK_PASSWORD = False # Check the workers password? (Many pools don't)
# Transaction Settings
COINBASE_EXTRAS = '/stratumPool/'   # Extra Descriptive String to incorporate in solved blocks
ALLOW_NONLOCAL_WALLET = True    # Allow valid, but NON-Local wallet's
# Litecoind communication polling settings (In Seconds)
PREVHASH_REFRESH_INTERVAL = 5  # How often to check for new Blocks
    # If using the blocknotify script (recommended) set = to MERKLE_REFRESH_INTERVAL
    # (No reason to poll if we're getting pushed notifications)
MERKLE_REFRESH_INTERVAL = 60 # How often check memorypool
    # This effectively resets the template and incorporates new transactions.
    # This should be "slow"
INSTANCE_ID = 31  # Used for extranonce and needs to be 0-31
# ******************** Pool Difficulty Settings *********************
#  Again, Don't change unless you know what this is for.
# Pool Target (Base Difficulty)
POOL_TARGET = 1   # Pool-wide difficulty target int >= 1
VDIFF_TARGET = 30  # Target time per share (i.e. try to get 1 share per this many seconds)
```

```
VDIFF_RETARGET = 300  # Check to see if we should retarget this often
# ******************** Pool Difficulty Settings **********************
VDIFF_X2_TYPE = True  # powers of 2 e.g. 2,4,8,16,32,64,128,256,512,1024
VDIFF_FLOAT = True    # Use float difficulty


# Pool Target (Base Difficulty)
POOL_TARGET = 32                        # Pool-wide difficulty target int >= 1
# Variable Difficulty Enable
VARIABLE_DIFF = True                # Master variable difficulty enable
# Variable diff tuning variables
#VARDIFF will start at the POOL_TARGET. It can go as low as the VDIFF_MIN and as high as min(VDIFF_MAX or
Litecoin's difficulty)
USE_LITECOIN_DIFF = False   # Set the maximum difficulty to the litecoin difficulty.
DIFF_UPDATE_FREQUENCY = 86400 # Update the litecoin difficulty once a day for the VARDIFF maximum
VDIFF_MIN_TARGET = 16                #  Minimum Target difficulty
VDIFF_MAX_TARGET = 1024                # Maximum Target difficulty
VDIFF_TARGET_TIME = 15                # Target time per share (i.e. try to get 1 share per this many seconds)
VDIFF_RETARGET_TIME = 120                # Check to see if we should retarget this often
VDIFF_VARIANCE_PERCENT = 30        # Allow average time to very this % from target without retarget
# ******************** Admin settings **********************
# Use scripts/generateAdminHash.sh <password> to generate the hash
# for calculating SHA256 of your preferred password
ADMIN_PASSWORD_SHA256 = '7a08095e5c4c9ae158314a1a00Sp00f3d98433dSp00fe3db9db38e365b0' # SHA256 of the password
# ******************** E-Mail Notification Settings **********************
NOTIFY_EMAIL_TO = 'youremail@gmail.com'            # Where to send Start/Found block notifications
NOTIFY_EMAIL_TO_DEADMINER = ''  # Where to send dead miner notifications
NOTIFY_EMAIL_FROM = 'root@localhost'    # Sender address
NOTIFY_EMAIL_SERVER = 'localhost'       # E-Mail Sender
NOTIFY_EMAIL_USERNAME = 'youremail@gmail.com'              # E-Mail server SMTP Logon
NOTIFY_EMAIL_PASSWORD = 'youremail@gmail.com'
NOTIFY_EMAIL_USETLS = True
```

The entire config.py file end ...

We are not ready to start the Stratum yet, but it is installed and configured. In a later section we will be bringing all of the applications up rather simultaneously. For now it is on to the MPOS application suite, Apache/PHP, and MySQL.

The installation script:

```
#!/bin/bash -
#title           :dogon_--_mpos_automation.sh
```

```
#description    :Step 04 -- The Stratum Proxy
#author         :SJmariogolf
#date           :20140401
#version        :0.0-1
#notes          :
#bash_version   :4.2.45(1)-release
#==============================================================================

#-- make sure we have the prereqs
sub_prerequisites(){

sudo apt-get install python-twisted python-mysqldb python-dev python-setuptools
python-memcache python-simplejson python-pylibmc
sudo /usr/bin/easy_install -U distribute

return 0
}

runyesorno(){

if [ "${1}" ] ; then
        yesorno="n"
        read -p "Inform: Would you like to run this step? [${2}/${1}] Enter y/n (n)? "
yesorno
                case "$yesorno" in
        y*|Y*)  ${1};;
                n*|N*)  echo "Skipped";;
        esac
fi

return 0
}
```

```
sub_install_stratum(){

cd $HOME/mpos/stratum-mining
git submodule init
git submodule update

cd $HOME/mpos/stratum-mining/externals/litecoin_scrypt
sudo python setup.py install

cd $HOME/mpos/stratum-mining/externals/stratum
sudo python setup.py install

return 0
}

runyesorno "mkdir -p $HOME/mpos" "Make the $HOME/mpos directory."
runyesorno "cd $HOME/mpos" "Change directory to $HOME/mpos."
runyesorno "sub_prerequisites" "Insure we have the prerequisites."
runyesorno "git clone https://github.com/Tydus/litecoin_scrypt.git" "Obtain the litecoin
scrypt."
runyesorno "git clone https://github.com/ahmedbodi/stratum-mining.git" "Obtain the
stratum mining proxy."
runyesorno "git clone https://github.com/ahmedbodi/stratum.git" "Obtain the stratum
core."
runyesorno "sub_install_stratum" "Install the Stratum proxy."
```

Chapter 6 the initial MPOS installation using http

Enter MPOS

From the github MPOS site: MPOS is a web based Mining Portal for various crypto currencies. It was created by

TheSerapher and has hence grown quite large. Recently it was migrated into a Github Organization to make development easier. It's a community driven open source project. Support can be requested on IRC at https://webchat.freenode.net/?channels=#mpos.

MPOS is a web based application comprised mainly of PHP. http://www.php.net/. Combined with, and tuned to the Stratum proxy it offers you a mining portal akin, of not identical to, those pooled we based crypto-currency mining portals we should be all to familiar with. MPOS utilizes MySQL, and Apache 2, and as such, these two sub systems are integral, and must be installed and configured prior to the MPOS launch. This chapter is devoted to the MPOS installation, while further chapters devote themselves entirely to the configuration, execution, and security considerations of the MPOS, PHP, and the Apache components.

In this chapter we basically are going to download MPOS and stage it inside out own home directory. In a later chapter we put I all together for http, then further, we will secure it with https. It is always best to install out of a sub directory, so we navigate to our $HOME/mpos directory and execute the following steps in order.

Insure we have the latest prerequisites as we should at this point.

```
sudo apt-get install uuid-runtime memcached php5-memcached php5-mysqlnd php5-curl
php5-json libapache2-mod-php5
```

Change directory into the $HOME/mpos, and use the git tool to download MPOS.

```
cd $HOME/mpos
```

```
git clone git://github.com/MPOS/php-mpos.git MPOS
```

Change directory into the MPOS sub directory and obtain the latest code.

```
cd $HOME/mpos/MPOS
```

```
git checkout master
```

Change the ownership of the files Apache is required write permission to.

```
APACHE_GROUP='www-data'
```

```
sudo chown -R ${APACHE_GROUP} $HOME/mpos/MPOS/public/templates/compile
$HOME/mpos/MPOS/public/templates/cache $HOME/mpos/MPOS/logs
```

Copy the global configuration distribution file to the global configuration file.

```
cp $HOME/mpos/MPOS/public/include/config/global.inc.dist.php
$HOME/mpos/MPOS/public/include/config/global.inc.php
```

Seed the random SALT variables with proper salted seeds.

```
SALT=`uuidgen`
```

```
SALTY=`uuidgen`
```

```
sed -i "s/PLEASEMAKEMESOMETHINGRANDOM/${SALT}/g"
$HOME/mpos/MPOS/public/include/config/global.inc.dist.php
```

```
sed -i "s/THISSHOULDALSOBERRAANNDDOOM/${SALTY}/g"
$HOME/mpos/MPOS/public/include/config/global.inc.dist.php
```

```
grep "SALT" $HOME/mpos/MPOS/public/include/config/global.inc.dist.php
```

Patch the python web socket transport.

```
sudo sed -i "s/from autobahn.websocket import WebSocketServerProtocol/from autobahn.twisted.websocket import WebSocketServerProtocol/" /usr/local/lib/python2.7/dist-packages/stratum-0.2.13-py2.7.egg/stratum/websocket_transport.py
```

```
grep "from autobahn.twisted.websocket" /usr/local/lib/python2.7/dist-packages/stratum-0.2.13-py2.7.egg/stratum/websocket_transport.py
```

Initially configure the MPOS global configuration file.

```
sed -i "s/'username'] = 'testnet'/'username'] = '${RPCUSER}'/" $HOME/mpos/MPOS/public/include/config/global.inc.php
```

```
sed -i "s/'password'] = 'testnet'/'password'] = '${RPCPASSWORD}'/" $HOME/mpos/MPOS/public/include/config/global.inc.php
```

```
sed -i "s/'host'] = 'localhost:19334'/'host'] = '${RPCLISTEN}'/" $HOME/mpos/MPOS/public/include/config/global.inc.php
```

```
cp $HOME/mpos/MPOS/public/include/config/global.inc.php .
```

We have now successfully downloaded MPOS, performed some rudimentary configuration, and patched a python bug that would have otherwise bitten us. We are now ready to proceed into MySQL, Apache, and Send mail. For send mail we will use a local MTA and google as a transport. This is for ease of setup and it is a very good introduction to using a mail transport relay. So let us proceed to the next chapter. See you there!

The entire script.

```
#!/bin/bash -
#title          :dogon_--_mpos_automation.sh
#description    :Step 04 -- MPOS Installation
#author         :SJmariogolf
#date           :20140401
#version        :0.0-1
#notes          :
#bash_version   :4.2.45(1)-release
#==============================================================================

APACHE_GROUP="www-data"
COIN="litecoin"
RPCUSER=`grep rpcuser $HOME/.${COIN}/${COIN}.conf | cut -d'=' -f2`
```

```
RPCPASSWORD=`grep rpcpassword $HOME/.${COIN}/${COIN}.conf | cut -d'=' -f2`
RPCLISTEN="localhost:9332"

#-- make sure we have the prereqs
sub_prerequisites(){

sudo apt-get install uuid-runtime memcached php5-memcached php5-mysqlnd php5-curl
php5-json libapache2-mod-php5

return 0
}

sub_restart_apache(){

sudo /etc/init.d/apache2 restart

return 0
}

runyesorno(){

if [ "${1}" ] ; then
        yesorno="n"
        read -p "Inform: Would you like to run this step? [${2}/${1}] Enter y/n (n)? "
yesorno
                case "$yesorno" in
        y*|Y*)  ${1};;
                n*|N*)  echo "Skipped";;
        esac
fi

return 0
}
```

```
sub_change_salt(){


SALT=`uuidgen`
SALTY=`uuidgen`
sed -i "s/PLEASEMAKEMESOMETHINGRANDOM/${SALT}/g"
$HOME/mpos/MPOS/public/include/config/global.inc.dist.php
sed -i "s/THISSHOULDALSOBERRAANNDDOOM/${SALTY}/g"
$HOME/mpos/MPOS/public/include/config/global.inc.dist.php
grep "SALT" $HOME/mpos/MPOS/public/include/config/global.inc.dist.php


return 0
}


sub_fix_websocket_transport(){


if [ -f
/usr/local/lib/python2.7/dist-packages/stratum-0.2.13-py2.7.egg/stratum/websocket_transpor
t.py ];then
  sudo sed -i "s/from autobahn.websocket import WebSocketServerProtocol/from
autobahn.twisted.websocket import WebSocketServerProtocol/"
/usr/local/lib/python2.7/dist-packages/stratum-0.2.13-py2.7.egg/stratum/websocket_transpor
t.py
  grep "from autobahn.twisted.websocket"
/usr/local/lib/python2.7/dist-packages/stratum-0.2.13-py2.7.egg/stratum/websocket_transpor
t.py
fi


return 0
}


sub_configure_mpos(){


sed -i "s/'username'] = 'testnet'/'username'] = '${RPCUSER}'/"
$HOME/mpos/MPOS/public/include/config/global.inc.php
sed -i "s/'password'] = 'testnet'/'password'] = '${RPCPASSWORD}'/"
```

```
$HOME/mpos/MPOS/public/include/config/global.inc.php
 sed -i "s/'host'] = 'localhost:19334'/'host'] = '${RPCLISTEN}'/"
$HOME/mpos/MPOS/public/include/config/global.inc.php


 cp $HOME/mpos/MPOS/public/include/config/global.inc.php .


 return 0
 }


 runyesorno "mkdir -p $HOME/mpos" "Make the $HOME/mpos directory."
 runyesorno "cd $HOME/mpos" "Change directory to $HOME/mpos."
 runyesorno "sub_prerequisites" "Insure we have the prerequisites."
 runyesorno "git clone git://github.com/MPOS/php-mpos.git MPOS" "download the MPOS
source."
 runyesorno "cd $HOME/mpos/MPOS" "Change directory to $HOME/mpos/MPOS."
 runyesorno "git checkout master" "Obtain the master branch."
 runyesorno "sudo chown -R ${APACHE_GROUP} $HOME/mpos/MPOS/public/templates/compile
$HOME/mpos/MPOS/public/templates/cache logs" "Change ownership for Apache 2."
 runyesorno "cp $HOME/mpos/MPOS/public/include/config/global.inc.dist.php
$HOME/mpos/MPOS/public/include/config/global.inc.php" "Copy the default global
configuration."
 runyesorno "sub_change_salt" "Change the random salt."
 runyesorno "sub_fix_websocket_transport" "Fix the web socket transport."
 runyesorno "sub_configure_mpos" "Initial configuration."
```

Chapter 7 the initial MySQL installation and configuration

Enter MySQL

The MySQL database has become the world's most popular open source database because of its consistent fast performance, high reliability and ease of use. MySQL is used on every continent – yes, even in Antarctica! – by individuals, Web developers, as well as many of the world's largest and fastest-growing organizations such as

industry leaders Yahoo!, Alcatel-Lucent, Google, Nokia, YouTube and others to save time and money powering their high-volume websites, business-critical systems, and packaged software.

As most products do, MySQL comes "ready-to-work" out of the box. Usually, security is not a major consideration when installing this kind of product. Often, the most important issue is to get it up and running as quickly as possible so that the organization can benefit. This document is intended as a quick security manual to help you bring an installed MySQL database server into conformity with best security practices.

While we are here we might as well conform to some best practices. We are after all scripting this installation which should make the whole ordeal somewhat simpler. MPOS out of the box installs as the MySQL root user and assigns the default database name on mpos. This seems to me like a very likely source for those Internet bad guys out there to hack. So we will change things up a bit.

Make a backup of the current my.cnf for preservation.

