McAfee® Command Live Scanner for Windows
Live Linux Boot CD Project

User's Guide

Version 1.0.0
February 5, 2010
Michael G. Spohn
mspohn@malware-hunters.net

Table of Contents

Table of Figures

Table of Figures

Introduction

This document is the detailed user's guide for the McAfee Command Line Scanner Project (MCLSP). It is designed to assist non Linux power-users in building a custom Ubuntu Linux distribution that runs the McAfee Command Line Scanner.

For more experienced Linux administrators there is a condensed version of this document named *Quick-Start Guide.* Also, be sure to read the *FAQ* document which provides good background information about this project.

The MCLSP provides a unique method of scanning and cleaning Windows hosts of malware, using a Live Ubuntu Linux CD/DVD or USB thumb drive. It was created for use in emergency incident response situations where series malware compromises require alternate A/V scanning tactics.

This magic is accomplished through the use of the Linux *Wine* package. This amazing open-source project provides a very efficient mechanism to run Windows binaries on Linux systems. In this case, we wish to run the *McAfee A/V Command Line Scanner for Windows* on Linux. Having the ability to do this provides all sorts of options to innovative incident responders and system administrators.

The real advantage of this approach is the fact the host is not running Windows when the A/V scan occurs. This allows unfettered access to all files on a file system and ensures a thorough scan will be performed.

The MCLSP provides you all the tools you need to create a custom Ubuntu Linux distribution (distro) specifically designed to automatically boot and scan Windows host for malware. There are three steps in the process of building a custom distro ISO.

1. *Creating a build platform you will use to create your custom distro ISO*

   The build environment must be a computer running Ubuntu Linux. It is strongly recommended you use Ubuntu Desktop Version 10.10 (Maverick Meerkat) as your build system OS.

2. *Setting the correct configuration options to suit your needs*

   There are three configuration options you must consider. First, you must decide on the desktop background you want use. A pretty cool background image is provided, but you are free to change it. Second, you need to configure the A/V scan environment settings. Finally, you need to configure the McAfee Command Line Scanner options.

3. *Building the custom distro ISO*

   There are three scripts you need to run in order to accomplish this. Once the scripts have done their work, you will have a custom distro ISO configured to boot and scan Windows host for malware. You can burn this ISO to CD/DVD's or USB thumb drives.

The rest of this document walks you through these three steps.

Creating a Build Platform

Building a Host System

In order to build a custom Ubuntu Linux distro you need to find or build a laptop or workstation running Linux. I strongly recommend you use Ubuntu Desktop Version 10.10 (Maverick Meerkat) due to its extensive driver coverage for the latest hardware.

Find an old workstation or laptop and install Ubuntu Desktop 10.10. You can download it from here. You will need to burn the downloaded ISO image to a CD/DVD or USB thumb drive. Instructions for doing this can be found in Deployment section of this document.

If your old hardware is running Windows, you must decide if you want to install Linux alongside Windows (Dual-boot configuration). For new Linux users this can get complex. Again, I suggest you dedicate an old piece of hardware to this project and install a clean version of Ubuntu 10.10.

If you do not have old hardware lying around, an alternate method is to install Ubuntu 10.10 on a bootable external USB hard drive. You install Ubuntu 10.10 on the USB drive and boot any computer with it when you want to build your custom ISO. I wrote very detailed instructions on how to do this. You can find the document here. The only complication is that some old hardware BIOS's cannot boot a system from a USB device.

Decide on your approach and get your system of choice running Ubuntu 10.10. If you are not sure how to achieve this find a buddy who is familiar with Linux to help you get your build system setup.

One final thing; once you have your system running make sure it is connected to the Internet. The Ubuntu install system is very good about identifying network card(s) and configuring them properly. Fire up Firefox and browse to your favorite web site to be sure.

Installing MCLSP Project Files

Now that you have a build platform up and running, the next thing you need to do is install the MCLSP project files. This is a simple and quick process. Open up Firefox on your build system and navigate to the Malware-Hunters.net web site and download the latest MCLSP project archive. By default, Firefox will place the file in your user profile Downloads folder (e.g. /home/mspohn/Downloads). The file is named *mclsp.tar.gz.*

Once you have the MCLSP tarball on your build system, move the file to the /opt folder. You will need to be root to do this. Open up a terminal session and navigate to the folder where you downloaded the tarball. From the command line type the below commands:

```
cd ~/Downloads
sudo mv mclsp.tar.gz . /opt
```

You should now have the *mclsp.tar.gz* in the /opt folder as shown below in Figure 1.

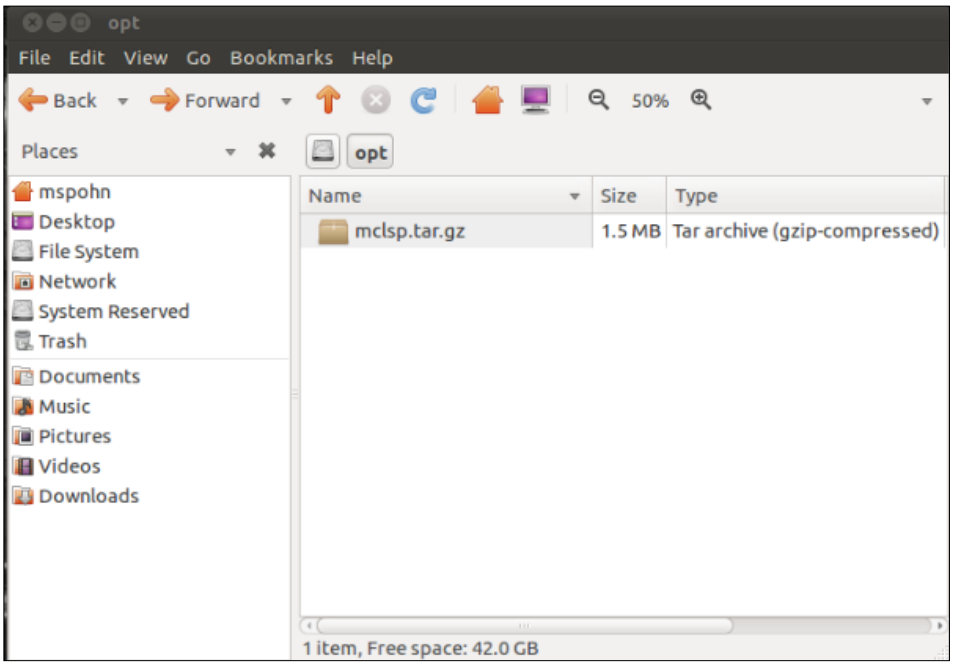

Figure 1: mclsp.tar.gz file in /opt folder

From the command line change directories to /opt and expand the tarball (Figure 2).

cd /opt

sudo tar –zxvf mclsp.tar.gz

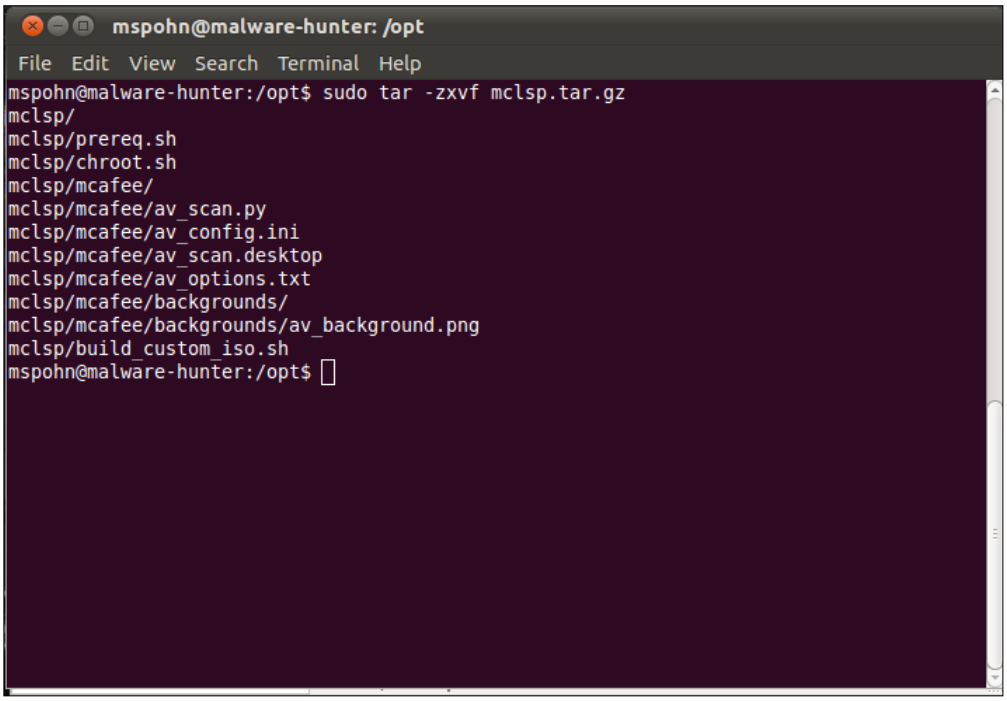

Figure 2: Expand mclsp.tar.gz Tarball

Next, you will need to place a copy of the *Ubuntu-10.10-desktop-i386.iso* you used to install your build system in the */opt/mclsp* folder. Again, you will need to be root to do this. In the below command, the *Ubuntu-10.10-desktop-i386.iso* file is in my user-profile *Downloads* folder.

cd ~/Downloads

sudo cp Ubuntu-10.10-desktop-i386.iso /opt/mclsp

Finally, you need to copy the McAfee Command Line Scanner for Windows product install zip file to */opt/mclsp. You can obtain this file using your McAfee Grant-ID from your McAfee portal. You can also download a demo version of the product from the McAfee web site.*

cd ~/Downloads

sudo cp vscl-w32-6.0.3-e.zip /opt/mclsp

You now have the required files in place to build your custom distro. Figure 3 below shows what the /opt/mclsp folder tree looks like in the file explorer. Figure 4 shows the folder tree from the command line.

