MINT IT Systems Documentation

Rainer Karcher

August 2013

Contents

1 Structure of this Document

The foundation of this documentation is largely credited to former MINT IT Student Assistant Paul Kilgo, while I have made numerous additions as necessary, as well as semantic revisions for accessibility.

Each major section of this document will be focused on a particular machine or piece of equipment and each subsection will focus on a component of that machine. In each section, relevant listings, file system locations, or any other points of information will be appended after the main text of the section.

2 Buzz

Buzz is the hostname of the MINT Center's general-use Linux system. At the time of writing, it runs on CentOS 5.5. It hosts many utility-type services to help sustain day-to-day operations at MINT, the most important probably being the security cameras, faculty and student web sites, and the system imager. The hardware itself was purchased in the summer of 2007 and has since remained in active use.

2.1 History

Buzz initially ran Red Hat Enterprise Linux (RHEL) 5, as it was pre-installed when it was originally purchased. Its original purpose was to act as a web host for a summer project which has now fallen into disuse. Over the course of its life at MINT, it has assumed more versatile duties.

2.1.1 Security Notes

A system intrusion occurred at the end of 2010, and the system needed to be re-loaded from scratch to ensure that no backdoors were left behind. During the course of this reload we made the switch to CentOS, as its system environment is virtually identical to RHEL, but lacking the Red Hat Network support (which we never used, anyway). The specific intrusion occurred when an outdated version of phpMyAdmin left an opening to download some remote code and execute it (specifically, SQL injection). Presumably, the attacker only gained control of the web server user. The exact route to rooting is not known, but it presumably occurred either by breaking the root password (since a relatively weak hashing algorithm (MD5) was used), or by means of an outdated component of the machine that left it vulnerable to a rootkit.

On system reload, we took precautions to prevent this from happening again. Namely, we left SELinux running on this installation since it includes policies to prevent rogue web applications from behaving unusually. Additionally, the system has also been set to use SHA512 for password hashing, though a few users on the system still have their passwords encrypted using MD5. These passwords will need to be changed in order to hash with SHA512.

Relevant commands to this section:

```
## Shows current hashing algorithm
$ authconfig --test | grep hashing

## Sets password hashing algorithm to sha512
$ authconfig --passalgo=sha512 --update

## Forces user to change password on next login (an actual shell must be used, not scponly, etc.)
$ change -d 0 [username]
```

2.2 General System Configuration

Buzz's configuration doesn't deviate too far from a default CentOS installation. There are, however, a few bits important for security reasons or maintaining certain services, all of which will be covered in the following.

2.2.1 Using git and gitosis

We do our best to try to keep Buzz's entire system configuration in source control using a git repository and gitosis. For the uninitiated, more information on using git/GitHub can be found at the end of this section.

There are a few repositories in use on Buzz, all of which are located in the directory specified at the end of this section: `sysconf`, `gitosis-admin`, `documentation`, and `signup`

- `sysconf` is the repository which contains the major deviations, or forked branches, in system configuration
- `gitosis-admin` is the gitosis administration repository, where new users can be added to source control
- `documentation` contains the repository for this documentation
- `signup` contains the source code for the faculty and student account creator.

The preferred method for making a change to system configuration is to pull a working copy from gitosis as root, make the changes, copy them to their destination, commit them when they

are verified as working (don't forget to add a message describing the commit), and push back to gitosis. The attraction of this method allows for reversion back to any previously committed system configuration state in the event of some catastrophic meltdown. The `sysconf` repository is probably the most useful one, and its file structure mimics the system's exactly – the directory `/etc/motion` would be `etc/motion` inside the `sysconf` repository.

Relevant commands to this section:

```
## Clone a repo called <name>
$ gitclone git@buzz.mint.ua.edu:name.git

## Add a previously nonexistent file to your source control
$ git commit –am 'Making the following change(s)…'

## Push back into the main repository
$ git push origin master

## IF YOU MESS SOMETHING UP (a likely occurrence!)
## Remove all changes/revisions which have not been 'git added' (not in the index directory)
$ git checkout .

## Remove changes from the git index (not the file system)
$ git reset

## Revert changes that you had previously committed
$ git revert …
```

There is much, much more to git than I can specify in this document, so I've included a couple of articles below which should be helpful to both seasoned users and the uninitiated.