```
sudo cp /etc/mysql/my.cnf /etc/mysql/my.cnf.`date +%B%d%Y`
```

The configuration line “bind-address = 127.0.0.1” disables the initiation of networking during MySQL startup. Please note that a local connection can still be established to the MySQL server. Look for the bind-address to be set as localhost only.

```
grep bind-address /etc/mysql/my.cnf | grep 127.0.0.1;if [ $? = 0 ];then echo Passed.;
else echo Failed.; fi
```

The next change is to disable the use of the *"LOAD DATA LOCAL INFILE"* command, which will help to prevent unauthorized reading from local files. This is especially important when new SQL injection vulnerabilities in PHP applications are found.

```
sudo sed -i '/skip-external-locking/ilocal-infile=0' /etc/mysql/my.cnf
```

The default administrator username on the MySQL server is **"root"**. Hackers often attempt to gain access to its permissions. To make this task harder, provide it with a long, complex mlitecoinnumeric password. If we change root to something else, then we have to remember two things. What is my root password, and what is the name of the root account. There is much controversy, and it seems to lean toward sticking with root, and a complex password. A good way to generate a complex password is by using uuidgen. Let us grab the fourth and fifth octet of the uuidgen and use that as our root password. We will also store it locally as myroot as a safeguard measure. Of course we would want to secure any and all documents that refer to, or contain any password. We will be altering the root password to something akin to this mlitecoin-numeric string: 97c2-97f4e89fb7c5. There is a pretty good product out there for storing passwords securely entitled password safe. But for now please keep this password locally someplace safe and somewhere you can retrieve it.

```
cd $HOME/mpos

MyRoot=`uuidgen | cut -d'-' -f4,5`

echo${MyRoot} > myroot

mysqladmin -u root -p password ${MyRoot}
```

Drop any unwanted users.

```
mysql -u root -p -e "use mysql;DELETE FROM user WHERE user=\"\";"
```

```
mysql -u root -p -e "FLUSH PRIVILEGES;"
```

Turn off the show databases functionality.

```
sudo sed -i '/skip-external-locking/iskip-show-database' /etc/mysql/my.cnf
```

During the installation procedures, there is a lot of sensitive information that can assist an intruder to assault a database. This information is stored in the server's history and can be very helpful if something goes wrong during the installation. By analyzing the history files, administrators can figure out what has gone wrong and probably fix things up. However, these files are not needed after installation is complete.

```
cat /dev/null > ~/.mysql_history
```

Now we restart the MySQL daemon using its start and stop script.

```
/etc/init.d/mysql restart.
```

If for any reason it fails to start, messages will reside in /var/log/syslog. We are now fairly secure, and ready to proceed with the MPOS MySQL setup. The default database name for MPOS is mpos. It may be a good idea to call this database something other than mpos. We however will know it as the mpos database. So instead of creating the mpos database as mpos, let us consider naming the database **mlitecoin** since this is our mlitecoin installation.

My account is steven. This account name is located in an environment variable LOGNAME. This is the account, in my opinion that we should use to create the database and table structure. We need to create steven as a MySQL user. Note that this will NOT be your root password. We will also give steven the same password as the root user. We can change this in a later chapter. For now it saves us some confusion.

```
#-- Note that 97c2-97f4e89fb7c5 is NOT your root. This is an example.
```

```
cd $HOME/mpos/MPOS
```

```
#-- Create the database previously known as mpos (now mlitecoin).
```

```
mysql -u root -p'97c2-97f4e89fb7c5' -e "CREATE DATABASE mlitecoin;"
```

```
#-- Create the new user steven
```

```
mysql -u root -p'97c2-97f4e89fb7c5' -e "CREATE USER '$LOGNAME'@'localhost' IDENTIFIED BY
'97c2-97f4e89fb7c5';"
```

```
#-- Grant the permission for steven and the mpos database an tables.
```

```
mysql -u root -p'97c2-97f4e89fb7c5' mlitecoin -e "GRANT ALL PRIVILEGES ON mlitecoin.* TO
'$LOGNAME'@'localhost'; FLUSH PRIVILEGES;"
```

```
#-- Note that steven is now allowed privileges similar to root for this new database.
```

```
#-- Import the base MPOS structure using our new user id.
```

```
mysql -p mlitecoin < sql/000_base_structure.sql
```

Finally to complete the MySQL step we must change the global MPOS configuration file to include the database

details.

```
sed -i "s/user'] = 'someuser'/user'] = '$LOGNAME'/g"
$HOME/mpos/MPOS/public/include/config/global.inc.php
```

```
sed -i "s/pass'] = 'somepass'/pass'] = '97c2-97f4e89fb7c5'/g"
$HOME/mpos/MPOS/public/include/config/global.inc.php
```

```
sed -i "s/name'] = 'mpos'/name'] = 'mlitecoin'/g"
$HOME/mpos/MPOS/public/include/config/global.inc.php
```

```
grep db $HOME/mpos/MPOS/public/include/config/global.inc.php
```

The entire script.

```
#!/bin/bash -
#title          :dogon_--_mpos_automation.sh
#description    :Step 06 -- prime the MySQL database
#author         :SJmariogolf
#date           :20140401
#version        :0.0-1
#notes          :
#bash_version   :4.2.45(1)-release
#==============================================================================

MPOS_DB="mlitecoin"



#-- make sure we have the prereqs
sub_prerequisites(){

sudo apt-get install build-essential libboost-all-dev libcurl4-openssl-dev libdb5.1-dev
libdb5.1++-dev mysql-server

return 0
}

runyesorno(){

if [ "${1}" ] ; then
        yesorno="n"
        read -p "Inform: Would you like to run this step? [${2}/${1}] Enter y/n (n)? "
yesorno
                case "$yesorno" in
        y*|Y*)  ${1};;
```

```
                n*|N*)  echo "Skipped";;
        esac
fi

return 0
}

sub_change_mysql_root(){

cd $HOME/mpos
if [ -f myroot ];then
 cp myroot myroot.`date +%B%d%Y`
fi
MyRoot=`uuidgen | cut -d'-' -f4,5`
echo ${MyRoot} > myroot

mysqladmin -u root -p password ${MyRoot}

return 0
}

sub_drop_unwanted_users(){

MYROOT=`cat $HOME/mpos/myroot`
MYROOT="'$MYROOT"
MYROOT="$MYROOT'"
cat <<EOF>drop_unwanted.sh
mysql -v -u root -p$MYROOT -e "use mysql;DELETE FROM user WHERE user=\"\";"
mysql -v -u root -p$MYROOT -e "FLUSH PRIVILEGES;"
EOF
sh drop_unwanted.sh

return 0
}

sub_create_mpos_database(){

cd $HOME/mpos/MPOS
if [ -f "$HOME/mpos/myroot" ];then
```

```
 MYROOT=`cat $HOME/mpos/myroot`
 MYROOT="'$MYROOT"
 MYROOT="$MYROOT'"
 cat <<EOF>create_mpos_database.sh
 cd $HOME/mpos/MPOS
 #-- Create the database previously known as mpos (now $MPOS_DB).
 mysql -v -u root -p$MYROOT -e "CREATE DATABASE $MPOS_DB;"
 #-- Create the new user from $LOGNAME
 mysql -v -u root -p$MYROOT -e "CREATE USER '$LOGNAME'@'localhost' IDENTIFIED BY
$MYROOT;"
 #-- Grant the permission for from $LOGNAME and the mpos database an tables.
 mysql -v -u root -p$MYROOT $MPOS_DB -e "GRANT ALL PRIVILEGES ON $MPOS_DB.* TO
'$LOGNAME'@'localhost'; FLUSH PRIVILEGES;"
 #-- Note that from $LOGNAME is now allowed privileges similar to root for this new
database.
 #-- Import the base MPOS structure using our new user id.
 mysql -v -p$MYROOT $MPOS_DB < sql/000_base_structure.sql
EOF
 sh create_mpos_database.sh
else
 echo "Warn: Cannot determine new mysql root password from myroot."
 mysql -v -p $MPOS_DB < sql/000_base_structure.sql
fi

return 0
}

sub_configure_mpos(){

sed -i "s/user'] = 'someuser'/user'] = '$LOGNAME'/g"
$HOME/mpos/MPOS/public/include/config/global.inc.php
sed -i "s/pass'] = 'somepass'/pass'] = $MYROOT/g"
$HOME/mpos/MPOS/public/include/config/global.inc.php
MPOS_DB="'$MPOS_DB"
MPOS_DB="$MPOS_DB'"
sed -i "s/name'] = 'mpos'/name'] = $MPOS_DB/g"
$HOME/mpos/MPOS/public/include/config/global.inc.php
grep db $HOME/mpos/MPOS/public/include/config/global.inc.php

cp $HOME/mpos/MPOS/public/include/config/global.inc.php .
```

```
return 0
}

sub_skip_local_infile(){

cat <<EOF>mysql.temp.sh
sudo sed -i '/skip-external-locking/ilocal-infile=0' /etc/mysql/my.cnf
EOF
sudo sh mysql.temp.sh

return 0
}

sub_skip_show_database(){

cat <<EOF>mysql.temp.sh
sudo sed -i '/skip-external-locking/iskip-show-database' /etc/mysql/my.cnf
EOF
sudo sh mysql.temp.sh

return 0
}
runyesorno "mkdir -p $HOME/mpos" "Make the $HOME/mpos directory."
runyesorno "cd $HOME/mpos" "Change directory to $HOME/mpos."
runyesorno "sub_prerequisites" "Insure we have the prerequisites."
runyesorno "sudo cp /etc/mysql/my.cnf /etc/mysql/my.cnf.`date +%B%d%Y`" "Backup the mysql
config."
runyesorno "cd $HOME/mpos/MPOS" "Change directory to $HOME/mpos/MPOS."
runyesorno "sub_change_mysql_root" "Change the mysql root password."
runyesorno "sub_skip_local_infile" "Turn off local infiles."
runyesorno "sub_drop_unwanted_users" "Drop any unwanted user from mysql."
runyesorno "sub_skip_show_database" "Turn off show databases."
runyesorno "sub_create_mpos_database" "Create the MPOS database."
runyesorno "/etc/init.d/mysql restart" "Restart the mysql daemon."
runyesorno "sub_configure_mpos" "Configure MPOS."
runyesorno "cat /dev/null > $HOME/.mysql_history" "Clear mysql history."
```

Chapter 8 mail server send mail setup

Enter postfix

We will be setting up postfix as our mail transport agent, and forwarding the smtp to our local agent who will then relay much like the swiftmailer options within the MPOS configuration file. We will first have to install the postfix package and interact with it in a brief manner. The idea is to use postfix as a local agent that will relay to a send mail agent which is external, such as, in this example, gmail. Any number of options exist, however this is quite simple and removes us from much of the overhead of handling email, encoding, delivery, etc.

We start by downloading postfix.

```
sudo apt-get install postfix mailutils libsasl2-2 ca-certificates libsasl2-modules
```

Now we interact with the installation...Choose Internet site:

This is the server and domain that I am currently installing on. Your domain can basically be anything you choose. The domain name should be available, but this is completely up to you. An example could be hashmaster.org. The installation configures an initial main configuration file for the postfix application. We will be making adjustments to this file, so really the name is not so important at this point in the install as it can be changed as desirable.

Proper output is show below...

```
Adding system user `postfix' (UID 117) ...
Adding new user `postfix' (UID 117) with group `postfix' ...
Not creating home directory `/var/spool/postfix'.
Creating /etc/postfix/dynamicmaps.cf
Adding tcp map entry to /etc/postfix/dynamicmaps.cf
Adding sqlite map entry to /etc/postfix/dynamicmaps.cf
Adding group `postdrop' (GID 124) ...
Done.
setting myhostname: steven-M275.home
setting alias maps
setting alias database
setting myorigin
setting destinations: steven-M275.home, localhost.home, , localhost
setting relayhost:
setting mynetworks: 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
setting mailbox_command
setting mailbox_size_limit: 0
```

```
setting recipient_delimiter: +
setting inet_interfaces: all

Postfix is now set up with a default configuration.  If you need to make
changes, edit
/etc/postfix/main.cf (and others) as needed.  To view Postfix configuration
values, see postconf(1).

After modifying main.cf, be sure to run '/etc/init.d/postfix reload'.