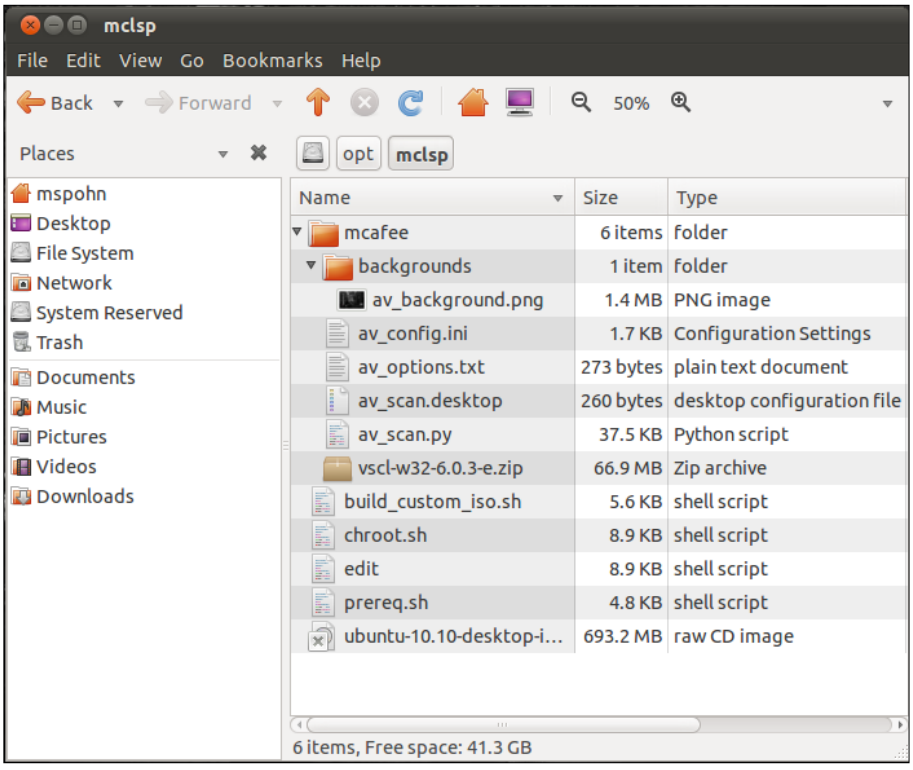

Figure 3: /opt/mclsp Folder Structure (File Explorer)

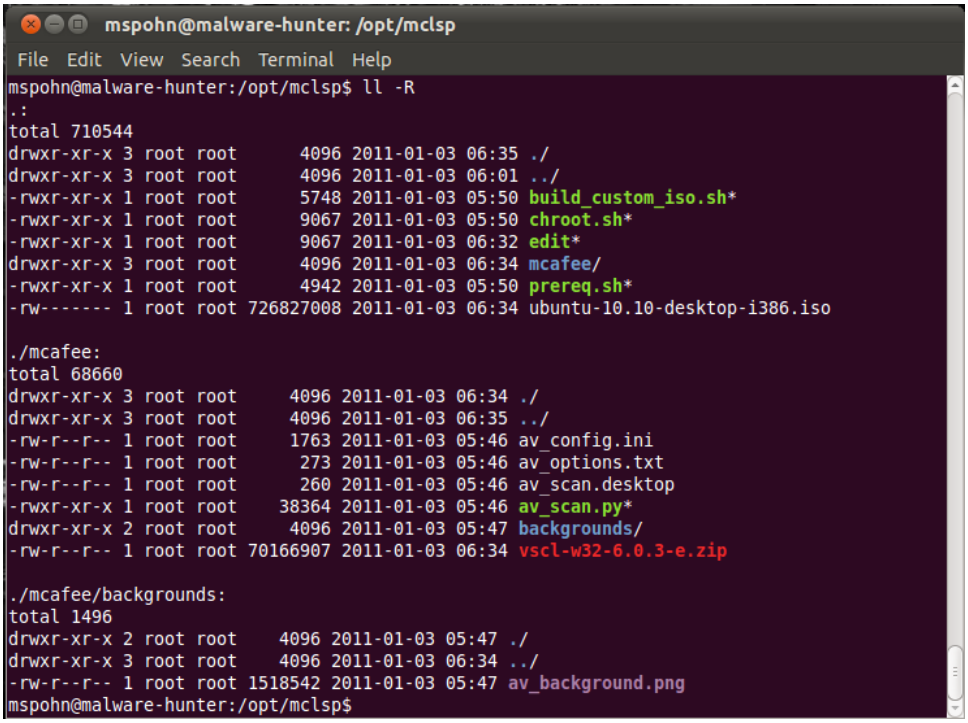

Figure 4: /opt/mclsp Folder Structure (Command Line)

Verifying Package Sources

The required files are now in place. Before we start a build, we need to ensure the correct package sources are enabled. The build scripts download content from the Internet so it is critically important the right repositories are enabled.

To do this, fire up the Synaptic Package Manager applet as shown in Figure 5.

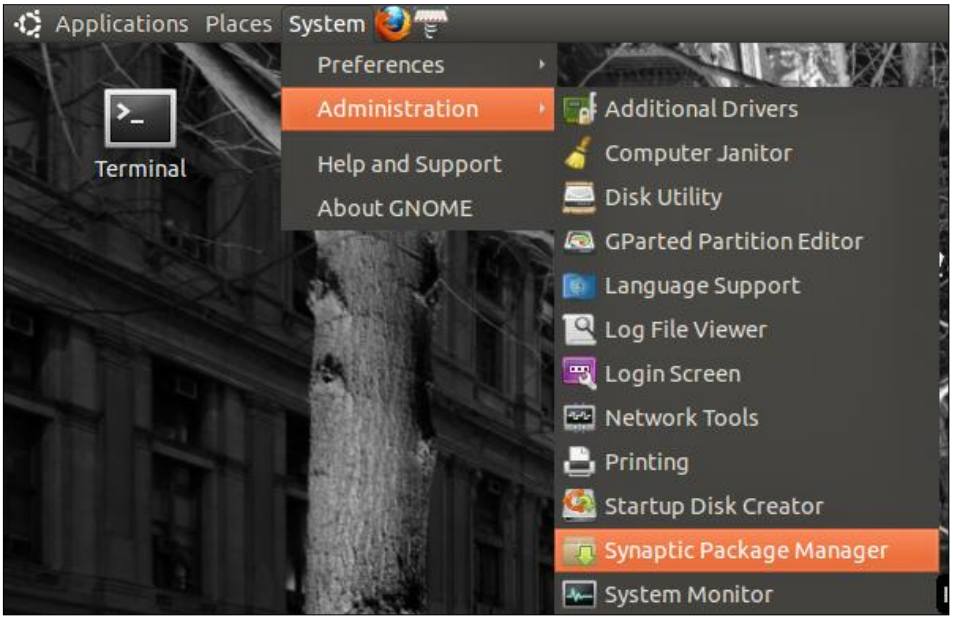

Figure 5: Synaptic Package Manager

You will need to enter in your sudo password because you must be root to work with package manager. Click on *Settings* | *Repositories* as shown in Figure 6.

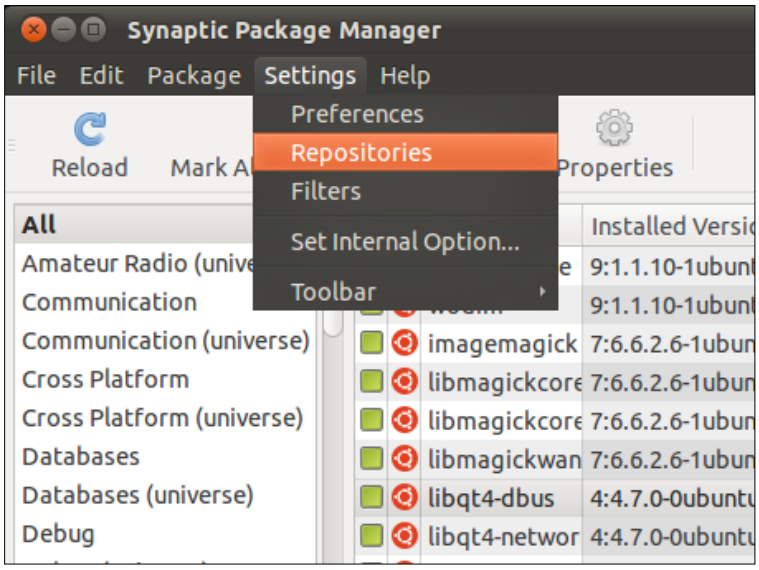

Figure 6: Edit Repositories

Select the *Ubuntu Software* tab and make sure the (main), (universe), and (restricted) repositories are checked as shown in Figure 7.