Relevant files/directories to this section:

- `/home/git/repositories`

Helpful links relevant to this section:

- Lifehacker Article: "How The Heck Do I Use Github?" http://lifehacker.com/5983680/how-the-heck-do-i-use-github
- Git for Beginners: The Definitive Practical Guide http://stackoverflow.com/questions/315911/git-for-beginners-the-definitive-practical-guide
- How to Transfer Your Existing Git Repository to Github http://kovshenin.com/2011/transfer-your-existing-git-repository-to-github/

2.2.2 The Firewall

We use iptables for our firewall settings, which is the default firewall configuration application for CentOS. As for the 'custom tweaks' mentioned earlier, these consist of two rules:

1. A rule that refuses SSH packets from any host which tries to connect more than 3 times within a single minute. Essentially, this prevents the system from being saturated with an overwhelming number of external communication requests from another machine, thereby preventing denial-of-service attacks. This custom chain is titled `'LIMIT SSH'` in iptables.
2. A special exception which allows for traffic on UA's campus networks.

Relevant files to this section:

- `/etc/sysconfig/iptables`

Relevant commands to this section:

```
## Restart iptables, e.g., to apply configuration changes
$ service iptables restart
$ /etc/init.d/iptables restart
```

Helpful links relevant to this section:

- Using and editing iptables – guide for complete beginners
  http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch14_:_Linux_Firewalls_Using_iptables#Introduction

2.3 File System

The file system of buzz is a pretty standard setup. As for the usual Linux mounts, we have 113G allocated to the root, 200G for `/home`, 4G for `/var`, and 2G for `/boot`. Buzz is also hooked up to a large external drive store which is 1.2T.

Its mountpoint is `/aos`. The `/aos` mountpoint is where we store miscellaneous disk-hungry items that we don't want clogging up the file system. Any backup systems should backup their files to `/aos/backup`.

2.4 Backups

Buzz runs a couple of automated backup scripts which are intended to pull the contents of `/home/www/` from the MINT webserver, via rsync, to be stored on Buzz in the `/aos/backup/` directory.

The backup scripts are initiated via a `cron` job (with password-free pubkey authentication) executed by a locked system user called 'backup'; the remote system (in this case, the MINT webserver) is set up to authenticate the private key of the 'backup' user on Buzz, so no passwords are required.

Relevant commands to this section:

`Crontab` for the backup user on buzz.mint.ua.edu
Install by running `'crontab -e'` as the backup user

```
@hourly             $ run-parts /aos/backup/bin/cron.hourly \
              > /aos/backup/log/cron.hourly.log


@daily              $ run-parts /aos/backup/bin/cron.daily \
                         > /aos/backup/log/cron.daily.log

@weekly             $ run-parts /aos/backup/bin/cron.weekly \
                         > /aos/backup/log/cron.weekly.log
```

Relevant files/directories to this section:

- `/aos/backup/`

2.5 System Imager

Clonezilla is the application we use to for disk imaging - using a Clonezilla Live CD/USB, a system administrator can create and restore operating system images to computers. In the server room (Rm. 1065), Buzz is outfitted with a switch mounted to its alternate network interface (192.168.50.1), to which a system in need of imaging can be connected. Currently, Buzz is configured to use PXELINUX to deliver a host of existing operating system images to a PXE-capable client (such as Clonezilla), but alternatively one can image a system using an operating system image ISO downloaded from a project's website.

The images available for use by Clonezilla are stored in the `/aos/partimag/` directory, though Clonezilla favors the default path `/home/partimag/`, so on Buzz there is a softlink in place which redirects Clonezilla to `/aos/partimag/` accordingly. Also, Buzz contains a user

group called '`clone`', and any user assigned to this group should be able to read data from the proper locations to perform a system image clone.

When stationed in the server room, an administrator should use NFS as the access method to the images (this method is addressed in greater detail in section 2.5.5). When working from any other location, an administrator should use SSH and provide his or her login credentials to perform the image clone (this method is addressed in greater detail in section 2.5.4).
The administrator's user account must be in the '`clone`' group to clone via SSH.

PXELINUX is currently housed in the `/tftpboot/` directory on Buzz, configured using `/tftpboot/pxelinux.cfg/default`, and the current roster of bootable live OSes are found in `/tftpboot/dists/`. The PXE boot configuration is a little hairy and involves exacting coordination of TFTP, a DHCP server, PXELINUX, and NFS, so there is a lot of room for things to go awry. That being said, backup via SSH is the most solid choice available at this time.