Running newaliases
[ ok ] Stopping Postfix Mail Transport Agent: postfix.
[ ok ] Starting Postfix Mail Transport Agent: postfix.
(Reading database ... 127134 files and directories currently installed.)
Removing exim4-base ...
Processing triggers for man-db ...
Selecting previously unselected package guile-1.8-libs.
(Reading database ... 127065 files and directories currently installed.)
Unpacking guile-1.8-libs (from .../guile-1.8-libs_1.8.8+1-8_i386.deb) ...
Selecting previously unselected package libntlm0.
Unpacking libntlm0 (from .../libntlm0_1.2-1_i386.deb) ...
Selecting previously unselected package libgsasl7.
Unpacking libgsasl7 (from .../libgsasl7_1.8.0-2_i386.deb) ...
Selecting previously unselected package mailutils-common.
Unpacking mailutils-common (from .../mailutils-common_1%3a2.99.97-3_all.deb) ...
Selecting previously unselected package libmailutils4.
Unpacking libmailutils4 (from .../libmailutils4_1%3a2.99.97-3_i386.deb) ...
Selecting previously unselected package mailutils.
Unpacking mailutils (from .../mailutils_1%3a2.99.97-3_i386.deb) ...
Processing triggers for man-db ...
Setting up guile-1.8-libs (1.8.8+1-8) ...
Setting up libntlm0 (1.2-1) ...
Setting up libgsasl7 (1.8.0-2) ...
Setting up mailutils-common (1:2.99.97-3) ...
Setting up libmailutils4 (1:2.99.97-3) ...
Setting up mailutils (1:2.99.97-3) ...
update-alternatives: using /usr/bin/frm.mailutils to provide /usr/bin/frm (frm) in auto mode
update-alternatives: using /usr/bin/from.mailutils to provide /usr/bin/from (from) in auto mode
update-alternatives: using /usr/bin/messages.mailutils to provide /usr/bin/messages (messages) in auto mode
```

```
update-alternatives: using /usr/bin/movemail.mailutils to provide /usr/bin/movemail (movemail) in auto mode
update-alternatives: using /usr/bin/readmsg.mailutils to provide /usr/bin/readmsg (readmsg) in auto mode
update-alternatives: using /usr/bin/dotlock.mailutils to provide /usr/bin/dotlock (dotlock) in auto mode
```

Add the forwarding configuration to the postfix main.cf.

```
sudo chmod a+w  /etc/postfix/main.cf
#-- Comment out the old relay
sudo sed -i "s/^relayhost =/#relayhost =/g" /etc/postfix/main.cf
sed -i "s/^mailbox_command = procmail/#mailbox_command = procmail/g" /etc/postfix/main.cf
cat <<EOF>>/etc/postfix/main.cf
relayhost = [smtp.gmail.com]:587
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_CAfile = /etc/postfix/cacert.pem
smtp_use_tls = yes
EOF
sudo chmod a-w  /etc/postfix/main.cf
```

Now we have to have a GMAIL account. This is easily accomplished online. We MUST place our credentials into the postfix sasl_passwd file within the /etc/postfix directory using sudo.

[smtp.gmail.com]:587 yourusername@gmail.com:yourpassword

While I imagine this to be fairly simple, it does however merit some explanation. This is the email username of your GMAIL account that YOU have setup for this site. It can be any available GMAIL account, however, the password is necessary as this is a TLS delivery. These files within the postfix directory are owned by root, and are devoid of write permission, so this is a fairly safe configuration. Now having said that, it is likely a good idea to create a NEW GMAIL account to be used specifically for this server. Maybe hashmaster? If it is available?

```
MYUSER="YourGMAILUserID"
MYPASSWD="YourGMAILPassword"
sudo chmod a+w /etc/postfix/
sudo echo "[smtp.gmail.com]:587    $MYUSER@gmail.com:$MYPASSWD" >
/etc/postfix/sasl_passwd
sudo chmod 400 /etc/postfix/sasl_passwd
sudo postmap /etc/postfix/sasl_passwd
#-- Place a valid premium server cert into postfix. This file can be chosen.
cat /etc/ssl/certs/Thawte_Premium_Server_CA.pem | sudo tee -a /etc/postfix/cacert.pem
```

Next we will reload postfix to see it it functions with our changes in effect.

```
sudo /etc/init.d/postfix reload
```

```
[ ok ] Reloading Postfix configuration...done.
```

And finally, let us test a SMTP delivery.

```
echo "Test mail from postfix" | mail -s "Test Postfix" YourGMAILUserID@gmail.com
```

Due to the sensitive nature of email delivery I thought it best not to place this simple code into a script. Rather, it is better to execute it via the command line. All applicable commands are organized in currier 11 font. Any output highlighted in currier 9 font with a green color is simply the desired output and is placed herein as a reference. I assume it it possible to use alternating email relays, such as Yahoo's Zimbra, or another similar service, however this has not been tested.

Within the /var/log directory output messages can be perused within the mail.log, and syslog files using sudo.

The entire script.

```
#!/bin/bash -
#title          :dogon_--_mpos_automation.sh
#description    :Step 07 -- postfix configuration
#author         :SJmariogolf
#date           :20140401
#version        :0.0-1
#notes          :
#bash_version   :4.2.45(1)-release
#==============================================================================
#-- These MUST be changed
MYGMAILUSER="steven@gmail.com"
MYGMAILPASSWORD="sp00f3d"

#-- make sure we have the prereqs
sub_prerequisites(){

sudo apt-get install postfix mailutils libsasl2-2 ca-certificates libsasl2-modules

return 0
}
```

```
runyesorno(){

if [ "${1}" ] ; then
        yesorno="n"
        read -p "Inform: Would you like to run this step? [${2}/${1}] Enter y/n (n)? "
yesorno
                case "$yesorno" in
        y*|Y*)  ${1};;
                n*|N*)  echo "Skipped";;
        esac
fi

return 0
}

sub_forwarding_postfix(){

sudo chmod a+w  /etc/postfix/main.cf
cat <<EOF>>/etc/postfix/main.cf
#-- `date +%B%d%Y`
relayhost = [smtp.gmail.com]:587
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_CAfile = /etc/postfix/cacert.pem
smtp_use_tls = yes
EOF
sudo chmod a-w  /etc/postfix/main.cf
cat /etc/ssl/certs/Thawte_Premium_Server_CA.pem | sudo tee -a /etc/postfix/cacert.pem
cat >>sasl_passwd<<EOF
[smtp.gmail.com]:587    $MYGMAILUSER:$MYGMAILPASSWORD
EOF
sudo cp sasl_passwd /etc/postfix/
```

```
sudo postmap /etc/postfix/sasl_passwd


return 0
}


runyesorno "mkdir -p $HOME/mpos" "Make the $HOME/mpos directory."
runyesorno "cd $HOME/mpos" "Change directory to $HOME/mpos."
runyesorno "sub_prerequisites" "Insure we have the prerequisites."
runyesorno "sudo cp /etc/postfix/main.cf /etc/postfix/main.cf.`date +%B%d%Y`" "Backup the
main.cf config."
runyesorno "sub_forwarding_postfix" "Change the postfix main for forwarding."
runyesorno "sudo /etc/init.d/postfix restart" "Restart the postfix daemon."
```

Chapter 9 testing with Apache HTTP

Enter MPOS HTTP

Prior to moving into a secure https environment, we can test our application using port 80. This is the http interface of Apache. In a later chapter we will be converting this over to https. We need to copy recursively the MPOS directory with the same user permissions we have already set, to /var/www/ which is the Apache 2 document root.

Change directory to $HOME/mpos, and recursively copy the MPOS subdirectory to the Apache document root.

```
cd $HOME/mpos
sudo cp -rp MPOS /var/www
```

Next we will restart the apache process.

```
sudo /etc/init.d/apache2 restart
```

Prior to navigating to the application using a web browser, there is a couple of items we can use to check out our PHP environment, and also take a look at the current security concerns. By creating a simple phpinfo.php script within the document root, we can navigate to our server by IP address, or 127.0.0.01, and invoke this phpinfo.php.

```
cd $HOME/mpos
cat >phpinfo.php<<EOF
<?php
// Show all information, defaults to INFO_ALL
phpinfo();
?>
```

```
EOF
sudo cp phpinfo.php /var/www
```

Now open a web browser and navigate to 127.0.0.1/phpinfo.php and examine the available PHP information presented to you, as for example.

Additionally there is a package for PHP to download and verify the system from a security perspective. Please download and check out the security warnings and errors triggered by phpsecinfo and fix those before attempting to run MPOS: http://phpsec.org/projects/phpsecinfo/. This is a simple gzipped file that can be directly extracted inside of the /var/www directory.

```
/var/www$ sudo unzip phpsecinfo.zip
```

These files should currently exist within your /var/www directory. Open the browser and navigate to your server IP address or localhost 127.0.0.1/phpsecinfo-20070406.

```
index.html  MPOS  phpinfo.php  phpsecinfo-20070406  phpsecinfo.zip
```

Following the link for More information will yield a postulated solution as is shown above.

It is now time for the first look at the actual application. Open a browser and navigate to the IP address, or 127.0.0.1/MPOS/public.

I feel it is important to point out a possible work around if you are unable to confirm via email confirmations for any particular reason. You can temporarily disable the feature by changing the default from a 1 to a 0 inside admin_settings.inc.php MPOS/public/include/config directory. Itg is a good idea to re-enable this feature prior to considering production.

```
$aSettings['system'][] = array(
  'display' => 'Disable e-mail confirmations', 'type' => 'select',
  'options' => array( 0 => 'No', 1 => 'Yes' ),
  'default' => 0,
  'name' => 'accounts_confirm_email_disabled', 'value' =>
$setting->getValue('accounts_confirm_email_disabled'),
  'tooltip' => 'Should users supply a valid e-mail address upon registration. Requires them to confirm the
address before accounts are activated.'
);
```

Chapter 10 building the Apache SSL HTTPs Server

Enter Apache HTTPs

Apache 2 with SSL encryption is a pretty strong *Secure Sockets Layer* interface. In this chapter we migrate from the TCP Port 80 HTTP to the TCP Port 443 HTTPs, and incorporate a realm of secure transport far and above that of plain text, and non encrypted data transfer. We will be generating self signed certs for use within this chapter. At the end of this chapter some notes directly from the Apache SSL fact are included as reference. SSL will encrypt the transmission to and from the web site and obfuscate personal information such as email addresses, passwords and the like, making this a very important, and rather mandatory contribution.

Change directory to $HOME/mpos directory prior to continuing this setup.

Generate a private key. The pass phrase can be anything you desire.

```
openssl genrsa -des3 -out server.key 1024
```

```
Inform: Would you like to run this step? [Generate a Private Key/openssl genrsa -des3 -out server.key 1024]
Enter y/n (n)? y
Generating RSA private key, 1024 bit long modulus
..............................++++++
.....................++++++
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:
```

Generate a certificate signing request, CSR.

```
openssl req -new -key server.key -out server.csr
```

```
Inform: Would you like to run this step? [Generate a CSR Certificate Signing Request/openssl req -new -key
server.key -out server.csr] Enter y/n (n)? y
Enter pass phrase for server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:VA
Locality Name (eg, city) []:Richmond
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:steven-M275.home
```

```
Email Address []:stevensp00f3d@gmail.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:sjmariogolf
An optional company name []:sjmariogolf
```

Remove the pass phrase from the key.

```
cp server.key server.key.org; openssl rsa -in server.key.org -out server.key
```

Generate the self signed certificate.

```
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

```
Inform: Would you like to run this step? [Generating a Self-Signed Certificate/openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt] Enter y/n (n)? y
Signature ok
subject=/C=US/ST=VA/L=Richmond/O=Internet Widgits Pty Ltd/CN=steven-M275.home/emailAddress=stevensp00f3d@gmail.com
```

Copy the files into the Apache directory.

```
sudo mkdir -p /etc/apache2/ssl
sudo cp server.key /etc/apache2/ssl/server.key
sudo cp server.crt /etc/apache2/ssl/server.crt
```

Configure Apache with the key and cert by adding the new server cert and key into the default-ssl file within the /etc/apache2/sites-available/default-ssl file.

```
cat >apache_ssl_patch<<EOF
42,43c42,44
<       SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
<       SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
---
>       SSLEngine on
>       SSLCertificateFile /etc/apache2/ssl/server.crt
>       SSLCertificateKeyFile /etc/apache2/ssl/server.key
EOF

sudo patch -i apache_ssl_patch /etc/apache2/sites-available/default-ssl
```

Next in order is to change the document root within the default-ssl file from /var/www to /var/www-ssl.

```
sudo sed -i 's/www/www-ssl/' /etc/apache2/sites-available/default-ssl


sudo rm /etc/apache2/sites-enabled/000-default
```

```
sudo ln -s /etc/apache2/sites-available/default-ssl
/etc/apache2/sites-enabled/000-default
```

We then substitute the port 443 for the old port 80 within the ports configuration. We also comment out the first occurrence of the listen.

```
sudo sed -i 's/80/443/g' /etc/apache2/ports.conf
sudo sed '0,/Listen 443/s/Listen 443/#Listen 443/' ports.conf
```

Finally we restart the apache 2 process.

```
sudo /etc/init.d/apache2 stop
sudo /etc/init.d/apache2 start
```

Depending on the installation, the apa che error log can be located using a command similar to this.

```
locate error | grep apa
```

The error log is normally entitled apache error log, or apache ssl error log, and can generally can be located here. /var/log/apache2/error.log . At this juncture the https://127.0.0.1, or by IP address should be available. It is normal to see a Firefox warning for an “untrusted” self-signed certificate. There are any number of certificate signing sites that will offer non self-signed certificates for a nominal fee. The good news is that we are now a secure transport for web traffic to and from our site.

To complete the initial hardening of Apache, You must manually edit the /etc/apache2/sites-available/default-ssl configuration file and lock down the document root directories including the MPOS directory itself as depicted. You can simply replace the block that is within the file with the contents below as an example.

```
        DocumentRoot /var/www-ssl
        <Directory />
                Options None
                AllowOverride None
   order deny,allow
        </Directory>
        <Directory /var/www-ssl/MPOS/public>
                Options Indexes FollowSymLinks
                Order deny,allow
        </Directory>
        <Directory /var/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order deny,allow
                deny from all
```

```
</Directory>
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
<Directory "/usr/lib/cgi-bin">
        AllowOverride None
        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Order allow,deny
        deny from all
</Directory>
```

The following notation is directly from the Apache SSL FAQ, and is placed here solely as a SSL reference.