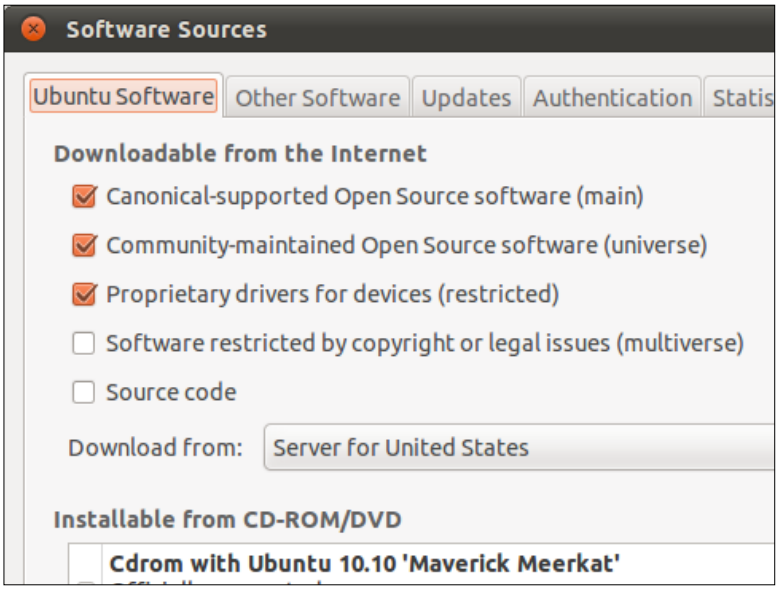

Figure 7: Select Repositories

Before you exit out of the Synaptic Package Manager you will be warned that the repositories need to be updated. Be sure you allow the packages to be updated. You can also do this from the command line as shown in Figure 8.

sudo apt-get update

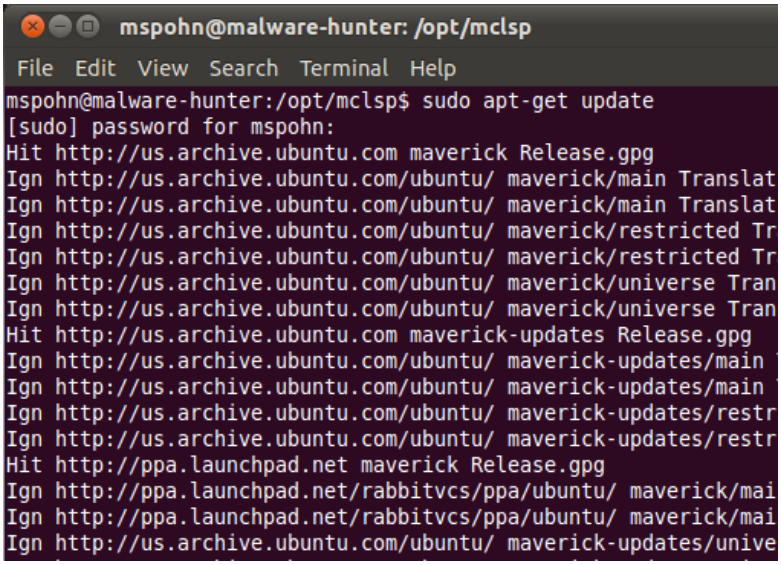

Figure 8: Updating Repositories (Command Line)

Your build platform is now complete. You have everything in place you need to build your custom Ubuntu distro that will run the McAfee A/V Command Line Scanner for Windows.

Configuring Your Custom Ubuntu Distro

Configuration Settings

Before you build your custom ISO, you need to make some configuration decisions. There are three configuration options you must consider. First, you must decide on the desktop background you want use. A pretty cool background image is provided, but you are free to change it. Second, you need to configure the A/V scan environment settings. Finally, you need to configure the McAfee Command Line Scanner options.

1. Configuring the desktop background image.

   The Desktop background image file the build will use can be found in */opt/mcsp/mcafee/backgrounds* and is named *av_background.png*. The build configuration settings place this file in the folder /*home/mcafee/backgrounds/* in your custom distro. To change the Desktop background, replace the file */opt/mcsp/mcafee/backgrounds*/*av_background.png* with your own image file. I did a lot of experimenting and determined the ideal image size is 1600x1200. This sized image renders well on most display devices. Remember, change the file contents but do not change the file name. The Gnome Desktop manager is configured to use a background file named av_background.png.

2. Configuring the environment options.

   You control the behavior of the McAfee A/V scan environment by changing settings in a traditional INI file. The file is named *av_config.ini* and it is located in the build folder */opt/mclsp/mcafee/av_config.ini* file. There are five (5) sections in the INI file; [AUTORUN], [DAT], [SMB], [SCAN], and [MOUNT].

   The [AUTORUN] section contains one variable named '*autoscan*.' If this variable value is set to 1 (Default), then a scan of all FAT and NTFS volumes will be performed whenever your custom distro is booted. If it is set to 0, then no automatic scan will take place.

   The [DAT] section contains the location and name of the McAfee daily DAT file. If the variable named *autodownload* is set to 1 (Default), then the A/V scan script will connect to the McAfee ftp site and download the latest DAT *before* the scans start. This will only work if the workstation booted with your custom distro is connected to the Internet.

   The [SMB] section contains the location of an SMB file share location the scan script will use to download files. This is a convenient way for you to place all of the required A/V scan files on a fileshare. If the variable named *autodownload* is set to 1, then the A/V scan script will connect to this SMB share and download all the files it finds there. (The Default setting for this variable is 0). These files will be placed in the */home/mcafee* folder on the workstation to be scanned. Once the files are downloaded, any zip files will be uncompressed before the A/V scan begins. This will only work if the workstation booted with your custom distro is connected to the Internet.

The user name and password for the SMB fileshare is stored unencrypted in the INI file. This is frowned upon as a security practice of course. If you use this setting, make sure the SMB file share is configured for readonly use and does not contain any sensitive files.

The [SCAN] section contains file path locations and the name of the McAfee A/V options file. The *home_path* variable contains the location of the MCLSP files on the custom distro. The default location is */home/mcafee.* The variable *report_path* contains the location the scan script should use to place the report outputs. The default location is */tmp/mcafee*. The *infected_path* variable contains the location the scan script will quarantine infected file. The default location in /tmp/mcafee/infected.

The [MOUNT section contains a single variable named *mount* .The value of this variable determines how FAT/NTFS file systems are mounted on the host system being scanned. If the value is *ro* (Default), then all filesystems will be mounted read-only. This means the McAfee scan engine will *report* on viruses found, *but will not clean or delete compromised files*. If you want the scan engine to clean or delete infected files, change this setting to (rw). If the value is *rw* then filesystems will be mounted read-write.

NOTE: The default mount configuration (*ro*) will not clean malware from a system, it will only report what the scan engine found. If you want the McAfee A/V scan engine to delete or quarantine (Move) infected files, you *must* do two things: 1) Change the *mount* parameter in the [MOUNT] INI file section to *rw* **and** 2) Add the /CLEAN option to the *av_options.txt* file.

The INI configuration layout is efficient and simple to understand. It is also extensible. If you want to change the behavior of the system, you can add sections and variables to the INI file. If you do so, then you have to make changes to the av_*scan*.py Python script. The INI values are processed in the *ReadConfigFile*() function in the Python script. If you read this function you will quickly determine how it works.

NOTE: One other important thing to be aware of. Your custom Ubuntu A/V scan distro operates as a *Ram-Disk*. This means the operating system runs in system memory. As described above, the default A/V scan report output and malware quarantine location is */tmp/mcafee*. This location is in memory and *not* on disk.

The importance of this should be obvious. *You should not power off a host that has been scanned until you read or collect the scan output reports and quarantined files.* Once the host is powered off, the scan output will disappear.

A sample av_config.ini report is shown in Table 1.

[AUTORUN]
autoscan=1
[DAT]
autodownload=1
dat_url=ftp://ftp.mcafee.com/commonupdater/
dat_name=avvdat-????.zip
[SMB]
autodownload=0
file_share=//hostname/mcafee_dats/
user=mspohn
pwd=password
[SCAN]
home_path=/home/mcafee/
report_path=/tmp/mcafee/
infected_path=/tmp/mcafee/infected/
av_options_file=z:/home/mcafee/av_options.txt
[MOUNT]
mount=ro

Table 1: Typical INI File Layout

3. Configuring the McAfee scan engine options.

   McAfee has provided excellent documentation with its command line scanner product. You can find it in the PDF document named *vcl6wpg.pdf* in the product distribution zip file. It is highly suggested you read this document to understand the power and capabilities of the scan engine.

   You control the behavior of the McAfee A/V scan engine by changing settings in the */opt/mclsp/mcafee/av_options.txt* file. This file contains the scan options that will be used by the command line scan.exe engine. There are a lot of options so it is critically important you understand them. The default options file provided will scan all files on a volume and report them in */tmp/mcafee/av_badlist.txt.* It will take no action against any found malware. You will certainly want to change the options in this file to suit your particular needs.

   Table 2 lists the default settings in the *av_options.txt* file.

Setting	Description
/ALL	Scan all files on the volume
/ANALYZE	Use heuristic analysis
/BADLIST=z:\tmp\mcafee\av_badlist.txt	Create list of malware found
/DAM	Delete infected macro files
/MAILBOX	Scan plain-text mailboxes
/MANALYZE	Scan Microsoft Office macros
/MAXFILESIZE=2000	Max file size to scan (2GB)
/MIME	Scan MIME encoded files
/NOEXPIRE	Disable old DAT warning messages
/PANALYZE	Use heuristic analysis on program files
/PLAD	Preserve last access time on files scanned
/PROGRAM	Scan for 'Potentially Unwanted Programs"
/RECURSIVE	Scan subdirectories
/REPORT=z:\tmp\mcafee\av_scan_rpt.txt	Create a report and place it here
/RPTOBJECTS	Report on objects scanned
/STREAMS	Scan NTFS streams
/THREADS=8	Concurrent thread count
/TIMEOUT=300	Timeout scan after 300 seconds
/UNZIP	Scan zip archives

Table 2: Default McAfee A/V Scan Engine Options

Creating Your Custom ISO

To build your custom live Ubuntu ISO with the McAfee scan engine you need to run three scripts: 1) *prereq.sh*, 2) *chroot.sh*, and 3) *build_custom_iso.sh*. You will find these scripts in the */opt/mclsp* folder. These scripts are heavily commented and describe what each set of commands do. It is highly recommended you review each script to get an idea of how they work. These scripts must be run with root privileges so you ***must*** be logged in as root or use the *sudo* prefix. Also be sure you are connected to the Internet prior to running the scripts.