The following steps cover DHCP server configuration for PXE usage (to let DHCP clients know that PXELINUX exists), in the event that anything catastrophic happens to the current configuration file:

1. Edit your `dhcpd.conf` file (i.e.: `$ vim /etc/dhcpd.conf`)
2. In the global section or in a subnet declaration add `filename = "pxelinux.0"`
3. Restart your dhcpd server (i.e.: `$ /etc/init.d/httpd restart`)

Relevant files/directories to this section:

- `/aos/partimag/`
- `/tftpboot/`
- `/tftpboot/dists/`
- `/tftpboot/pxelinux.cfg/default`

Helpful links relevant to this section:

- Clonezilla Live Guides & Documentation
  http://clonezilla.org/clonezilla-live-doc.php
- PXE Server Configuration on CentOS 5
  http://www.linuxphobia.com/2011/10/pxe-server-configuration-on-centos5.html

2.5.1 Booting from PXE

When connecting to the switch at the workbench in the server room, boot the machine to PXE and you'll be instructed to enter the name of a live OS you want to boot. If you type '`gparted`' and hit 'Enter', the machine will boot to GParted Live, and if 'none' is selected, the machine will boot to Clonezilla Live. If booting Clonezilla Live, a menu displaying the available OS images

(served from the `/tftpboot/pxelinux.cfg/boot.txt` file) should come up; this method is covered in further detail in section 2.5.3.

Relevant files/directories to this section:

- `/tftpboot/pxelinux.cfg/boot.txt`

2.5.2 Sysprepping for Windows XP

Note: Before attempting to use Sysprep, make sure to do the following:

1. Shrink the destination partition with GParted (failing to do so prior to beginning the process will be problematic later)
2. Make sure that you have blanked out the local administrator password (otherwise you'll have a pesky dialog forcing you to click "OK" dozens of times)
3. Inside the Sysprep directory, run sysprep.exe. Once everything is ready to go, check "MiniSetup" and "Pre-activated" and click the "Reseal" button.
4. Grab a book, because sometimes it takes a while, but sometimes not (and I still have no idea why).

Helpful links relevant to this section:

- How to use the Sysprep tool to automate successful deployment of Windows XP http://support.microsoft.com/kb/302577

2.5.3 Using Clonezilla Live (start here)

Note: Before saving, be sure you have shrunk the partition and sysprepped!

1. Boot Clonezilla Live on destination machine (via PXE or disk)
2. The first two questions don't really matter, so just hit 'Enter'.
3. Start Clonezilla
4. Select "device-image"
5. Continue reading at "Imaging over SSH" (section 2.5.4) or "Imaging over NFS" (section 2.5.5)

2.5.4 Imaging over SSH

1. Select "ssh server"
2. Select "dhcp"
3. Type in "buzz.mint.ua.edu" for the server
4. Leave the port as 22
5. Type your system username on Buzz
6. Leave next field as "`/home/partimag`"
7. Type "yes" to accept the fingerprint.
8. Type your password.
9. Continue reading at "Save an image" (section 2.5.6) or "Restore an image" (section 2.5.7)

2.5.5 Imaging over NFS

Note: This should only be done at the workbench in the server room.

1. Select "nfs server"
2. Leave the default IP as is (192.168.50.1)
3. Leave the next field as "/home/partimag"
4. Continue reading at "Save an image" (section 2.5.6) or "Restore an image" (section 2.5.7)

2.5.6 Saving an Image

Note: Before you save an image, you should first shrink the partition down to a widely compatible size (approx. 40GB usually works). Clonezilla will hate you if you try to image something massive onto a machine which can't support it. If you use GParted to do it, boot up the machine, reseal, then shut down. Clonezilla doesn't like it when you shrink with gparted because it thinks the NTFS partition is corrupted. So, it's best to boot it up and let Windows fix it. (Also, you should always reseal Windows images!)

1. Select "savedisk"
2. Type in a name for the new image.
3. Select which disks you want to image.
4. Continue as instructed.

2.5.7 Restoring an Image

1. Select "restoredisk"
2. Select the image you want to restore
   ('hda' images are for PATA and 'sda' images are for SATA drives)
3. Select the destination hard drive.
4. Continue as prompted on the screen.

2.5.8 System Imager: Caveats

When saving an image via NFS (which writes as root) and then trying to read the image via SSH (which reads as the '`clone`' group), problems could arise. If you get any odd errors while working via SSH, SSH into Buzz and use chmod to reconfigure group read+write permissions in the `/aos/partimag/` directory (covered in detail below).