How do I create a self-signed SSL Certificate for testing purposes?

1. 1.Make sure OpenSSL is installed and in your PATH.
2. 2.Run the following command, to create server.key and server.crt files:

$ openssl req -new -x509 -nodes -out server.crt -keyout server.key

These can be used as follows in your httpd.conf file:

```
SSLCertificateFile    /path/to/this/server.crt
SSLCertificateKeyFile /path/to/this/server.key
```

3. 3.It is important that you are aware that this server.key does *not* have any passphrase. To add a passphrase to the key, you should run the following command, and enter & verify the passphrase as requested.

$ openssl rsa -des3 -in server.key -out server.key.new

$ mv server.key.new server.key

Please backup the server.key file, and the passphrase you entered, in a secure location.

How do I create a real SSL Certificate?

Here is a step-by-step description:

1. 1.Make sure OpenSSL is installed and in your PATH.
2. 2.Create a RSA private key for your Apache server (will be Triple-DES encrypted and PEM formatted):

$ openssl genrsa -des3 -out server.key 1024

Please backup this server.key file and the pass-phrase you entered in a secure location. You can see the details of this RSA private key by using the command:

$ openssl rsa -noout -text -in server.key

If necessary, you can also create a decrypted PEM version (not recommended) of this RSA private key with:

$ openssl rsa -in server.key -out server.key.unsecure

3. 3.Create a Certificate Signing Request (CSR) with the server RSA private key (output will be PEM formatted):

$ openssl req -new -key server.key -out server.csr

Make sure you enter the FQDN ("Fully Qualified Domain Name") of the server when OpenSSL prompts you for the "CommonName", i.e. when you generate a CSR for a website which will be later accessed via https://www.foo.dom/, enter "www.foo.dom" here. You can see the details of this CSR by using

$ openssl req -noout -text -in server.csr

4. 4.You now have to send this Certificate Signing Request (CSR) to a Certifying Authority (CA) to be signed. Once the CSR has been signed, you will have a real Certificate, which can be used by Apache. You can have a CSR signed by a commercial CA, or you can create your own CA to sign it.

Commercial CAs usually ask you to post the CSR into a web form, pay for the signing, and then send a signed Certificate, which you can store in a server.crt file. For more information about commercial CAs see the following locations:

1. 1.Verisign

http://digitalid.verisign.com/server/apacheNotice.htm

2. 2.Thawte

http://www.thawte.com/

3. 3.CertiSign Certificadora Digital Ltda.

http://www.certisign.com.br

4. 4.IKS GmbH

http://www.iks-jena.de/leistungen/ca/

5. 5.Uptime Commerce Ltd.

http://www.uptimecommerce.com

6. 6.BelSign NV/SA

http://www.belsign.be

For details on how to create your own CA, and use this to sign a CSR, see below.

Once your CSR has been signed, you can see the details of the Certificate as follows:

$ openssl x509 -noout -text -in server.crt

5. 5.You should now have two files: server.key and server.crt. These can be used as follows in your httpd.conf file:

SSLCertificateFile /path/to/this/server.crt
SSLCertificateKeyFile /path/to/this/server.key

The server.csr file is no longer needed.

How do I create and use my own Certificate Authority (CA)?

The short answer is to use the CA.sh or CA.pl script provided by OpenSSL. Unless you have a good reason not to, you should use these for preference. If you cannot, you can create a self-signed Certificate as follows:

1. 1.Create a RSA private key for your server (will be Triple-DES encrypted and PEM formatted):

$ openssl genrsa -des3 -out server.key 1024

Please backup this host.key file and the pass-phrase you entered in a secure location. You can see the details of this RSA private key by using the command:

$ openssl rsa -noout -text -in server.key

If necessary, you can also create a decrypted PEM version (not recommended) of this RSA private key with:

$ openssl rsa -in server.key -out server.key.unsecure

2. 2.Create a self-signed Certificate (X509 structure) with the RSA key you just created (output will be PEM formatted):

$ openssl req -new -x509 -nodes -sha1 -days 365 -key server.key -out server.crt

This signs the server CSR and results in a server.crt file.

You can see the details of this Certificate using:

$ openssl x509 -noout -text -in server.crt

Chapter 11 putting it All Together with Litecoin

Enter the MPOS Application Suite

This chapter is where we put it all together, so to speak and introduce the MPOS application with Litecoin. In the chapters that follow, Dogecoin, then Infinitecoin ad eMark are configured as the mining crypto-currency, and by example, should be sufficient for any coin. I chose Dogecoin and Infinitecoin for specific reasons. Dogecoin is a very popular coin, and Infinitecoin, well, it is rather difficult indeed to find any information at all on Infinitecoin. Most information states that it cannot even be used with a Stratum. So, for a nice challenge, I chose Infinitecoin.

In actuality, simply copying recursively, preserving the ownership of the MPOS files, it is a basic “cp -rp MPOS” file copy from the working directory to the /var/www, or /var/www-ssl, Apache receiving directory. There are a couple of items up front though worth mentioning. As with most things we try to accomplish, we seldom encounter them devoid of problems. With this in mind, this may be a good place re-elaborate the Apache, and Postfix error logs. These logs exist in the /var/log directory as apache/error.log, and mail.log respectively. These logs are our friends, as they contain dialog, and often fixes to the problems we encounter. I'd like this opportunity to point out a couple of problems I had in the beginning. One problem is the date is not set within the php.ini file, and as such errors are encountered in several places within the application leaving a blank white screen. To locate the php.ini, issue a command line “locate php.ini”, there is generally two of these files found. The actual php.ini file read by PHP can be found using a simple php script phpinfo discussed in a previous chapter, however the fix is to edit the php.ini file and enter the time and time zone, as for example below, making certain it is a valid Time Zone. Valid time zones can be found with simple Google searches, like “php time zone.”

```
[Date]
; Defines the default timezone used by the date functions
; http://php.net/date.timezone
date.timezone = America/New_York
```

Another common problem is permissions on very certain files where Apache requires write privileges. This can easily be rectified using the chown below:

```
cd /var/www, or /var/www-ssl
sudo chown -R www-data public/templates/compile public/templates/cache logs
```

With this information at hand we are ready to copy the MPOS files into our /var/www-ssl MPOS target directory.

```
cd $HOME/mpos
```

```
sudo cp -rp MPOS /var/www-ssl
cd /var/www-ssl
sudo chown -R www-data public/templates/compile public/templates/cache logs
```

And we are now ready to access the MPOS Application. I have always found that the first user I add into the system fails to send an email. This creates a problem whereby we have to do some configuration in our application as the admin, but we cannot mine, because we cannot receive the email. This is a minor inconvenience, and is easily worked around, and besides, we have the mysql access, and can change the database if and when it is required. The general idea is to add in your admin account, or add an account and then use mysql to make this account an admin, then perform some minor configuration within the GUI that enables the sendmail functionality. Once this is accomplished it's “we are in business” time. So let's look at the application.

To launch the MPOS application, open your browser and navigate to the IP address, or 127.0.0.1 using for example, https://127.0.0.1/MPOS/public. The main entry screen should appear. If not, then we need to check the Apache error.log. The resulting page should resemble a normal mining web site complete with guest, sign in, and sign up as is depicted.

The application is fairly straight forward in its design, and will not take long to get a handle on. It is important though to remember the global configuration file within the /var/www-ssl/MPOS/public/include/config directory. This is where the database, and the rpcuser, and other pertinent global configuration data is derived, making this a fairly important file to remember. Here we must create an admin user. The admin user is an email address configured within the global configuration file previously mentioned. Using this admin user we can perform some post configuration necessary for the interface presentation, the coin type and mining denomination, email addresses and the like. Along the left hand side of the screen you will notice as admin, and ADMIN PANEL section. It is the Settings within this panel that we must traverse as shown.

The email confirmations tab is under Systems. Setting this to Yes will disable email confirmations prior to continuing. It is a good practice to disable this prior to production, then re-enable it once any “kinks” are worked out of the mail delivery functionality.

It is quite simply a matter of clicking through the sections along the top of the page, and completing the information such as a valid email address, or a pool name. After the configuration is saved, the email functionality should work without a problem. This application should be considered no different than a public site whereby we have to subscribe, then log in, then create our workers.

The Stratum

The apposing mechanism to the web interface is the Stratum. Before we delve to deeply into the application we

MUST properly configure and start up the Stratum. The MPOS application requires both the Web interface and the Stratum. Both of these elements work together, hopefully, harmoniously, thus this section will also detail the Stratum. In later Chapters we will see more of the application, and how to alter the coin type, and even the interface itself. But prior to moving on, we are required to enable a proper Stratum, and hopefully by the end of this Chapter we will have the Web interface, and the Stratum proxy up and running, and communicating with each other in harmony.

As we previously discussed, the Stratum is an overlay network on the top of coin daemon P2P protocol, creating simplified facade for lightweight clients and hiding unnecessary complexity of decentralized protocol. The Stratum is configured to interface with the TRUSTED rpc port of the con base daemon. In Litecoin this TCP port is 9332. When the Stratum is started it creates a resource pool, or a proxy between the coin daemon and the stratum+tcp listen TCP port, which defaults to 3333. So when the stratum is started by default we will establish a TCP listen locally as stratum+tcp://127.0.0.1:3333. This port is configurable within the Stratum's conf/config.py configuration file.

The Stratum software should be installed and configured at this juncture, but we might as well point out the installation here, and reiterate some of the ore important configuration items. To install the Stratum we follow these steps for Litecoin, or any other SCRYPT based coin. For SHA256d based coins we install the normal slush0 Stratum as I will point out.

download the python scrypt library, the stratum core, and the stratum mining proxy into your $HOME/mpos directory. A note here, is that it is good to work out of a single directory like mpos. This consolidates the sub directories and files within one “folder.” The following code with download the necessary files into their proper sub directories.

For SCRYPT based coins.

```
cd $HOME/mpos
git clone https://github.com/Tydus/litecoin_scrypt.git
git clone https://github.com/ahmedbodi/stratum-mining.git
git clone https://github.com/ahmedbodi/stratum.git
sudo easy_install -U distribute
```

Now continue with the initialization, setup and on to configuration.

```
cd $HOME/mpos/stratum-mining
git submodule init
git submodule update
cd $HOME/mpos/stratum-mining/externals/litecoin_scrypt
sudo python setup.py install
cd $HOME/mpos/stratum-mining/externals/stratum
sudo python setup.py install
```

For SHA256d based coins.

```
git clone https://github.com/slush0/stratum.git
git clone https://github.com/generalfault/stratum-mining.git
sudo apt-get install python-dev
sudo apt-get install python-setuptools
sudo apt-get install python-crypto
sudo easy_install -U distribute
sudo easy_install stratum
```

I may suggest, as I do, to rename the stratum-mining directories as stratum-mining-scrypt, then stratum-mining-sha256d, or even stratum-mining-dogecoin, and stratum-mining-bitcoin, etc. for an easy way to remember. Now the syntax of the config.py depends on the Stratum. For the Litecoin stratum, the KEYWORDS within the config.py are prefixed COINDAEMON, for other installations, it may very well be LITECOIN, or BITCOIN, however the contents of the config.py are fairly well consistent. Having said that, while starting the stratum mining proxy using the twistd -ny launcher.tac command Error messages will inform you as to whether, or not you've got the right prefix configured within your config.py. It is an easy global change from BITCOIN to COINDAEMON using the vi command syntax: ":1,$s/BITCOIN/COINDAEMON/g". This is a fairly redundant task when switching up coinage.

So now we will discuss some of the more important configurable key value pairs within the config.py file starting with the CENTRAL WALLET. This is an important item so I do speak to it rather over and over. The central wallet is normally the default coin address within your installation. Although it can be some other address, it MUST be within the wallet on this server, or no remuneration.

To glean the default address use the coindaemon command getaccountaddress "". Place this address, until we know better, into the config.py as below substituting your actual address as indicated.

```
CENTRAL_WALLET = 'Sup3rC@ll@fr@j@l1sticExpeAll1d0t1ius'    # local bitcoin address where money goes
```

Next is the RPC. These entries govern the rpc communication to the coindaemon. If you do not know the rpc port, Google can easily help you out. For Litecoin it is 9332. Use the Linux command "cat ~/.litecoin/litecoin.conf" to examine the contents of the litecoin.conf. These will be your entries for RPC.

```
COINDAEMON_TRUSTED_HOST = 'localhost'
COINDAEMON_TRUSTED_PORT = 9332
COINDAEMON_TRUSTED_USER = 'litecoinrpc'
COINDAEMON_TRUSTED_PASSWORD = 'edSp00f3d-408c-8200-Sp00f3dad5'
```

The prefix of the key value pairs totally depends on the Stratum. Under all of the coins I have implemented I have

ONLY used LITECOIN, COINDAEMON, and BITCOIN. Perhaps if we create ONE config.py with all key values replicated for all three keys? Just a thought.

For MPOS there is a DB section that requires our attention for the MySQL user, Database, and DB Password. For this example, the user can be root, or as we have established, your account. The database will then be mlitecoin, and the password will be set accordingly as is indicated.

```
# ******************** Database  **********************


# MySQL
DATABASE_DRIVER = 'mysql'
DATABASE_EXTEND = False
DB_MYSQL_HOST = 'localhost'
DB_MYSQL_DBNAME = 'mlitecoin'
DB_MYSQL_USER = 'steven'
DB_MYSQL_PASS = 'b0e8-Sp00f3dfd'
DB_MYSQL_PORT = 3306
```

This is sufficiently enough configuration to enable us to progress to starting the Stratum. To start the Stratum, we change directory to stratum-mining. Insure we have configured the config.py file, and then start the Stratum daemon process using the command illustrated.

```
cd stratum-mining
twistd -ny launcher.tac
```

Note, if the coindaemon process is downloading coins, the Stratum will sleep 30 seconds and try and reconnect. It will perform this action until the coindaemon has successfully downloaded all block chains and the data within is “normalized.” With a new coin, this process from the coin installation to normalization can take up to 2 days. It is a good idea to install the coindaemon ahead of time, and let the data normalize. You can run the Bitcoin, and Litecoin, and Dogecoin daemons all on the same server, as they each utilize different RPC ports, and will not generally affect one another. They simply will compete for system resources as they normalize their respective block chains.