Script *prereq.sh*

The first script you need to run is named *prereq.sh* and it is located in */opt/mclsp* folder. This script satisfies all of the prerequisite tasks required to customize an Ubuntu Linux distribution. The script performs the following tasks:

- Sets the $BASEDIR environment variable
- Unmounts filesystems used in previous builds
- Removes files and folder used in previous builds
- Creates required folders
- Downloads and installs squashfs-tools package if not already installed
- Downloads and installs genisoimage package if not already installed
- Mounts the *ubuntu-10.10-desktop-i386.iso* filesystem on loop device
- Copies the *ubuntu-10.10-desktop-i386.iso* filesystem files to *extract-cd* folder
- Unsquashes *filesystem.squashfs* and moves it to *edit* folder

- Copies the build system network config files to *edit* folder.
- Mounts the build system /dev folder on *edit/dev*
- Copies the *chroot.sh* script to *edit* folder

There is a lot going on here, so let's make sure it is clear what happens. First the script deletes all files and folders from previous builds so we start fresh. Next, it creates the folders we will need (*mnt*, and *extract-cd*). These folders are relative to the base directory */opt/mclsp*. The *mnt* folder is where we will mount the *ubuntu-10.10-desktop-i386.iso* image. The *extract-cd* folder is where are going to place the *ubuntu-10.10-desktop-i386.iso* image squashfs contents. In other words, we are going to mount the Ubuntu ISO image, and unsquash its filesystem so we can make changes to it.

Next, the script downloads the tools needed to manipulate squashfs files and create ISO files. Once these tools are in place, the script mounts the *ubuntu-10.10-desktop-i386.iso* image on the loopback device so we can access it on the *mnt* folder. Once this is done, we have access to the contents of the ISO image.

Once the ISO image is mounted, the script unsquashes (de-compresses) a file on the image named *filesystem.squashfs*. This file is a compressed image of the entire Ubuntu Linux filesystem. The squashfs tool will place the filesystem in a folder named *squashfs-root*. The script then changes the name of this folder to *edit*. The *edit* folder is where we will make changes to the Ubuntu distro for our custom purposes.

The script next copies the build system network configuration files to the *edit* folder. We need to do this to ensure we can access the Internet when we change the location of the build system root location to the *edit* folder in the next script. The files that are copied include */etc/resolve.conf*, */etc/hosts*, and */etc/apt/sources.list*. If there are other customized network configuration files on your build system needed to access the Internet, make sure these files are copied to the appropriate location in the *edit* folder.

Now the script mounts the build host system */dev* (device folder) on the *edit/dev* folder. This provides us a working filesystem when we *chroot* in the next script. The last thing the script does is copy the script *chroot.sh* to the *edit* folder. This script contains all the command needed to customize our distro.

Now that you know what the prereq.sh script does, let's run it using the below commands from a terminal session:

```
cd /opt/mclsp
sudo ./prereq.sh
```

You will be prompted for your password in order to run the script as root user. The script will begin execution as shown in Figure 9. Monitor the progress of the script to ensure it completes successfully.

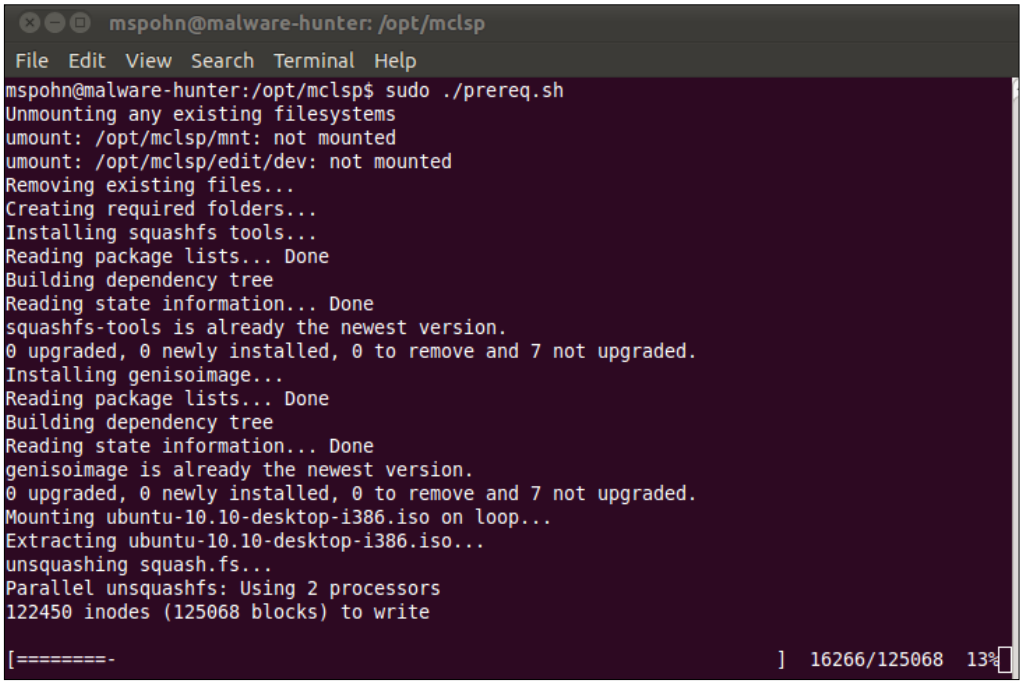

Figure 9: prereq.sh Script Execution

Figure 10 below shows the */opt/mclsp* folder contents after the prereq.sh script execution completes.

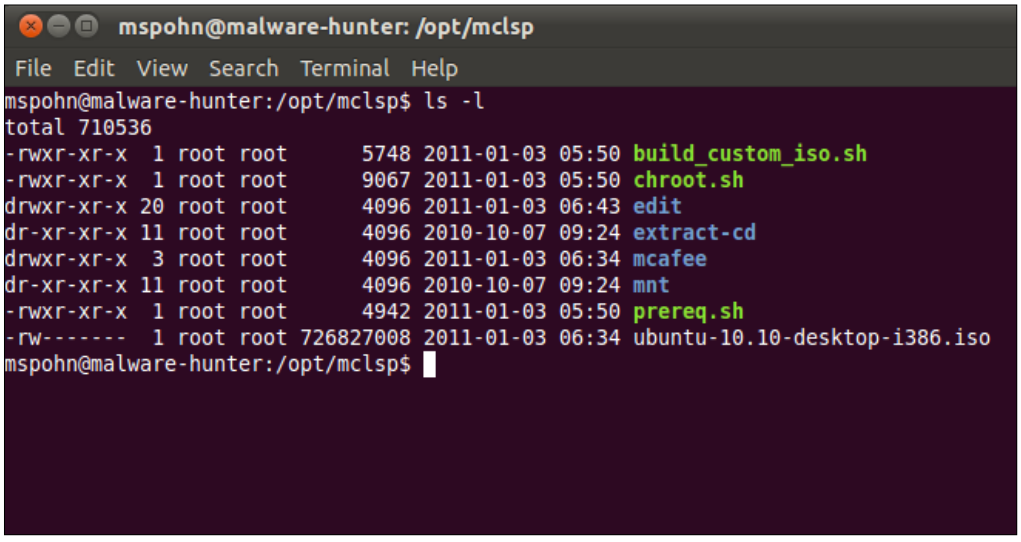

Figure 10: prereq.sh Folder Structure

Script chroot.sh

The second script you need to run is named *chroot.sh*. It is also located in the */opt/mclsp* folder. This script makes all the necessary changes to the base Ubuntu distro. Before running the script, you must change the location of the *root* filesystem on you host build system to */opt/mclsp/edit*.

The script performs the following tasks:

- Mounts pseudo devices on edit*/proc, edit/sys, and /dev/pts*
- Removes a large number of unwanted packages
- Downloads and installs the Wine package
- Downloads, compiles, and installs unzip 6.10b package
- Downloads and installs Wireshark
- Cleans up the package repository to save space
- Creates the required *initramfs* file (*initrd.lz*)
- Sets the default desktop background image to *av_background.png*
- Unmounts all mounted filesystems used in the build

Let's walk through the steps so you are clear on what is happening. You run the script with the commands:

cd /opt/mclsp

sudo chroot edit chroot.sh

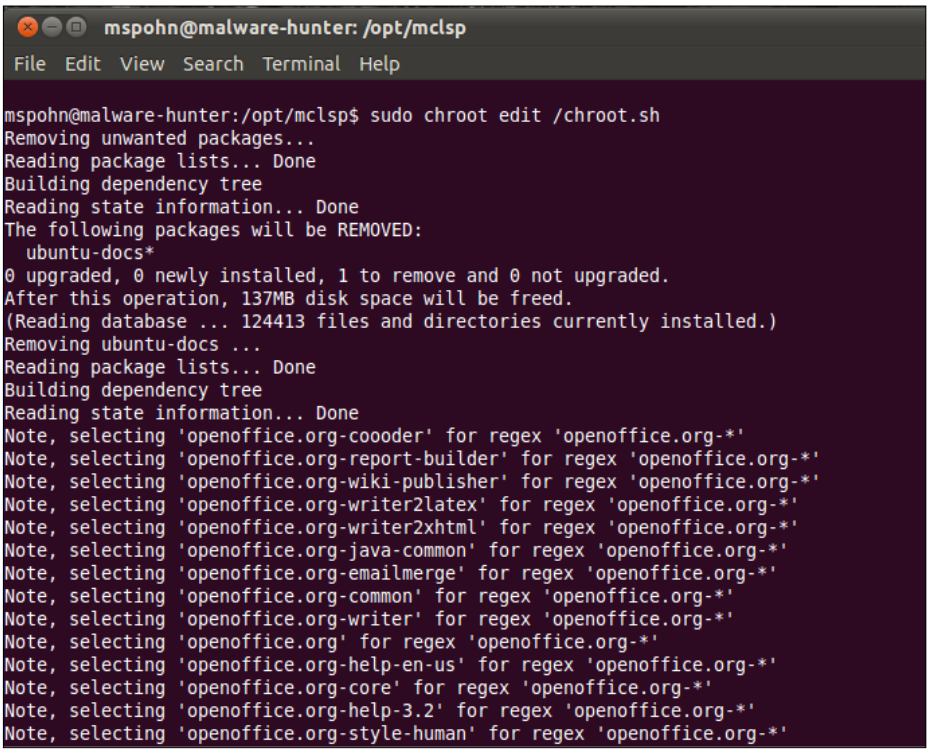

Figure 11: chroot.sh Script Execution

The *chroot* command switches the root of your host build filesystem to */opt/mclsp/edit*. This means that everything you do from this point forward is relative to */opt/mclsp/edit* not /. We need to do this so the changes we make in the script apply to the filesystem located at */opt/mclsp/edit.* The command also tells the *chroot* command to execute the script *chroot.sh* upon switches to root filesystem location. This script was copied to the *edit* folder by the *prereq.sh* script.