Relevant files/directories to this section:

- `/tftpboot`
- `/tftpboot/pxelinux.cfg/default`
- `/etc/dhcpd.conf`
- `/etc/exports`

Relevant commands to this section:

```
## Fix Clonezilla permissions (as root)
$ chown -R root:clone /aos/partimag/
$ find /aos/partimag -type f exec chmod -R 0664 {} \;
$ find /aos/partimag -type d exec chmod -R 0775 {} \;
```

2.6 Motion

Motion is the daemon on Buzz which polls the security cameras (all of which are IP-based) for motion activity. Most of motion's activity happens in `/aos/cameras/`, where you'll find all the images that it captures. The cameras also have a share on the Samba server, where members of the '`security`' group can view the archived snapshots and videos.

Motion is configured via its config file, `/etc/motion/motion.conf`. Each camera is represented by a 'thread' which is listed at the bottom of the main config file. There are subdirectories in `/etc/motion` which correspond to each 'domain' of cameras (MINT/CAF/MFF). Each night, video files are generated for the previous day's images. As an

added bonus, it will enlace a random epic track to play alongside the video to help pass the time. The epic tracks are grabbed from the `/aos/pub/epic/` directory by the script `make-epicvideo`.

Relevant files/directories to this section:

- `/aos/cameras/`
- `/etc/motion/motion.conf`
- `/aos/pub/epic/`
- `/aos/cameras/bin/make-epicvideo`

Helpful links relevant to this section:

- Motion (Developers' Wiki)
  http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome

2.6.1 Archived Motion Image Directories

The motion image directories follow this hierarchy:
`/aos/cameras/images/<domain>/<timespan>/<name>/<YYYY>/<MM>/<DD>`

- `<domain>` is either `'mint'`, `'mff'`, or `'caf'`;
- `<timespan>` is either `'two-week'` or `'forever'`;
- `<name>` is the name of the camera;
- `<YYYY>`, `<MM>`, and `<DD>` are the year, month, and day respectively.

2.6.2 Cron

The `cron` jobs are pretty important to the cameras, as they do various things such as deleting old camera footage, generating videos, and the like. Check out the `'sysconf'` git repository `/aos/cameras/crontab/` to see what the suggested `crontab` should look like.

Relevant files/directories to this section:

- `/aos/cameras`
- `/etc/motion`

Helpful links relevant to this section:

- Intro to `Cron`
  http://www.unixgeeks.org/security/newbie/unix/cron-1.html

2.7 Web Server

Buzz is host to a few websites, namely the following:

- www.faculty.mint.ua.edu
- www.students.mint.ua.edu
- www.wiki.mint.ua.edu

(For brevity's sake, however, this document will assume that the wiki is defunct and will not address it specifically.)

All of these domains are configured as virtual hosts in Apache (within `/etc/httpd/`). In that directory, the configuration for each virtual host is in `/vhost.d/`, where any file ending in `*.conf` will be included in the Apache configuration. To add a new virtual host, an administrator can just copy one of the existing `*.conf` files and modify it accordingly. For the moment, all virtual host directories are being kept in `/home/www/vhost/.`

Relevant files/directories to this section:

- `/etc/httpd/`
- `/etc/httpd/vhost.d/`
- `/etc/httpd/vhost.d/clients.conf`
- `/etc/httpd/vhost.d/wiki.conf`
- `/etc/httpd/vhost.d/default.conf`
- `/etc/httpd/vhost.d/ex.conf`
- `/home/www/vhost/`

Helpful links relevant to this section:

- Apache VirtualHost Examples
  http://httpd.apache.org/docs/current/vhosts/examples.html

2.8 MySQL

MySQL is not heavily utilized on Buzz; there are only a few applications here and there which actually use it. As of writing, the only things which use Buzz's MySQL server are:

1. A long-dead and never-revived MediaWiki installation used by Dr. Mankey uses '`gmankey`';
2. An old Computer-Based Honors project using '`magbiomatls`';
3. The faculty/student account creator which uses '`mint`';
4. Dr. Leclair's Moodle installation which uses '`pleclair`';
5. Another student project which uses '`samplerecordsdatabase`'.