When the Stratum is up and operational, messages similar in nature to these below will serve as an indicator. Additionally, the MPOS web interface will now be able to communicate with the Stratum, and any screen error messages pertaining to “cannot communicate with stratum...” will disappear.

```
2014-04-09 10:09:35,029 INFO block_updater # Checking for new block.
2014-04-09 10:09:40,029 INFO block_updater # Checking for new block.
2014-04-09 10:09:45,029 INFO block_updater # Checking for new block.
2014-04-09 10:09:50,029 INFO block_updater # Checking for new block.
```

```
2014-04-09 10:09:55,029 INFO block_updater # Checking for new block.
2014-04-09 10:10:00,029 INFO block_updater # Checking for new block.
2014-04-09 10:10:05,029 INFO block_updater # Checking for new block.
2014-04-09 10:10:05,038 INFO block_updater # Merkle update! Prevhash: 5526d49bc87a5460912e21883d761e570379939437b0eb0a45b187eed80e2193
2014-04-09 10:10:05,051 INFO template_registry # New template for 5526d49bc87a5460912e21883d761e570379939437b0eb0a45b187eed80e2193
2014-04-09 10:10:05,052 INFO subscription # BROADCASTED to 3 connections in 0.000 sec
2014-04-09 10:10:05,053 INFO template_registry # Update finished, 0.003 sec, 3 txes
2014-04-09 10:10:10,041 INFO block_updater # Checking for new block.
2014-04-09 10:10:15,041 INFO block_updater # Checking for new block.
2014-04-09 10:10:15,066 INFO BasicShareLimiter # Checking Retarget for sjmariogolf55.p1 (2) avg. 22 target 30+-15
2014-04-09 10:10:15,067 INFO interfaces # 000000007a6d85c316dc9be33c8ea8ce41d187e56e716d9f950e8d269122f06f (2) valid sjmariogolf55.p1
2014-04-09 10:10:20,041 INFO block_updater # Checking for new block.
2014-04-09 10:10:25,041 INFO block_updater # Checking for new block.
2014-04-09 10:10:26,166 INFO interfaces # 0000000001efbc9f98538ef44173b979cafde7
```

At this juncture we are able to mine normally using cpuminer, cgminer with GPU, etc. just as though we were pointing to a stratum+tcp out on the internet. We use the exact same syntax, substituting the URL from an Internet site to our own IP address, or in this case as localhost:

```
./cgminer -o stratum+tcp://127.0.0.1:3333   -O yourworker:yourpassword
```

Navigating back to the MPOS web GUI, we should now be able to see active workers on our new Mining site.

It may be a good time at this juncture to use the admin interface and surf the new site, altering the site name, perhaps setting the message of the day, and to investigate what we can alter on the interface using the admin panel and the available configuration items. I suggest you play with the interface learning as much about it as is possible by trial and error. Additionally, this may be a good place to take a tar backup. This should be a good base line starting point for the site, and a backup may do us well in the future. I cannot hurt. For backups I use what works, and that is tar zcvf somefilename.tar.gz $HOME/mpos, then secure copy this backup off the server. As well the coindaemon can be backed up, or simply stopped, the scp -rp $HOME/.coindaemon from this server to another server. There exists hundreds of comments and articles on proper backup. Far too many to mention here, however tons of information is available to guide you in the ways of tar, and scp however it is really quite simple as for example...if I want to scp the entire litecoin structure replicating it on another server...I may issue these commands

from 192.168.1.2.

```
litecoind stop
cd $HOME
scp -rp .litecoin 192.168.1.3:/home/steven
```

To backup the mpos directory using tar...

```
cd $HOME
mkdir backup
tar zcvf backup/mpos.`date +%B%d%Y`.tar.gz mpos
```

I think I have done my best to walk with you through this first MPOS and stratum interchange, however it is rarely without problems. For this reason I am including an Appendix detailing some of the problems encountered, and the postulated, or actual fixes. Everything is easy once it is understood, and perhaps performed a few dozen times. I am sure this is the case for the heart surgeon as well after their 100th bypass. What I am saying here is that this is not a simple process, and it will take some iterations, and perhaps changes, but nothing, as they say is worthy if not for a struggle. Consider the experience gained by performing these tasks, and the resulting personal gains achieved by accomplishing them. And with these tools at your side, I may note, that you can make a pretty darn good living utilizing them.

Next on the agenda is to “switch up” the coindaemon from Litecoin to Dogecoin. This example can be used for any coin. Keep in mind though, the Stratum and the MPOS application will have a few configuration changes when we switch, but I think you will be surprised at how easy it is to accomplish.

Chapter 12 converting over to Dogecoin

Enter Dogecoin and the MPOS Application Suite

Installing any coindaemon is basically following the same exact, or very similar procedures as all daemons are derived form the same source base. To install any coin, first identify the source for that coin using Google and say for instance: “github dogecoin.” Navigate to the github site and copy the URL for cloning the source. As for example: https://github.com/dogecoin/dogecoin.git. These instructions may be beneath your level of extertise, and for that I do apologize, however to install most any coin follow these basic instructions.

Change directory to $HOME/mpos and git clone the source. Then change directory into the coin/src directory, and make -f makefile.unix. Now copy the resulting coindaemon “d” executable to /usr/bin using sudo. Create the

coindaemon.conf file within the $HOME/.coindaemon directory, and start the daemon.

As in the example directly posted from the dogecoin github readme file.

```
sudo apt-get install build-essential \
                     libssl-dev \
                     libdb5.1++-dev \
                     libboost-all-dev \
                     libqrencode-dev \
                     libminiupnpc-dev
git clone https://github.com/dogecoin/dogecoin.git
cd src/
make -f makefile.unix USE_UPNP=1 USE_IPV6=1 USE_QRCODE=1
```

Sample dogecoin.conf file.

```
rpcuser=doge
rpcpassword=wow
rpcallowip=192.168.1.*
addnode=67.205.20.10
addnode=146.185.181.114
addnode=95.85.29.144
addnode=78.46.57.132
addnode=188.165.19.28
addnode=162.243.113.110
rpcport=22555
server=1
daemon=1
```

Now start the daemon with `dogecoind -daemon`, and wait 12 to 24 hours for the blockchain to normalize. The Dogecoin daemon has a new rpc port indicated above as 22555. Likely it will have a new rpcpassword, and definitely a new rpcuser. These values will be directly changed in both of the configuration files for MPOS, and Stratum. Additionally we will want to create a new database, possibly named mdogecoin, and import the sql from MPOS into this new database. The new database name will also be change in both the MPOS configuration, and the config.py. Very few changes are necessary within the global configuration for the MPOS, however they should be fairly well explained below. Let us now take this step by step.

Step one is to install the coin daemon. The steps above adequately describe how to accomplish this, and are transferable to most any coin daemon. The trick is to install these daemons prior to moving forward with MPOS, or

the Stratum. The coindaemons you install is completely up to you. Once your coin blockchains are “normalized,” the process occupy very little system resources. However, saying that, some of the coin daemons still utilize the gen=1 flag in the coindaemon.conf file. Some coindaemon processes actually incorporate their own internal mining mechanisms, and by setting the gen=1 flag into the conf file the daemon will actually mine for coins. This may result in some rather intensive cpu and memory usage, and is generally not recommended. However it is rather fun to experiment with.

Step two is to create a new database for this coin. In this example it is the Dogecoin, so the resulting database will be mdogecoin. This presents a logical separation from Litecoin to Dogecoin knowing that this will require us to re enter the admins, users, and workers.

```
cd $HOME/mpos/MPOS
mysql -u root -p -e 'CREATE DATABSE mdogecoin;'
sudo mysql -p mdogecoin < sql/000_base_structure.sql
```

Step three is to edit the config.py. It would, in my opinion behoove us copy recursively the stratum-proxy directory into another directory, namely stratum-proxy-dogecoin. Here we would edit the config.py sections for the wallet, rpc, and database accordingly. The RPC Port for dogecoin is 22555.

```
# ******************** BASIC SETTINGS ***************
# These are the MUST BE SET parameters!


CENTRAL_WALLET = 'P3psiexc33dsCok3andIsYumy4u85sp00f3dwif'    # local coin address where
money goes


COINDAEMON_TRUSTED_HOST = 'localhost'
COINDAEMON_TRUSTED_PORT = 22555
COINDAEMON_TRUSTED_USER = 'dogecoinrpc'
COINDAEMON_TRUSTED_PASSWORD = 'esp00f3d-b7d1-408c-8200-sp00f3d8ad5'
```

This time we are using the stratum from ahmedbodi, since this is a SCRYPT coin as per the instructions indicated below which calls for a COINDAEMON key value in the config.py.

```
cd $HOME/mpos
git clone https://github.com/Tydus/litecoin_scrypt.git
git clone https://github.com/ahmedbodi/stratum-mining.git
git clone https://github.com/ahmedbodi/stratum.git
cd stratum-mining
git submodule init
git submodule update
```

```
cd externals/litecoin_scrypt
sudo python setup.py install

cd $HOME/mpos
cd stratum-mining/externals/stratum
sudo python setup.py install
```

In actuality, there exists any number of incarnations to the Stratum on the github site. Basically they are the same, or similar in nature.

Do not forget to alter the database portion if the config.py to alter the database name from mlitecoin to mdogecoin, and we are done with the Stratum.

Step 4 is to alter the global.inc.php within the /var/www-ssl/MPOS/public/include/config directory. Changing a few basic contructs.

The coin algorithm will stay as scrypt as Dogecoin is a SCRYPT algo.

```
/**
  * Coin Algorithm
  *  Algorithm used by this coin, sha256d or scrypt
  *   https://github.com/MPOS/php-mpos/wiki/Config-Setup#wiki-algorithm
  **/
$config['algorithm'] = 'scrypt';
```

The database name will change to mdogecoin.

```
/**
 * Database configuration
 *  MySQL database configuration
 *   https://github.com/MPOS/php-mpos/wiki/Config-Setup#wiki-database-configuration
 **/
$config['db']['host'] = 'localhost';
$config['db']['user'] = 'steven';
$config['db']['pass'] = 'sp00f3d';
$config['db']['port'] = 3306;
$config['db']['name'] = 'mdogecoin';
```

The wallet and RPC information will certainly change in accordance with YOUR particular $HOME/.dogecoin/dogecoin.conf file contents.

```
/**
```

```
 * Local wallet RPC
 *  RPC configuration for your daemon/wallet
 *   https://github.com/MPOS/php-mpos/wiki/Config-Setup#wiki-local-wallet-rpc
 **/
$config['wallet']['type'] = 'http';
$config['wallet']['host'] = 'localhost:22555';
$config['wallet']['username'] = 'dogecoinrpc';
$config['wallet']['password'] = 'Sp00f3d-b7d1-408c-8200-Sp00f3d18ad5';
```

Change the ticker API interface.

```
/**
 * Ticker API
 *  Fetch exchange rates via an API
 *   https://github.com/MPOS/php-mpos/wiki/Config-Setup#wiki-ticker-api
 **/
$config['price']['enabled'] = true;
$config['price']['url'] = 'https://www.allcoin.com';
$config['price']['target'] = '/trade/doge_btc';
$config['price']['currency'] = 'USD';
```

And change the currency.

```
/**
 * Currency
 *  Shorthand name for the currency
 *   https://github.com/MPOS/php-mpos/wiki/Config-Setup#wiki-currency
 */
$config['currency'] = 'DGC';
```

Now you can once again start the Stratum, and access the MPOS application using your web browser. As indicated.

Your email address is still the administrator, so go ahead and sign up, then log in to MPOS using this address. You will have to perform the same configuration settings similarly to Litecoin, however this time it is Dogecoin. Use the ADMIN PANEL interface to customize the web site as you see fit. This experience only adds to the overall experience and education vital for running a production site.

Next create your workers, and mine...

I prefer to use cgminer as the miner for scrypt using a similar cgminer shell script and cgminer.conf file as shown below.

The shell script...

```
#!/bin/sh
export DISPLAY=:0
export GPU_MAX_ALLOC_PERCENT=100
export GPU_USE_SYNC_OBJECTS=1
./cgminer -c cgminer.conf.local
```

The cgminer.conf file...

```
{
"pools" : [
        {
                "url" : "stratum+tcp://192.168.1.16:3333",
                "user" : "sjmariogolf.d1",
                "pass" : "peeb"
        }
]
,
"balance" : true,
"intensity" : "9,9",
"thread-concurrency" : "0,0",
"temp-cutoff" : "90,90",
"temp-overheat" : "85,85",
"temp-target" : "68,68",
"api-listen" : true,
"api-mcast-port" : "4028",
"api-port" : "4028",
"auto-fan" : true,
"no-pool-disable" : true,
"no-submit-stale" : true,
"queue" : "0",
```

```
"scrypt" : true,
"kernel-path" : "/usr/local/bin",
"api-allow" : "W:127.0.0.1"
}
```

Within the Stratum's conf/config.py, there is a configuration variable that determines the Listen port for the coin. As in the example below this can be most any port you desire. Stick with 4 digit port combinations like 3333, 4444, 5555, 6666, and so on, to begin with for an easier time in remembering them. The MPOS global include file and the Stratum's Listen port must be set to the same number.

```
# config.py
# Port used for Socket transport. Use 'None' for disabling the transport.
LISTEN_SOCKET_TRANSPORT = 7777
# global.inc.php
/**
 * Getting Started Config
 *  Shown to users in the 'Getting Started' section
 *   https://github.com/MPOS/php-mpos/wiki/Config-Setup#wiki-getting-started
 **/
$config['gettingstarted']['coinname'] = 'dogecoin';
$config['gettingstarted']['coinurl'] = '127.0.0.1';
$config['gettingstarted']['stratumurl'] = '127.0.0.1';
$config['gettingstarted']['stratumport'] = '7777';
```

And we are mining.