Next, the script mounts some pseudo devices to ensure the new root filesystem functions correctly.

The third thing the script does is very important. It removes a large number of unwanted packages to ensure our new custom distro will fit on a standard 700MB CD. We also need to make room for the Wine package which is very large. If you want to make additional customizations to your distro, you do them in the section of the *chroot.sh* script.

NOTE: If you decide to add or remove additional packages a stiff word of caution. When you purge a package from your custom distro, the package manager also purges *all* dependent packages. This is a good thing because it frees up a lot of space. A side effect of this – it breaks things. I spent *a lot* of time tinkering with this section of the script. I estimate I built more the 100 custom distros to determine what I could and could not purge. So be careful and test thoroughly. The only side-effect of adding packages is your ISO may grow beyond the 700MB limit of a standard CD.

The script next downloads and installs the Wine package. This is the package that allows us to run Windows binaries on Linux. I cannot tell you how much respect I have for the creators and maintainers of this project. It is truly a work of software engineering genius. I recommend you visit the Wine web site (http://www.winehq.org/) and take a look around.

Along the long road to getting this project to work, I discovered a very weird bug in the Linux unzip programs. Every once in a while, the unzip program failed to unzip the McAfee DAT files correctly. Instead of extracting the DAT file, crazy filesystem links were created. Extensive research into this problem revealed this is a known bug and is being corrected. To get around the issue, the script downloads, compiles, and installs a beta version of unzip (6.10b).

The script next installs the *Wireshark* package. This is such a handy tool, I decided to include it in the distro. For those of you who are not familiar with *Wireshark*, it is the world's best, and free, network sniffer.

Now that the script is done removing and installing packages, it makes sure the package manager repository is cleaned up. This is another effort to save as much space on our distro as possible.

Now that we have completed all the required changes to our custom distro, the script creates and compresses a new *initramfs* file. This file gets loaded into memory when your custom distro boot.

The next command in the script replaces the default Desktop background image to point to our *av_background.png* file. Finally, the script unmounts the mounted filesystems it used during the build.

Build_custom_iso.sh

The final script you need to run is named *build_custom_iso.sh*. It is located in the */opt/mclsp* folder. This script places our newly tweaked custom distro O/S into an ISO image.

The script performs the following tasks:

- Ensures the mounted files systems used in the previous scripts are unmounted
- Copies our new initrd.lz to the *extract-cd/capser* folder
- Creates the */home/mcafee* folder on our new distro
- Copies the McAfee A/V files, scripts, and config files to /home/mcafee
- Creates a global auto-run script to fire off the *av_scan.py* file on boot
- Downloads the latest DAT from the McAfee website
- Unzips the McAfee A/V Command Line Scanner product package
- Unzips the downloaded DAT zip file
- Removes all zip files from */home/mcafee* to save space
- Creates the required filesystem manifest for squashfs
- Creates a new *filesystem.squashfs* containing our custom changes
- Creates and MD5 file listing
- Creates an ISO named *Ubuntu-10.10-desktop-i386-mcafee-av.iso*

You run the script with the commands:

```
cd /opt/mclsp
sudo ./build_custom_iso.sh
```

The script first makes sure the filesystems mounted in earlier scripts are unmounted. Next, the script copies our newly created custom *initrd.lz* back over to the *extract-cd/casper* folder. As you may recall the *prereq.sh* script extracted the original file from the base image ISO and copied to our *edit* folder. Here we are putting our new one where it belongs on the new image.

The script next creates the */home/mcafee* folder in our new distro. This is where all our A/V scanning components will be placed. The script then copies all the required files to this folder.

NOTE: All files located in the /opt/mclsp/mcafee folder will be copied to the /home/mcafee folder on your new distro. Place any additional files you need in /opt/mclsp/mcafee including any EXTRA.DAT's you receive from McAfee Labs. This will ensure the A/V scan engine can find the EXTRA.DAT when scans are initiated.

Next, the script make a change to the Gnome Desktop manager configuration files to ensure our *av_scan.py* Python script gets executed when the system boots.

The script now goes out on the Internet to download the latest McAfee DAT from the McAfee web site. Once this completes, there are two zip files in the */home/mcafee* folder; the latest DAT zip file and the McAfee A/V command line scanner product zip file. The script will first unzip the McAfee A/V command line scanner zip file first because it contains old DAT files. If we unzip the latest DAT zip file first, the files will be overwritten with the old files in the product zip file. Once the zip files our unzipped, the script deletes the zip files to save space on the ISO.

The script next creates a file manifest that will be used by the *squashfs* tool. The *squashfs* tool is then executed and a custom *filesystem.squashfs* is created. This file is placed in the /extract-cd/casper/ folder. An MD5 sum file listing is created and the mkisofs tool is executed. This creates our custom ISO. The ISO name is *Ubuntu-10.10-desktop-i386-mcafee-av.iso.*

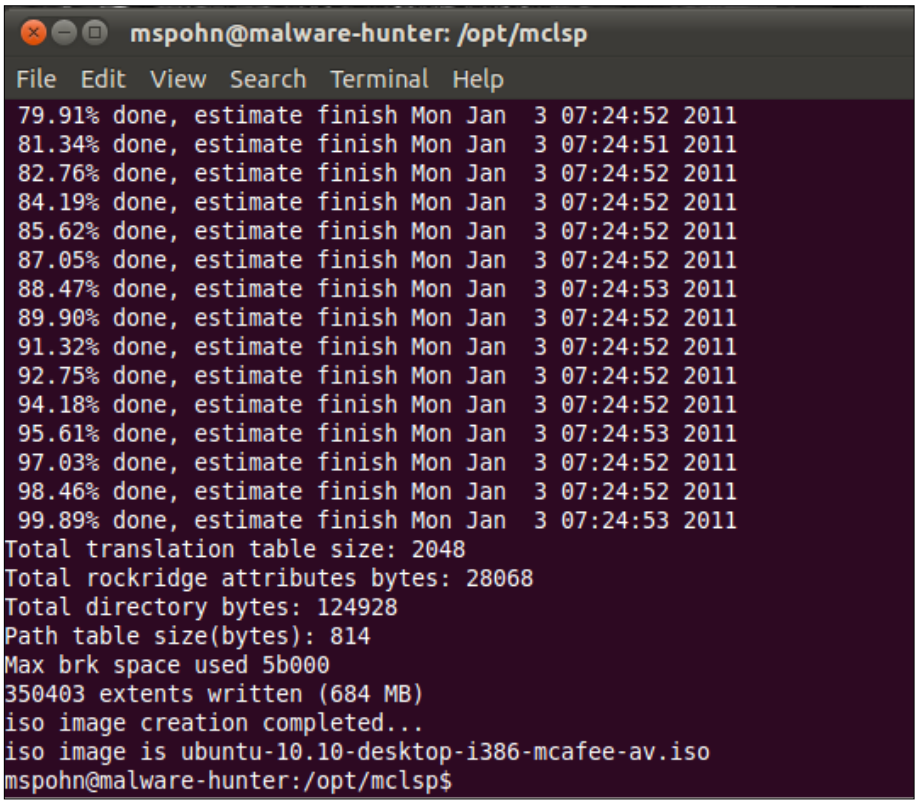

Figure 12: build_custom_iso.sh Script Execution

If your build environment is configured correctly and the three build scripts executed without any errors, you should have your custom Ubuntu McAfee A/V scan distro ISO sitting next to your Ubuntu 10.10 base image in */opt/mclsp*. This is shown in Figure 13.

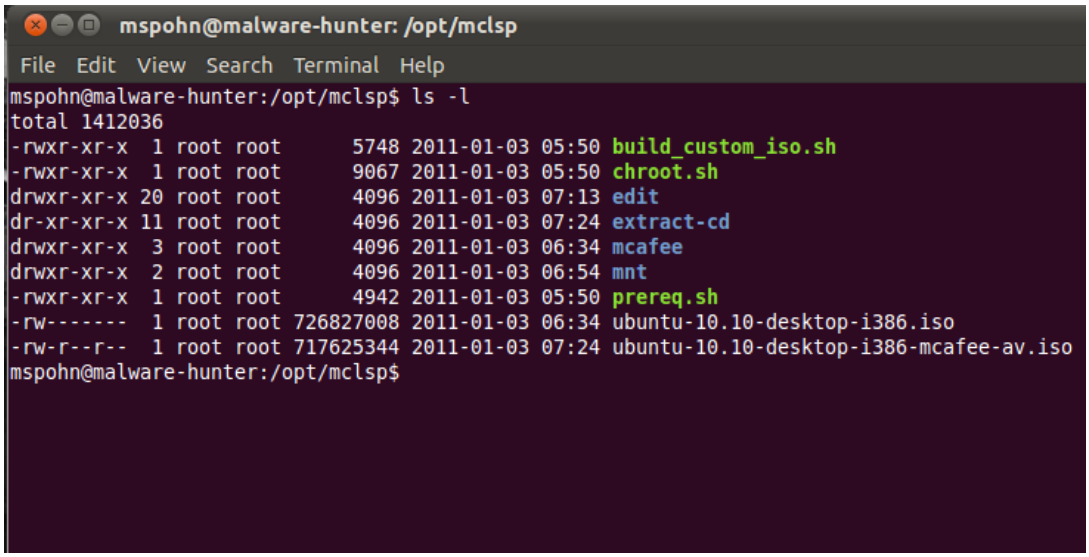

Figure 13: Completed Build Custom ISO

Deployment

Now that you have a custom ISO, you need to burn the image to bootable media such as a CD/DVD or USB thumb drive. This is easily done using open source tools. On Linux, my tools of choice are *Brasero* for burning CD/DVD's and *UNetbootin* for burning ISO's to thumb drives. You can install these apps on your build system using the Synaptic Package Manager or do it quickly from a terminal window using the below commands:

```
sudo apt-get install brasero
sudo apt-get install unetbootin
```

Once installed, you can access these applications from the *Applications* | *System* menu. You now can create bootable media to automatically boot and scan Windows hosts for malware. Figures 14 to 16 show the *Brasero* ISO burning process.

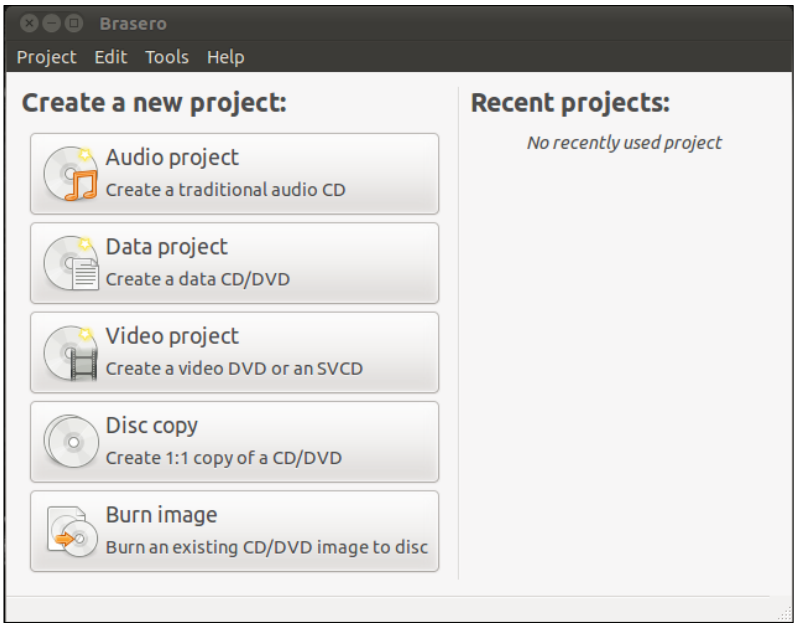

Figure 14: Brasero Project Dialog

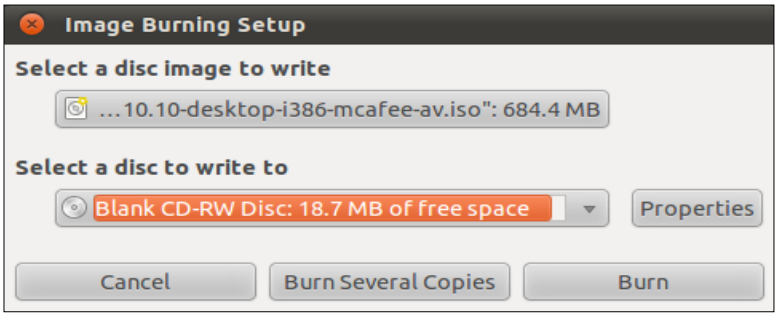

Figure 15: Brasero Image Burning Setup Dialog

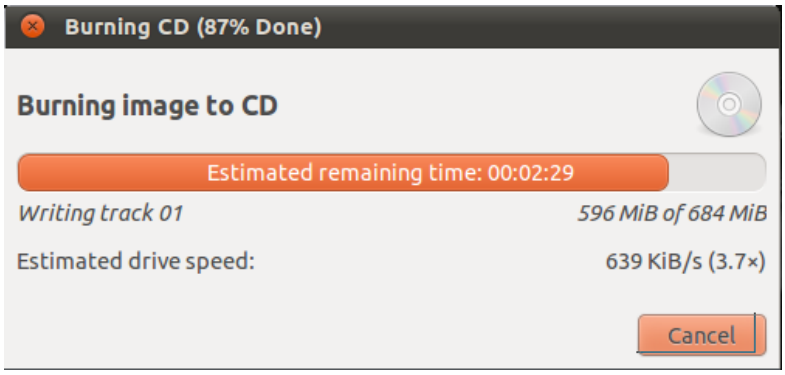

Figure 16: Brasero Image Burning Dialog

If you prefer to use USB thumb drives to scan systems *Unetbootin* will burn an ISO to a bootable thumb drive. Figures 17-22 show the *Unetbootin* bootable USB thumb drive steps.