Chapter 13 infinitecoin possibilities

Enter the MPOS and infinitecoin

I was searching alternate scrypt coins when I located infinitecoin. It is trading actively and at or around the same price as Dogecoin. I had no idea when a block is solved that it pays out at 1,200 coins. That was a pleasant surprise. The infinitecoin seems to be a fairly easily mined coin, and is a good choice for mining, and the rewards associated with solving blocks. You can download and compile basically any coin daemon and client. They are basically all reworks of the Bitcoin source base. The rpc information changes, and as a result, the MPOS configuration and stratum config.py must be changed according to the rpc port, name and password. It is good practice to copy recursively the stratum-mining into, say for example, stratum-mining-infinitecoin, for the infinitecoin. Additionally

to create a new database within MySQL, for example infinite. Now having said that, the coindaemon.conf file determines the listen rpc port for the daemon. You can choose any 4 digit listen port. The same applies for the stratum listen port. It is simply a matter of getting them all on your rpc, and stratum listen. Thus as many coins as you desire, within the limits of your hardware, can be serviceable on your system or systems. I do hope this is clear, because it is a lot of fun to explore new coins this way. Let's go over it in detail here using the infinitecoin as the example. Then whether scrypt or sha256d, pow or pos, or pow+pos, they are all basically the same and are clones of one another,and setup is a “miner,” pardon the pun, thing. Having said that, there is no time like the present to go over this in excruciating detail which will serve to reinforce the required information necessary to retain the breadth, than depth herein. Iteration, and reiteration are often keys to absorbing the vast array of informational constructs of any large system, and it is a known fact that it requires “on average,” seven times of performing *anything*, before it is understood.

Shall we play a game...

Step one is to download the infinitecoin daemon and compile it locally. To accomplish this task is fairly redundant and very straight forward by this point, however lets proceed. Download the source code via the git command and github.

```
cd $HOME/mpos
git clone https://github.com/infinitecoin/infinitecoin.git
cd infinitecoin/src
make -f makefile.unix
sudo cp infinitecoind /usr/bin
infinitecoind
```

Step two is to edit the $HOME/.infinitcoin/infinitecoin.conf configuration file for our rpc user and password and subsequent addnodes and listen port.

```
mkdir -p $HOME/.infinitecoin
cd $HOME/.infinitecoin
cat <<EOF > infinitecoin.conf
rpcuser=infinitecoinrpc
rpcpassword=esp00f3d-b7d1-408c-8200-sp00f3dad5
rpcallowip=192.168.1.*
addnode=5.249.152.159:9321
rpcport=9322
server=1
daemon=1
EOF
```

Step three is to start the infinitecoin daemon process and normalize the block count. This process will generally

take between 6 hours, but may take up to 3 days. Google “infinitecoin block count” for an accurate, “normalized,” block count.

```
infinitecoind getinfo
{
    "version" : 1080700,
    "protocolversion" : 69002,
    "walletversion" : 60000,
    "balance" : 13512.00000000,
    "blocks" : 836064,
    "connections" : 10,
    "proxy" : "",
    "difficulty" : 0.55975982,
    "testnet" : false,
    "keypoololdest" : 1397084575,
    "keypoolsize" : 101,
    "paytxfee" : 10.00000000,
    "mininput" : 0.01000000,
    "errors" : "Mandatory Update to v1.8.7"
}
```

Step four is to create our new database for the infinitecoin and the MPOS application.

```
cd $HOME/mpos/MPOS
mysql -v -u root -p infinite < sql/000_base_structure.sql
```

Step five is the prepare the stratum as a scrypt based proxy for the infinitecoin daemon listen specifying our own listen port for the stratum+tcp://.

As a general rule of thumb, I personally copy recursively preserving the file permissions os an existing scrypt based stratum directory naming this new directory stratum-mining-infinitecoind. Then proceed to make the “few” necessary changes to the copied conf/config.py file. There are five sections that require mandatory changes;

- •The CENTRAL_WALLET, obtain this information using the coindaemond getaccountaddress “” daemon command.
- •The TRUSTED_USER and TRUSTED_PASSWORD. These are derived directly from the $HOME/.infinitecoin.conf configuration file.
- •The ALGO which in this case is “scrypt”.
- •The LISTEN_SOCKET_TRANSPORT, which happens to be externalized into the resulting proxy listen port for the stratum+tcp:// on this server for this coin. Any 4 digits will suffice, and this is completely at your discretion. Example: 7777.

- •The DB_MYSQL_DBNAME which in this case should match the database we created. The database user and password are know values at this juncture and likely will not change. However if root is not the MySQL user, a table GRANT will be necessary for the selected user, which happens to be detailed in the MySQL chapter of this book.

As they say with “Grey Poupon,” but of course, this is your stratum, and it is the bridge and the gateway to the daemon process and consequently the interface between the miner software and the MPOS application. You are free to adjust the values within to your liking. Values like for instance, the variable difference, or the difference starting point. It is completely configurable, and adjusting the values is a fun and interesting way to test varying rig configurations.

Step six isto add block notificaton into the stratum. This is a good time to introduce a new concept into the stratum, the block notification. Within the stratum directory there exists a scripts directory, and within this directory there is one, a SHA encryption script specifically for generating the hashed ADMIN_PASSWORD, and two, the blocknotify.sh itself. What we are going to establish here is to add a new process that will synchronize the stratum and coindaemon processes when a block is found, eliminating needless processing time otherwise spent on mining previously solved blocks. We will one..., generate a new hashed password for our stratum, then two..., place this hashed value within our config.py file, then three..., add in a new “blocknotify” key, value attribute into our coindaemon.conf file. The steps to accomplish this are exemplified here using the infinitecoin stratum scripts.

```
cd $HOME/mpos/stratum-mining-infinitecoin/scripts #-- directory (or your stratum directory for this coin.)

uuidgen >@dmin

chmod a-rw @dmin

./generateAdminHash.sh `sudo cat @dmin`
```

Now add the resulting hashed string to the stratum's conf/config.py file as per the example below substituting your hashed response.

```
# ADMIN

ADMIN_PASSWORD_SHA256 = '295d6bsp00f3de406fb5ddEXAMPLE00f3dcf1fe5eebf'
```

Next edit the $HOME/.infinitecoin/infinitecoin.conf file adding the key, value attribute for the blocknotify. The password is the un-hashed ADMIN password as is otherwise, “hashed” within the config.py. Substitute your fully qualified path, and port for the stratum listen port externalized for this coin.

```
blocknotify=/home/steven/mpos/stratum-mining-infinitecoin/scripts/blocknotify.sh --password sp00f3d5-b392-6fdsp00f3d --port 5555
```

To summarize the block notify, it is two additional steps, as, generating the hashed secret password, and store this hashed value into our config.py file, then add a new attribute into the coindaemons coindaemon.conf file using the “un-hased” password and stratum listen port.

As a “rule of thumb,” and personal preference you can use the nohup command to launch your stratum. Using nohup will break the connection to your shell and will pass control to the kernel. The stratum will continue to

execute, regardless. However, without automation, like cron, the stratum will require a restart if the server is rebooted.

To start the stratum, we navigate to the stratum's directory and use the twistd -ny launcher.tac, or the twistd -y launcher.tac command for interactive, and or non interactive start of the stratum daemon process. Prefixing the stratum start command with “nohup sudo twi...” nohups the daemon, breaking the processes ties to your shell allowing it to execute even after you disconnect.

```
nohup twistd -y launcher.tac &
```

Both the nohup.out, and the log/stratum.log files are now at your disposal. It may be a good idea to first run the stratum normally, with the “-ny” flags and output to the screen, then when bug free, nohup the process. So go ahead and start the stratum process now.

Step seven is to perform the relatively few MPOS configuration steps to convert the application from the previous coin to infinitecoin. As a rule of thumb it may be a good idea to copy the global.inc.php into a saved version named by the previous coin. For example, within the /var/www-ssl/MPOS/public/include/config directory, copy the global.inc.php file to global.inc.php.dogecoin. Then using sudo edit the global.inc.php. These are the few changes necessary.

One fairly simple way of demonstrating the differences in the configuration is to perform a diff command. The output is below, and an explanation follows.

```
diff global.inc.php.infinite global.inc.php.doge
48c48
< $config['db']['name'] = 'infinite';
---
> $config['db']['name'] = 'mpos';
56,57c56,57
< $config['wallet']['host'] = 'localhost:9322';
< $config['wallet']['username'] = 'infinitecoinrpc';
---
> $config['wallet']['host'] = 'localhost:22555';
> $config['wallet']['username'] = 'dogecoinrpc';
65c65
< $config['gettingstarted']['coinname'] = 'infinitecoin';
---
> $config['gettingstarted']['coinname'] = 'dogecoin';
68c68
< $config['gettingstarted']['stratumport'] = '5555';
```

```
---
> $config['gettingstarted']['stratumport'] = '4444';
114c114
< $config['currency'] = 'IFC';
---
```

Line 48 is the database. This is where we change from doge to infinite. Lines 56 and 57 are changed in accordance to your coin listen port, and rpc username and password. Line 65 is the coinname which moves from dogecoin to infinitecoin. Line 68 is the stratum listen port and finally line 114 is the three character, call letter name of the new coin which in this case is IFC. And that's it.

So now we are working with a brand new database, infinite thus we have to sign up again, and go through the few administrative settings following the previous coin. For all coins we have a new database, so this is necessary. Generally speaking, we create the administrators sign in, then sign in as the administrator and make the necessary settings changes. So let us take a look.

Create the websites titles, messge of the day, slogan, etc. above, then adjust the blockchain information as indicated below.

Next we adjust the email address in the Systems tab and save the configuration, and it's pretty much a wrap.

Implement the cron tasks once on the server. There is a separate chapter for implementing these cron tasks, however simple, they are a crucial, and very necessary component of the mining web site as they update our block information, and disseminate coinage, as well as maintain the integrity of the system through regular maintenance processes. Thus as they say, the cron tasks are one of the most often forgotten vital components.

So now it is a matter of creating your workers, then mining with these workers using the stratum+tcp://ipaddress:listenport of the stratum. This simply works. If it fails, then it generally a minor adjustment in either the coin daemons configuration file, for example, a missing rpcallow parameter, or a *user error, lol*. There are very many resources via Google ready and waiting for answers to questions relating to user error. Don't take that the wrong way though, user error is a good tool for learning, and is all part of the normal cycle. So have fun and no worries.

The easiest way to test the “system” end to end is by using a cpuminer. Look for the (yay!!!). If you receive a (boo!!!) this always indicates a problem, and is likely the wrong ALGO. For example, your stratum is sha256d and your mining SCRYPT, or vs.

Look for your results as:

```
~/knary/exec/cpuminer$ ./minerd -a scrypt -o stratum+tcp://192.168.1.6:5555 -O
sjmariogolf55.d1:peeb
[2014-04-14 13:35:43] 2 miner threads started, using 'scrypt' algorithm.
[2014-04-14 13:35:43] Binding thread 0 to cpu 0
```

```
[2014-04-14 13:35:43] Starting Stratum on stratum+tcp://192.168.1.6:5555
[2014-04-14 13:35:43] Binding thread 1 to cpu 1
[2014-04-14 13:35:43] Stratum detected new block
[2014-04-14 13:35:45] thread 1: 4096 hashes, 3.08 khash/s
[2014-04-14 13:35:45] thread 0: 4096 hashes, 3.00 khash/s
[2014-04-14 13:35:51] thread 0: 17648 hashes, 2.90 khash/s
[2014-04-14 13:35:59] accepted: 1/1 (100.00%), 5.98 khash/s (yay!!!)
```

This concludes the chapter, however, below for you reference and consideration is a working stratum config.py.