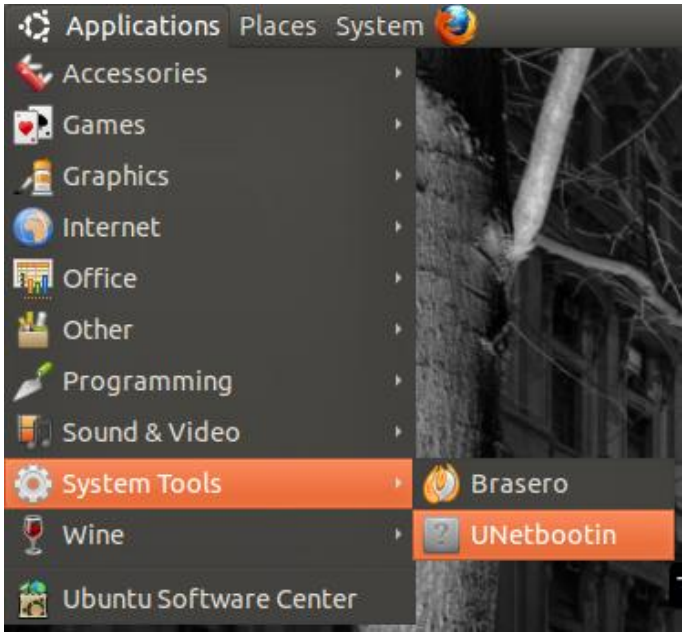

Figure 17: Unetbootin Launcher

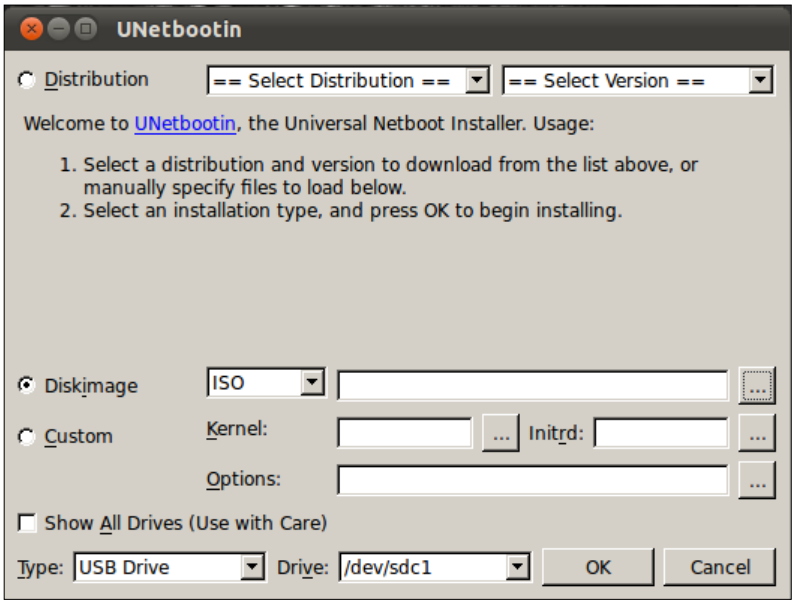

Figure 18: Unetbootin ISO Selection Dialog

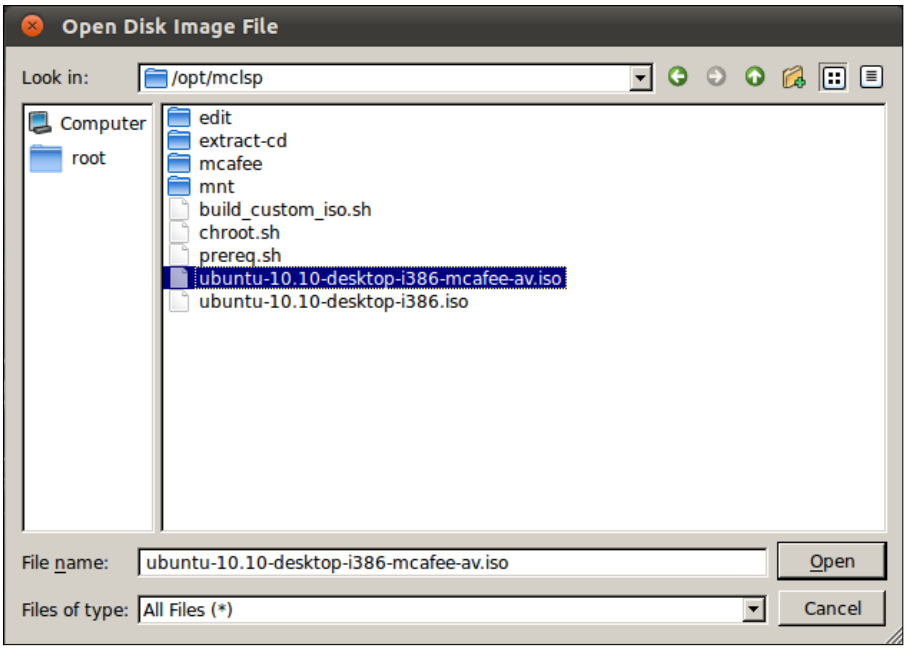

Figure 19: Unetbootin Open Disk Image Dialog

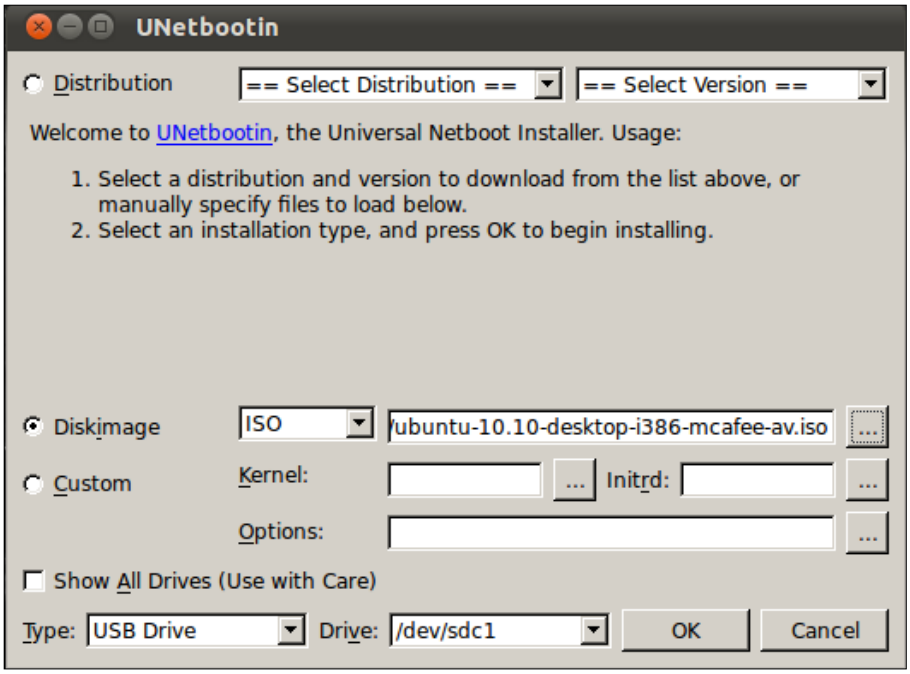

Figure 20: Unetbootin Build Dialog

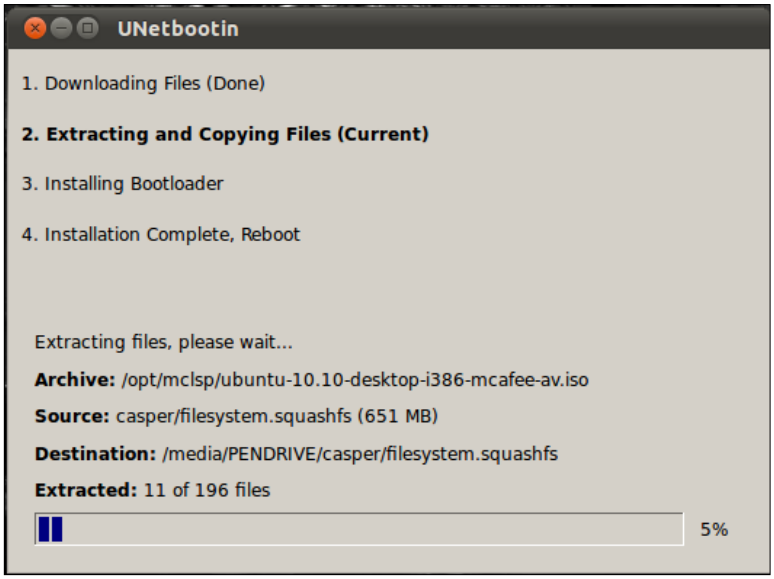

Figure 21: Unetbootin Progress Dialog

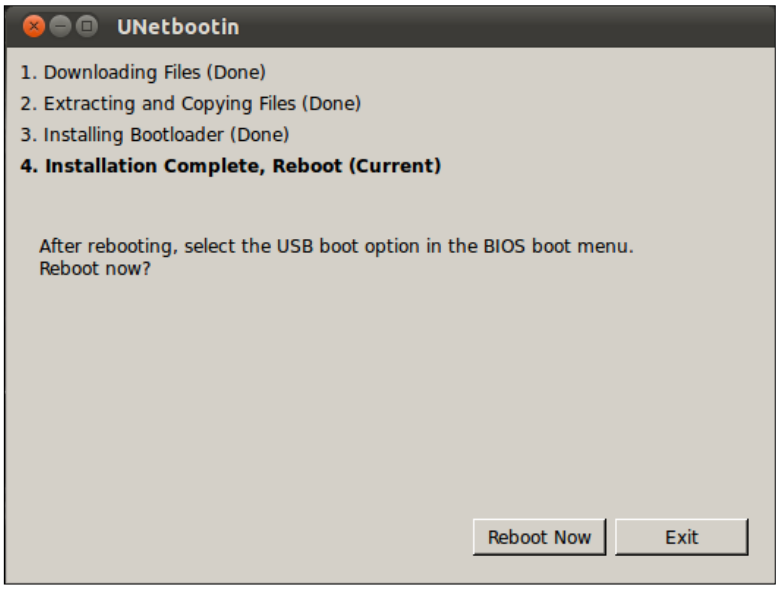

Figure 22: Unetbootin Completed Dialog

For those of you that prefer to do your ISO burning under Windows, my tool of choice for creating CD/DVD's from ISO images is InfraRecorder. If you are running Windows 7, you can use Windows File Explorer to burn an ISO. To burn an ISO to a thumb drive I really like *PendriveLinux*.

Scanning Windows Hosts

The work of creating a custom distro is now behind us. Let's boot a system with our distro to see how it works. When a system is booted from your CD/DVD or thumb drive, a user named *ubuntu* is created and automatically logged into the Gnome Desktop. The /home/mcafee/*av_background.png* is set as the background image and the /home/mcafee/*av_scan.py* Python script is executed.

The *av_scan.py* script is fairly large at more than 560 lines. If you are a Python wiz, the script should be easy to follow. If you are new to Python and want to learn its syntax, you should spend some time analyzing the script.

The *av_scan.py* script performs the following tasks:

- Processes the command line arguments
- Opens a temporary log file
- Downloads files from SMB share if parameters provided on command line
- Reads the *av_config.ini* file and sets appropriate configuration variables
- Opens the real log file and closed the temporary one
- If AutoScan is set to 0 then the script exits
- Executes *wineconsole* to ensure the *Wine* environment is initiated properly
- Downloads SMB files from location in INI [SMB] section
- Downloads latest McAfee DAT from McAfee web site
- Determines the host disk configuration
- Creates the mounted drive letters for *Wine*
- Mounts FAT/NTFS volumes that are not already mounted
- Identifies and prints McAfee scanner and DAT versions
- Creates the report path based on the INI *report_path* variable (/tmp/mcafee)
- Starts the McAfee Command Line Scanner using the *av_options.txt* file options
- Deletes the mounted drive letters
- Closes the log files
- Displays the scan output reports in *Gedit*

The script starts by processing command line arguments. You can specify an SMB share, user name, and password on the command line if you want the script to download files before the rest of the script runs. This is an advanced capability that most users will not use.

The script opens a temporary log file prior to reading the INI file. The name of this temporary log file is */tmp/av_error.txt*. If the *av_scan.py* script does not execute correctly be sure to review this log file for hints on what went wrong.

The script next reads the configuration file *av_config.ini* and sets some global variables. It opens the real log file (/tmp/mcafee/*autoscan_log.txt*)and closes the temporary one. If the [AutoStart] section *autostart* variable is any value other than "1", the script ends.

Next, the script executes the Wine package *wineconsole* command line application. This will ensure that Wine is configured properly for the *ubuntu* user. Alert users will see a small popup window advising that *Wine* is being configured if the current environment is not properly set.

The script will download files from an SMB share if the correct variables are set in the INI [SMB] section. This is a powerful feature because it allows you to place all of the McAfee scan files, DAT;s, and scripts in one location. It will also download the latest DAT from the McAfee web site if the INI [DAT] section variables are set.

NOTE: The network downloads will only work if the Windows host we boot is connected to the network and has a fixed IP address or is configured to use DHCP.

Next the script will examine the host disk configuration, specifically looking for FAT/NTFS volumes. It will mount all FAT/NTFS volumes that are not mounted and assign drive letters to them as required by *Wine*. The first drive letter used is *m.* Subsequent drives will be assigned *n:, o:*, etc.

Prior to starting the A/V scanner, the script determines the McAfee scanner and DAT versions and prints them on the console.

Finally, the McAfee command line scans are started. Each drive is scanned in alpha order and each drive will generate its own scan report. When the scans are complete, the drive letters are deleted, the log file is closed, and the scan reports are displayed in Gedit.

NOTE: Remember, the default configuration files provided with this project only *report* malware found on Windows volumes. Any malware found during scans can be found in the file */tmp/mcafee/av_badlist.txt* after the scans complete.

Figures 23-25 below show command line scans in progress and the reports provided after the scans complete.