A working config.py for infinitecoin.

```
'''
This is example configuration for Stratum server.
Please rename it to config.py and fill correct values.
This is already setup with sane values for solomining.
You NEED to set the parameters in BASIC SETTINGS
'''
CENTRAL_WALLET = 'iHueKW12mybabydonth@venosh0esjQSAfFm'    # local coin address where money goes
COINDAEMON_TRUSTED_HOST = 'localhost'
COINDAEMON_TRUSTED_PORT = 9322
COINDAEMON_TRUSTED_USER = 'infinitecoinrpc'
COINDAEMON_TRUSTED_PASSWORD = 'edcp00f3d60-b7d1-408c-8200-7e3sp00f3dad5'
COINDAEMON_ALGO = 'scrypt'
COINDAEMON_Reward = 'POS'
COINDAEMON_TX = 'no'
STRATUM_MINING_PROCESS_NAME= 'infinite-stratum-mining'
DEBUG = False
LOGDIR = 'log/'
LOGFILE = None      # eg. 'stratum.log'
LOGLEVEL = 'DEBUG'
LOG_ROTATION = True
LOG_SIZE = 10485760 # Rotate every 10M
LOG_RETENTION = 10  # Keep 10 Logs
THREAD_POOL_SIZE = 300
HOSTNAME = 'localhost'
ENABLE_EXAMPLE_SERVICE = False
LISTEN_SOCKET_TRANSPORT = 5555
LISTEN_HTTP_TRANSPORT = None
```

```
LISTEN_HTTPS_TRANSPORT = None
LISTEN_WS_TRANSPORT = None
LISTEN_WSS_TRANSPORT = None
PASSWORD_SALT = '1fa84e38-4230-4b06-91f6-0f6937a782f6'
DATABASE_DRIVER = 'mysql'        # Options: none, sqlite, postgresql or mysql
DATABASE_EXTEND = False          # SQLite and PGSQL Only!
DB_SQLITE_FILE = 'pooldb.sqlite'
DB_PGSQL_HOST = 'localhost'
DB_PGSQL_DBNAME = 'pooldb'
DB_PGSQL_USER = 'pooldb'
DB_PGSQL_PASS = '**empty**'
DB_PGSQL_SCHEMA = 'public'
DB_MYSQL_HOST = 'localhost'
DB_MYSQL_DBNAME = 'infinite'
DB_MYSQL_USER = 'root'
DB_MYSQL_PASS = 'yourpassword'
DB_MYSQL_PORT = 3306            # Default port for MySQL
DB_LOADER_CHECKTIME = 15        # How often we check to see if we should run the loader
DB_LOADER_REC_MIN = 10          # Min Records before the bulk loader fires
DB_LOADER_REC_MAX = 50          # Max Records the bulk loader will commit at a time
DB_LOADER_FORCE_TIME = 300      # How often the cache should be flushed into the DB regardless of size.
DB_STATS_AVG_TIME = 300         # When using the DATABASE_EXTEND option, average speed over X sec
                                # Note: this is also how often it updates
DB_USERCACHE_TIME = 600         # How long the usercache is good for before we refresh
USERS_AUTOADD = False           # Automatically add users to db when they connect.
                                # This basically disables User Auth for the pool.
USERS_CHECK_PASSWORD = False    # Check the workers password? (Many pools don't)
COINBASE_EXTRAS = '/stratumPool/'           # Extra Descriptive String to incorporate in solved blocks
ALLOW_NONLOCAL_WALLET = False               # Allow valid, but NON-Local wallet's
PREVHASH_REFRESH_INTERVAL = 5   # How often to check for new Blocks
                                #   If using the blocknotify script (recommended) set = to
MERKLE_REFRESH_INTERVAL
                                #   (No reason to poll if we're getting pushed notifications)
MERKLE_REFRESH_INTERVAL = 60    # How often check memorypool
                                #   This effectively resets the template and incorporates new transactions.
                                #   This should be "slow"
INSTANCE_ID = 31                    # Used for extranonce and needs to be 0-31
VDIFF_X2_TYPE = True  # powers of 2 e.g. 2,4,8,16,32,64,128,256,512,1024
VDIFF_FLOAT = False    # Use float difficulty
```

```
POOL_TARGET = 16                    # Pool-wide difficulty target int >= 1
VARIABLE_DIFF = True                # Master variable difficulty enable
USE_COINDAEMON_DIFF = False   # Set the maximum difficulty to the coin difficulty.
DIFF_UPDATE_FREQUENCY = 28800 # Update the coin difficulty once a day for the VARDIFF maximum
VDIFF_MIN_TARGET = 16               #  Minimum Target difficulty
VDIFF_MAX_TARGET = 1024             # Maximum Target difficulty
VDIFF_TARGET_TIME = 15              # Target time per share (i.e. try to get 1 share per this many seconds)
VDIFF_RETARGET_TIME = 120           # Check to see if we should retarget this often
VDIFF_VARIANCE_PERCENT = 30         # Allow average time to very this % from target without retarget
ALLOW_EXTERNAL_DIFFICULTY = False
SOLUTION_BLOCK_HASH = True # If enabled, enter the block hash. If false enter the scrypt/sha hash into the
shares table
BLOCK_CHECK_SCRYPT_HASH = False
ENABLE_WORKER_BANNING = True        # enable/disable temporary worker banning
WORKER_CACHE_TIME = 600             # How long the worker stats cache is good before we check and refresh
WORKER_BAN_TIME = 300               # How long we temporarily ban worker
INVALID_SHARES_PERCENT = 50         # Allow average invalid shares vary this % before we ban
NOTIFY_EMAIL_TO = ''                # Where to send Start/Found block notifications
NOTIFY_EMAIL_TO_DEADMINER = ''      # Where to send dead miner notifications
NOTIFY_EMAIL_FROM = 'root@localhost'  # Sender address
NOTIFY_EMAIL_SERVER = 'localhost'     # E-Mail Sender
NOTIFY_EMAIL_USERNAME = ''            # E-Mail server SMTP Logon
NOTIFY_EMAIL_PASSWORD = ''
NOTIFY_EMAIL_USETLS = True
MEMCACHE_HOST = "localhost"           # hostname or IP that runs memcached
MEMCACHE_PORT = 11211                 # Port
MEMCACHE_TIMEOUT = 900                # Key timeout
MEMCACHE_PREFIX = "infinitestratum_"  # Prefix for keys
```

Chapter 14 going international with the eMark

Enter European currency

DEM? Why not? Currently the European global currency is about 1.3 times the USD. Why not play with it? The exact same process is followed in the previous chapter verbatim substituting emark for infinitecoin. We must download the source daemon and compile it. We then start the daemon, and configure our $HOME/.eMark/eMark.conf daemon configuration file. We create the emark database, and import the MPOS sql

into our new database. Etc.

```
cd $HOME/mpos
git clone https://github.com/emarkproject/DEM.git
#-- compile it, then copy it to /usr/bin, then start it up
```

For your reference, below is an example eMark conf file. The rpcport default, I think is 6666, um..., nah, I think I'll go with 8925, as I like to reserve whole numbers for stratum listens. However, this is up to you.

```
gen=0
server=1
rpcuser=eMarkrpc
rpcpassword=edsp00f3d-b7d1-408c-8200-sp00f3d818ad5
rpcallowip=192.168.1.*
rpcconnect=127.0.0.1
rpcport=8925
addnode=85.84.67.125:5556
addnode=79.245.176.193:5556
addnode=192.241.136.114:5556
addnode=87.205.203.132:5556
addnode=92.43.97.9:5556
addnode=135.23.75.58:5556
addnode=108.30.68.204:5556
```

Now, eMark is a sha256d pow+pos currency, thus the stratum to copy or to download is pretty much the same, normal stratum we have been using, except for the COINDAEMON_ALGO = 'sha256d' . Of course the same rules apply here as are discussed in the previous chapter. We must use eMarkd getaccountaddress “”, and apply this wallet to our config.py. We must change the database name, and the listen port. All that applies in detail in the previous chapter can now quickly be applied to this coin, or for that matter any coin.

Testing can be accomplished in much the same manner as scrypt, using cpuminer, however, -a sha256d is passed as the algorithm as apposed to scrypt.

For additional content, consider that you have a back end mining device like the Block Erupter cube that requires a Getwork proxy between it and the stratum. This task can easily be accomplished by installing a stratum-mining-proxy.py in between the back end device and your stratum listen.

The slush0 stratum mining proxy is extremely easy to install as an intermediary software gateway, or proxy for those requiring the Getwork protocol, such as the BE Cube.

https://github.com/slush0/stratum-mining-proxy

Example slush0 startup:

```
#-- This will start up an 8332 getwork listen and transport

./mining_proxy.py -q -o 192.168.1.4 --port 7777 --custom-user sjmariogolf55.d2
--custom-password peeb
```

The same MPOS configuration items are applicable as was discussed in detail within the last chapter. We copy the global.inc.php into a saved file as infinitecoin, then make the same changes as described, now for eMark. Complete the new registration, minor configuration changes, workers, etc.

The mine DEM.

```
~/knary/exec/cpuminer$ ./minerd -a sha256d -o stratum+tcp://192.168.1.4:7777 -O
sjmariogolf55.d1:peeb

[2014-04-14 14:31:06] 2 miner threads started, using 'sha256d' algorithm.

[2014-04-14 14:31:06] Binding thread 1 to cpu 1

[2014-04-14 14:31:06] Starting Stratum on stratum+tcp://192.168.1.4:7777

[2014-04-14 14:31:06] Binding thread 0 to cpu 0

[2014-04-14 14:31:06] Stratum detected new block

[2014-04-14 14:31:08] thread 1: 2097152 hashes, 2664 khash/s
```

This concludes this chapter, however below is a config.py that can be used with the eMark coin.

The eMark config.py.

```
'''
This is example configuration for Stratum server.
Please rename it to config.py and fill correct values.
This is already setup with sane values for solomining.
You NEED to set the parameters in BASIC SETTINGS
'''
CENTRAL_WALLET = 'Nc@tchm31func@nc@use1mf@stXcYUmmy8B'    # local coin address where money goes
COINDAEMON_TRUSTED_HOST = 'localhost'
COINDAEMON_TRUSTED_PORT = 8925
COINDAEMON_TRUSTED_USER = 'eMarkrpc'
COINDAEMON_TRUSTED_PASSWORD = 'esp00f3d0-b7d1-408c-8200-sp00f3d18ad5'
COINDAEMON_ALGO = 'sha256d'
COINDAEMON_Reward = 'POS'
COINDAEMON_TX = 'yes'
STRATUM_MINING_PROCESS_NAME= 'emark-stratum-mining'
DEBUG = False
LOGDIR = 'log/'
LOGFILE = None      # eg. 'stratum.log'
LOGLEVEL = 'DEBUG'
```

```
LOG_ROTATION = True
LOG_SIZE = 10485760 # Rotate every 10M
LOG_RETENTION = 10  # Keep 10 Logs
THREAD_POOL_SIZE = 300
HOSTNAME = 'localhost'
ENABLE_EXAMPLE_SERVICE = False
LISTEN_SOCKET_TRANSPORT = 7777
LISTEN_HTTP_TRANSPORT = None
LISTEN_HTTPS_TRANSPORT = None
LISTEN_WS_TRANSPORT = None
LISTEN_WSS_TRANSPORT = None
PASSWORD_SALT = '77088357-ce0e-4e7b-8574-2bbfdbab4ca'
DATABASE_DRIVER = 'mysql'        # Options: none, sqlite, postgresql or mysql
DATABASE_EXTEND = False          # SQLite and PGSQL Only!
DB_SQLITE_FILE = 'pooldb.sqlite'
DB_PGSQL_HOST = 'localhost'
DB_PGSQL_DBNAME = 'pooldb'
DB_PGSQL_USER = 'pooldb'
DB_PGSQL_PASS = '**empty**'
DB_PGSQL_SCHEMA = 'public'
DB_MYSQL_HOST = 'localhost'
DB_MYSQL_DBNAME = 'emark'
DB_MYSQL_USER = 'root'
DB_MYSQL_PASS = 'yourpassword'
DB_MYSQL_PORT = 3306            # Default port for MySQL
DB_LOADER_CHECKTIME = 15        # How often we check to see if we should run the loader
DB_LOADER_REC_MIN = 10          # Min Records before the bulk loader fires
DB_LOADER_REC_MAX = 50          # Max Records the bulk loader will commit at a time
DB_LOADER_FORCE_TIME = 300      # How often the cache should be flushed into the DB regardless of size.
DB_STATS_AVG_TIME = 300         # When using the DATABASE_EXTEND option, average speed over X sec
                                # Note: this is also how often it updates
DB_USERCACHE_TIME = 600         # How long the usercache is good for before we refresh
USERS_AUTOADD = False           # Automatically add users to db when they connect.
                                # This basically disables User Auth for the pool.
USERS_CHECK_PASSWORD = False    # Check the workers password? (Many pools don't)
COINBASE_EXTRAS = '/stratumPool/'    # Extra Descriptive String to incorporate in solved blocks
ALLOW_NONLOCAL_WALLET = False        # Allow valid, but NON-Local wallet's
PREVHASH_REFRESH_INTERVAL = 30       # How often to check for new Blocks
                                     #   If using the blocknotify script (recommended) set = to
```

```
MERKLE_REFRESH_INTERVAL
                                    #   (No reason to poll if we're getting pushed notifications)
 MERKLE_REFRESH_INTERVAL = 60         # How often check memorypool
                                    #   This effectively resets the template and incorporates new
transactions.
                                    #   This should be "slow"
 INSTANCE_ID = 31                    # Used for extra-nonce and needs to be 0-31
 VDIFF_X2_TYPE = True  # powers of 2 e.g. 2,4,8,16,32,64,128,256,512,1024
 VDIFF_FLOAT = False    # Use float difficulty
 POOL_TARGET = 20                    # Pool-wide difficulty target int >= 1
 VARIABLE_DIFF = True                # Master variable difficulty enable
 USE_COINDAEMON_DIFF = True          # Set the maximum difficulty to the coin difficulty.
 DIFF_UPDATE_FREQUENCY = 3600        # Update the coin difficulty once a day for the VARDIFF maximum
 VDIFF_MIN_TARGET = 2                #  Minimum Target difficulty
 VDIFF_MAX_TARGET = 2048             # Maximum Target difficulty
 VDIFF_TARGET_TIME = 15              # Target time per share (i.e. try to get 1 share per this many seconds)
 VDIFF_RETARGET_TIME = 60            # Check to see if we should retarget this often
 VDIFF_VARIANCE_PERCENT = 10         # Allow average time to very this % from target without retarget
 ALLOW_EXTERNAL_DIFFICULTY = True
 SOLUTION_BLOCK_HASH = True          # If enabled, enter the block hash. If false enter the scrypt/sha hash into
the shares table
 BLOCK_CHECK_SCRYPT_HASH = False
 ENABLE_WORKER_BANNING = True        # enable/disable temporary worker banning
 WORKER_CACHE_TIME = 600             # How long the worker stats cache is good before we check and refresh
 WORKER_BAN_TIME = 300               # How long we temporarily ban worker
 INVALID_SHARES_PERCENT = 50         # Allow average invalid shares vary this % before we ban
 NOTIFY_EMAIL_TO = ''                # Where to send Start/Found block notifications
 NOTIFY_EMAIL_TO_DEADMINER = ''      # Where to send dead miner notifications
 NOTIFY_EMAIL_FROM = 'root@localhost'  # Sender address
 NOTIFY_EMAIL_SERVER = 'localhost'   # E-Mail Sender
 NOTIFY_EMAIL_USERNAME = ''          # E-Mail server SMTP Logon
 NOTIFY_EMAIL_PASSWORD = ''
 NOTIFY_EMAIL_USETLS = True
 MEMCACHE_HOST = "localhost"         # hostname or IP that runs memcached
 MEMCACHE_PORT = 11211 # Port
 MEMCACHE_TIMEOUT = 900 # Key timeout
 MEMCACHE_PREFIX = "emarkstratum_"   # Prefix for keys
```

- •.Finding new blocks created by the pool
- •.Updating block confirmations in the pool
- •.Counting round boundaries
- •.Processing user round/share payouts and debit transactions (manual and automatic)
- •.Processing worker notifications
- •.Updating statistical data used by the frontend
- •.Cleanup of archived shares
- •.Other smaller tasks that needs to be handled by the backend

Logging

All crons have *logging to console* disabled. Instead, a logfile is created for each cron under the cronjobs/logs directory in **MPOS**. If you are having issues with crons (you can see their status on the Admin Panel -> Monitoring page) this is the best place to check for additional information. If you wish to enabled debug mode, modify shared.inc.php and replace KLogger::INFO with KLogger::DEBUG. Ensure to disable this option again when you are done, logfile will grow rather large with it enabled!