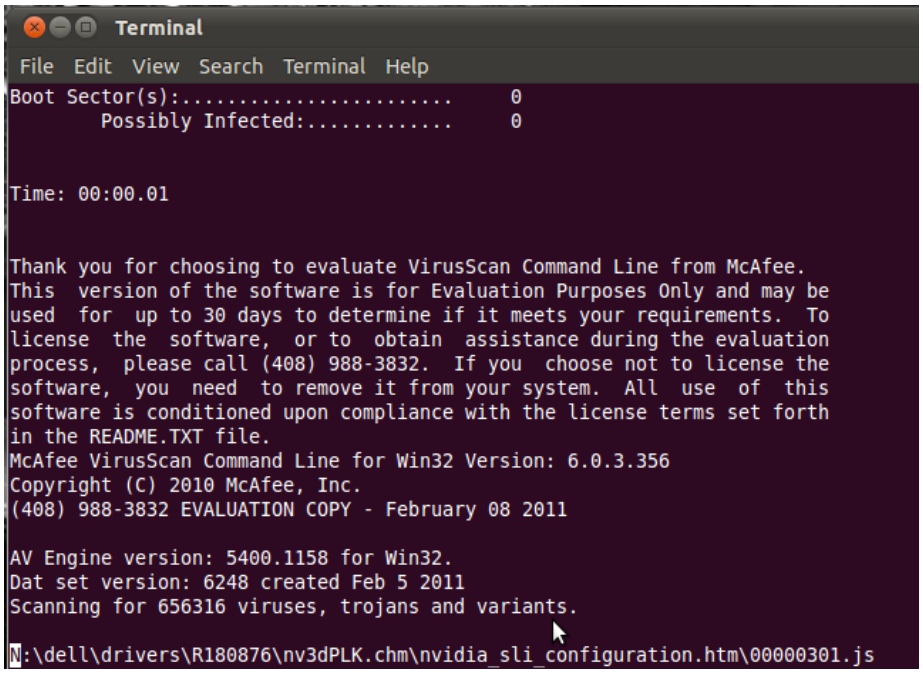

Figure 23: McAfee Command Line Scan In Progress

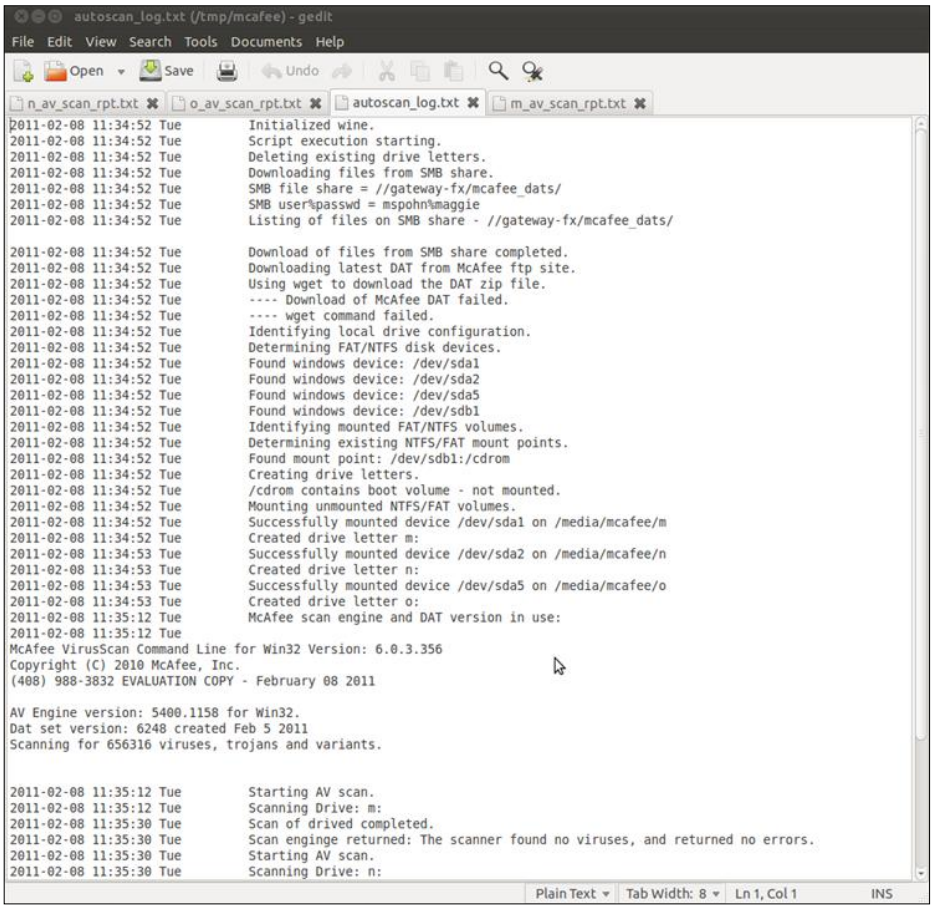

Figure 24: autoscan_log.txt

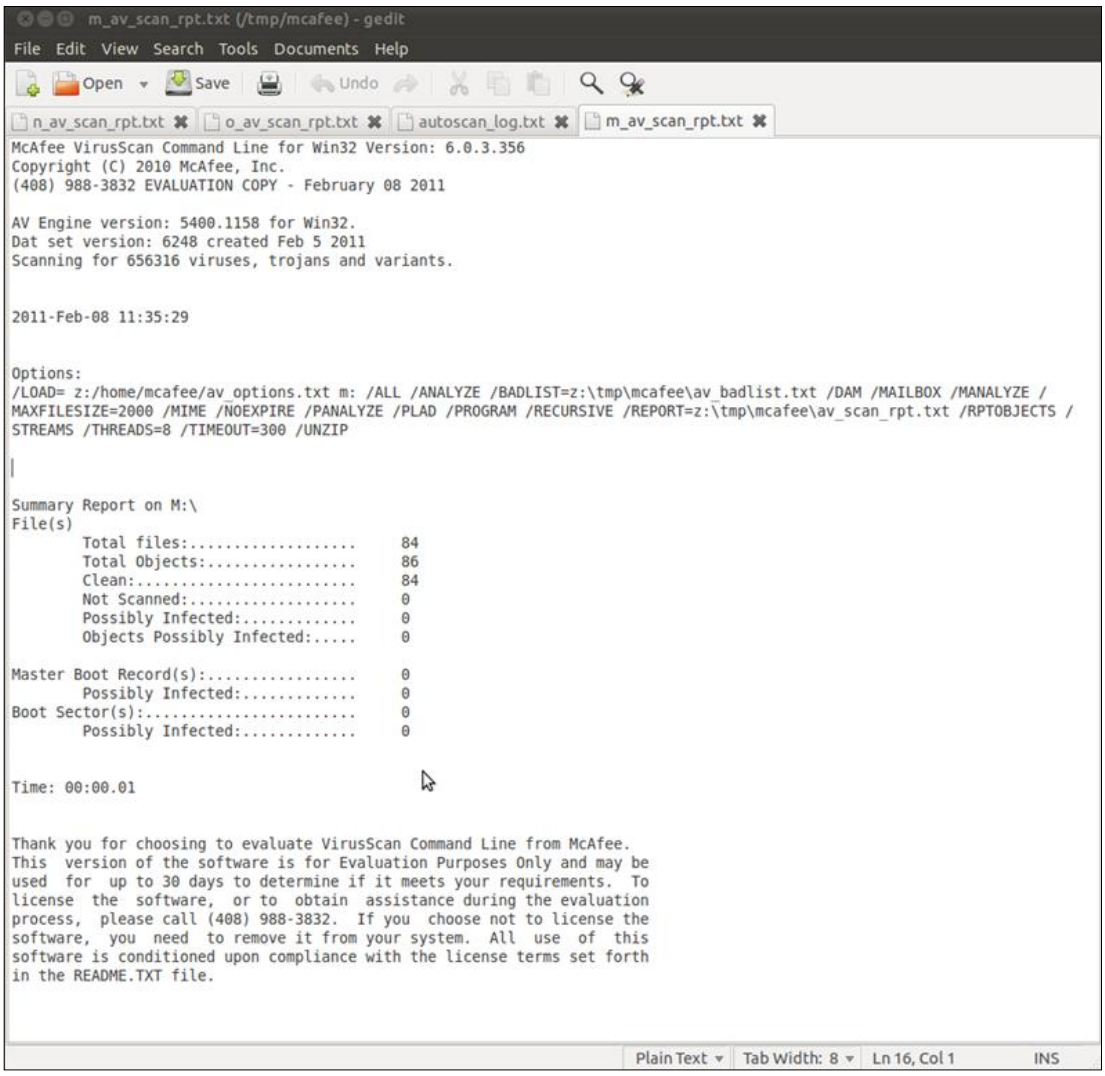

Figure 25: McAfee A/V Volume Scan Report

Troubleshooting

I spent a lot of time tweaking the build environment to make it as easy as possible for you to create a custom Ubuntu distro to clean systems of malware. That said, the three build scripts do a lot of work and there are many moving parts. If you are having trouble with your builds, review the below items to assist in your troubleshooting efforts.

- *You must be root to install and execute the scripts*

  There is no way around this since we must mount file systems and make changes to the environment. Be sure to use the *sudo* command (preferred), or login as root prior to running the build scripts.

- *Make sure your build environment is the same version as the base re-master*

  In other words – your host build environment should be the same version of Ubuntu Linux as the base system you are going to re-master. If you use *Ubuntu-10.10-desktop-i386.iso* for your custom build then you should be building the custom ISO on *Ubuntu 10.10 Desktop.* I have not experienced it, but there is a lot of discussion on the Internet about complications of mixing hosting and build versions of custom distro's. Save yourself some headaches and make sure your host and build environments are the same.

- *Make sure you have connectivity to the Internet*

  The build scripts download packages and files from the Internet so be sure your build host has Internet access before you try to build a custom ISO.

- *Make sure the chroot environment has networking configured properly*

  The *prereq.sh* script copies your build system *resolv.conf* and *hosts* file to the chroot location (/opt/mclsp/edit). When running the *chroot.sh* script, if there are problems connecting to the Internet, you need to ensure the chroot environment network settings are configured properly. You may have to edit the *prereq.sh* script as required.

- *Make sure you run all three scripts in the proper order*

  Once you have successfully built a custom ISO, you only need to re-run the chroot.sh and build_custom_iso scripts to ensure your build has the latest McAfee DAT file. If you are having trouble successfully building a custom ISO, reboot your host system and run the three scripts again in the correct order: *prereq.sh, chroot.sh* and *build_custom_iso.sh.*

If you are still having trouble, send me an Email describing the issue and I will look into it when I can. mspohn@malware-hunters.net.