Log Rotation

A logrotate example file is included in the repository at cronjobs/etc/logrotate.conf and can be triggered with the cronjobs/logrotate.sh script. It will cycle the files at a daily interval keeping up to 7 days as an archive.

Cronjobs

A brief description of all the *cronjobs* included in this project, sorted alphabetically.

archive_cleanup.php

Scans the shares_archive table for old shares that can be deleted according to the configuration. You can modify the cleanup behavior in the global config.

payouts.php

Runs all scheduled payouts setup or initiated by users. Each user will be processed independently so even if a user has setup a wrong LTC address (even though **MPOS** will try to ensure it's valid), other users are not affected by that. Disabling this payout via Admin Panel will allow for some maintenance work without payouts being processed.

blockupdate.php

Checks our RPC service for updated confirmation records for our found blocks and transfers them to the DB. Without this, no transactions would be confirmed and converted to confirmed credits.

findblock.php

As the name suggest, it will look into the RPC service for newly found blocks and add them to the database. Once all blocks are found, it will check each blocks upstream accepted share and calculate the round shares. No further processing is done, but this information will be used by the payout crons later.

notifications.php

As the name suggestes, notifications related to backend jobs like IDLE workers will be sent with this cronjob. It

will also ensure that notifications are reset once a worker becomes active again so new notifications can be send out. You can globally disable notifications but each user also has the ability to setup their notifications. Globally disabled notifications override all user setting.

pplns_payout.php & proportional_payout.php & pps_payout.php

These crons, depending on which one is enabled in the global configuration, will handle payouts to users. PPLNS and Prop both are round based while PPS will payout per share. Round ends are still processed for statistical purposes. Only unaccounted new blocks with a valid share ID will be processed!

statistics.php

Updates various cache keys in memcache to allow the frontend to respond fast to requests. Most notably it is checking for all users shares and increments them each run so this information is available to the frontend without checking the database.

tickerupdate.php

Runs API calls against external sites. Since we can't cross-call APIs due to XSS protection in browsers, this cron will check them instead. Results are added to the database and made available to the frontend.

tables_cleanup.php

Runs a few maintenance tasks to keep your tables clear.

Setup

Instead of running a single cronjob manually it is recommended to use a split automatic setup to ensure some crons are always running at set intervals, specifically our statistics.php. For long rounds or coins with a lot of shares, it is a requirement to run the statistics cron as often as possible to ensure a responsive site after a round ends.

run-statistics.sh

This cron will ONLY update the statistics cache. You can add it to your crontab at a minutely interval.

run-payout.sh

Manages the round ends, finding blocks and everything related to internal and external payouts. Can run as often as you need, for PPS pools it is recommend once per 30 minutes to one hour. This keeps the amount of transactions created as low as possible, enabling faster payout for users.

run-maintenance.sh

Clears up your tables, sends IDLE worker notifications and updates the coin price and uptime robot status. Can be run as often as needed, but once every minute should be fine.

Sample configuration

This is a sample configuration for a pool. It will disable mailing events through cron and ensure that we are not running into cron-conflicts between multiple MPOS instances by adding the subfolder parameter -d. Other coins can be called the same way by adding other subfolders where to store the PID files (usually /tmp/<FOLDER>/*.pid). To make your own MPOS configuration run your crontab by calling crontab -e and pasting the cronjob data you create

here at the end of the created file. Once your file is created and saved, you can view your crontab by calling it with the crontab -l command.

Example crontab:

```
# Edit this file to introduce tasks to be run by cron.
MAILTO=""
0,5,10,15,20,25,30,35,40,45,50,55 * * * * /var/www-ssl/MPOS/cronjobs/run-statistics.sh -d DGC
0,5,10,15,20,25,30,35,40,45,50,55 * * * * /var/www-ssl/MPOS/cronjobs/run-payout.sh -d DGC
0,5,10,15,20,25,30,35,40,45,50,55  *  *  *  *  /var/www-ssl/MPOS/cronjobs/run-maintenance.sh -d DGC
```

Chapter 16 it's all about the Defense

Enter the MPOS Application Suite

Distributed denial-of-service attacks have posed an increasingly severe problem for cryptocurrency exchanges and mining pools in recent weeks. Last month, several major pools in the mining community suffered debilitating DDoS attacks that resulted in significant delays, lost mining time and frustration for miners.

In extreme cases, as explained by TeamDoge administrator Forrest Fuqua, some pools received ransom messages from hackers demanding payoffs in exchange for pulling back their attacks.Fuqua said that security flaws in the Mining Portal Open Source (MPOS) pool software commonly used throughout the community have made it all too easy for cyber attackers to disrupt mining activities and extract ransoms from pools.

"Dogepool.pw actually got their database hacked at one point, due to the fact of insecurities in the main pool software that everybody owns. Even the biggest mining pool for Litecoin uses it as their backend. It's everywhere in their templates – they're using the exact same framework. So some of these security exploits affect us all."

First and foremost we must protect our site just as the walls protected us in past generations, thus it is true today, metaphorically speaking.

A minimal, and pretty good DoS protection can be found here.
https://github.com/MPOS/php-mpos/wiki/Basic-DoS-Protection.

With excerpts from the MPOS wiki. Security is an Onion. There is more to pool security than a 1 page bullet-point list and there is no magical program to keep you from getting hacked, this is just a primer.

Pre-Installation

- •.Get onto your production box, setup ssh keys.
- •.Update and install all the dependencies, mail etc.
- •.Complile ypur own PHP.
- •.Compile your own Apache.
- •.Make sure apache/php/mysql/$mailserver are playing nice together.
- •.Run phpsecinfo.
- •.Make sure display_errors is Off in your php.ini.
- •.Make sure your session.save_path is NOT web accessible in your php.ini.
- •.And if you're not running it yet and reading along, run phpsecinfo.

Apache / MySQL / PHP

- •.Lock down the Apache document root to only MPOS.
- •.If you have an SSL cert, make sure you have installed it correctly. Use a valid cert.
- •.Enable [cookies][secure] in global config and [strict__https_only] in security config .
- •.Your MySQL user should not be root, setup a new user with permissions you set.
- •.Log everything you can.

MPOS

- •.Turning on [twofactor] will protect your users from themselves.
- •.Get an SSL cert and take the extra 10 minutes, it's worth it.
- •.Make sure your [cookie] settings are correct.
- •.Memcache should be enabled unless you absolutely cannot use it (I don't believe you).
- •.Strict mode will stop a few types of attacks, so use it.
- •.If you're paranoid use strict__verify_server and set the strict__bind_'s to your server info.
- •.Remove unnecessary software; Your production box doesn't need phpmyadmin.
- •.Download and run phpsecinfo.
- •.Clear your bash history.
- •.Clear your MySQL history.

Some additional fixes we may consider include:

- •If a IP locks more than 2 accounts, it gets banned.
- •Using geoip database to block suspicious IPs from countrys like the Philippines and so on, probably does not help much because of TOR.
- •Added Re-Captcha's to sign-ups and logins.
- •IP Banning in Stratum much faster than the defaults, I'm banning already after 5 seconds of sending "nonsense" - which works quite well.
- •DDoS protection such as Blacklotus and Cloudflare.
- •Virtual F5 BigIP on a separate front end. Funnel all traffic through this F5 LB, taking advantage of their built in DDOS, FW, encryption/decryption capabilities.

- •Google Authenticator for all logins/payouts/adress changes/everything.

Chapter 17 summary and closing remarks

Enter the end

There is a lot of information within the chapters of this book, however, I must say that there is countless more words that need to be expressed as this is just an introduction, really to the amount of available content surrounding this singular topic. It is quite overwhelming, at best, and rather daunting at times, but I assure you there are people all around the world doing this successfully and profiting. If there is a singular topic that promotes itself above all others, it has to be security . We live in an ever changing, and volatile world. To quote your mom by saying “it's dangerous out there,” is an underestimate of just how dangerous it really is. There really are bad people that would love to steal everything from you, so I put security top amongst the most important topics covered.

As was quoted, “security is like an onion,” and well, it is. Unfortunately while this book provides the basic, and somewhat more than basic security tips, be it known that security is a great field to be in because it is always changing. Please consider security when designing anything, especially if it is a web based mining site.

If I were to convey to you all of the information you will require to be a successful Internet entrepreneur, well in the first place, I simply could not. Secondly, it would take a lifetime to both write and to read. What I can provide, I think I have, in a succinct, and repeatable manner.

Good luck and God speed. Until we speak again.

Thanks

Steven

Appendix A Sometimes the Proof is in the pudding

Statistics of one 24 hour run at or below 100 KH/s.

Solo mining with the getwork protocol just quite simply does not seem to do the job. One would literally spend months mining to the getwork port of a coin daemon and never see even one accepted share, let alone to ever have hopes of solving a block. For the sake of this book I ran one test of a simple mining server guaged at 100 KH/s. Yes I know this is small potatoes, however as a baseline it will do nicely. After a single day run here are the actual statistics for your perusal. No I did not solve a block but I did manage to find run over 800 blocks and greater than 200,000 shares of which over 9,000 were mined. This is encouragement at its very finest. A single Litecoin block solution is worth 50 coins.

```
[2014-03-31 15:02:23] Started at [2014-03-30 00:01:09]
```

```
[2014-03-31 15:02:23] Pool: stratum+tcp://192.168.1.16:3333
[2014-03-31 15:02:23] Runtime: 24 hrs : 1 mins : 13 secs
[2014-03-31 15:02:23] Average hashrate: 98.3 Kilohash/s
[2014-03-31 15:02:23] Solved blocks: 0
[2014-03-31 15:02:23] Best share difficulty: 981K
[2014-03-31 15:02:23] Share submissions: 9167
[2014-03-31 15:02:23] Accepted shares: 9127
[2014-03-31 15:02:23] Rejected shares: 40
[2014-03-31 15:02:23] Accepted difficulty shares: 209222
[2014-03-31 15:02:23] Rejected difficulty shares: 953
[2014-03-31 15:02:23] Reject ratio: 0.4%
[2014-03-31 15:02:23] Hardware errors: 0
[2014-03-31 15:02:23] Utility (accepted shares / min): 3.90/min
[2014-03-31 15:02:23] Work Utility (diff1 shares solved / min): 88.69/min

[2014-03-31 15:02:23] Stale submissions discarded due to new blocks: 0

[2014-03-31 15:02:23] Unable to get work from server occasions: 4
[2014-03-31 15:02:23] Work items generated locally: 11354
[2014-03-31 15:02:23] Submitting work remotely delay occasions: 0
[2014-03-31 15:02:23] New blocks detected on network: 886
[2014-03-31 15:02:23] Summary of per device statistics:
[2014-03-31 15:02:23] GPU0                | (5s):97.42K (avg):98.30Kh/s | A:209222 R:953
HW:0 WU:88.7/m
[2014-03-31 15:02:23]
```

Statistics of one 24 hour run infinitecoin at or below 500 KH/s.

This next example is the infinitecoin. I was pleasantly surprised to find out that the infinitecoin block solution pays 1,200 coins per block found. This is a very easy coin to mine, yet has about the same potential return as the Dogecoin. Below is a chart from MPOS showing expected normal PPLNS, (say from a normal Internet mining site,) and the actual return. The actual return is staggeringly higher than the expected.

These are actual return statistics from a little over 24 hours of mining below 500 KH/s.

```
 infinitecoind getinfo
{
    "version" : 1080700,
    "protocolversion" : 69002,
    "walletversion" : 60000,
    "balance" : 9216.00000000,
    "blocks" : 834169,
    "connections" : 12,
    "proxy" : "",
    "difficulty" : 0.29057961,
    "testnet" : false,
    "keypoololdest" : 1397084575,
    "keypoolsize" : 101,
    "paytxfee" : 10.00000000,
    "mininput" : 0.01000000,
    "errors" : ""
}
```

Statistics of one 24+ hour run sha256d eMark at or below 30 MH/s.

After 12 hours of mining the DEM eMark international coin, I awoke to another pleasant surprise. A block found. The eMark is a POS + POS and confirmed 50 coins directly into my wallet. I'd rather it pay 1,200 like infinitecoin, but let's be real, and I'll take it considering this coin is based on the Eur o, currently trading higher that the USD. It's not yet Bitcoin, but maybe it will be somday, who knows. This setup uses ½ of a BE cube and a stratum-mining-proxy supplying the Getwork 8332 port pointing to the Stratum servicing MPOS.