Relevant commands to this section:

```
## Create a new database (preferably run as root)
$ mysqladmin create <database name> -u root -p

## Bring up the MySQL prompt (preferably run as root)
$ mysql -u root -p

## Dump a backup sqldump
$ mysql -u root -p < dump.sql

## Grant all prvilieges to a local user on all tables in a database
$ GRANT ALL ON dbname.* TO 'username'@'localhost' IDENTIFIED BY
'password';
```

Helpful links relevant to this section:

- Short List of MySQL Commands
  http://www.bios.niu.edu/johns/bioinform/mysql_commands.htm

2.9 Account Creator

The account creator is the collection of scripts which make www.faculty.mint.ua.edu and www.students.mint.ua.edu possible. This was developed nearly six years ago from the time of writing, so the code is all over the place and uses both PHP for the web user interface and Perl for the user account creation script. This section will attempt to explain how this system works from a top-down perspective.

At the front end of the system are the PHP forms which provide username and password entry fields for faculty and students who are creating their accounts. This information is passed on to a newly-generated entry in the MySQL database, '`mint`', but with limited privileges until explicitly approved by an administrator.

An email to the administrators is generated stating that a new user has requested an account, with a randomly-generated key embedded in the URL, as the process assumes that anyone who guesses the correct key must be an administrator. The administrator may then, from their email inbox, approve or reject the account request. If the request is approved, the entry in the database is marked as approved. If it is denied, the MySQL database entry is removed and no action is taken.

The root user has permission to run the `modusers.pl` script, a `cron` job which polls the '`mint`' database for any approved users. If it finds newly approved user accounts, it removes the restrictions on the database entry and sets the user-defined password (specified when the user

submitted the original account request) into effect, sets proper permissions for the new user's home directory (`/home/username/`) and web directory (`/home/username/public_html/`), changes the SELinux context for the web directory, sets the disk quotas for the new user according to their user group, and emails the user to alert them that their account has been created successfully.

2.9.1 Changing the recipient of account creation scripted emails

The email addresses the scripts use are hard-coded in each of the separate subsystems. Check `modusers.pl` and `index.php` to modify the scripted email recipients.

Relevant files to this section:

- `/root/bin/modusers.pl`
- `/home/www/vhost/signup`

2.9.2 Account Creator: Caveats

Admittedly, many methods used throughout this system are not best practice. For example, the passwords are stored in plaintext in the MySQL database, which is never a good idea. This could be fixed by using PHP's `crypt()` function on the password and storing it, and when adding the user's password the hash could be specified manually.

Additionally, we trust this script too highly with regard to the security of email contents, which is also ill-advised. Passwords are transmitted in plaintext to the administrator(s) via email; the reason being for this was that the administrator could ensure that no fishy business was going on in the user requests (wouldn't want someone exploiting `moduser.pl` to gain root, etc.). If the aforementioned change were implemented, this would be moot, though even hashes shouldn't be transmitted via email.

On an extremely paranoid note, someone could potentially intercept the key embedded in the URL sent to administrators. This is unlikely (unless someone broke into the correct network closet), but still theoretically possible, though it would not be a catastrophe (that is, unless it actually was a catastrophe).

3 MINT Web Server

The MINT web server (www.mint.ua.edu) provides the main MINT website, as well as some various organization-centric Wordpress web sites. In theory, most of its configuration should be a lot like Buzz's since I tried to follow the same sort of scheme, although some of the file paths have changed slightly.

The following sites are hosted here:

- www.mint.ua.edu
- www.caf.ua.edu
- www.matsci.ua.edu
- www.mint.ua.edu/~ieee

All of the website domains hosted on the MINT web server (with the exception of www.mint.ua.edu/~ieee) are configured as virtual hosts in `/etc/httpd/conf/httpd.conf`.

Relevant files/directories to this section:

- `/home/www/`
- `/etc/httpd/conf/httpd.conf`
- `/home/sites/caf/`
- `/home/sites/matsci/`
- `/home/sites/ieee/`

3.1 MINT Website

The MINT website is defined as an Apache virtual host at the bottom of `/etc/httpd/conf/httpd.conf`. UserDir has been enabled for `/home/sites/` so a request made to `http://mint.ua.edu/ foo` will correspond to `/home/sites/foo`. The website's MySQL user is '`www live`', and the database has the same name. (The naming convention is a remnant of a previous effort to effect both live and staging versions of the website.)

3.2 WordPress Instances

All of the WordPress installations described in this section use a separate database from the main web site. You can easily make more by using the script tarball mkwpsite as user '`www`'. All of the web sites pull from `master-wp-config.php`, and the table prefix is guessed from the directory name in /home/sites from which it is hosted. For example, the web site at

/home/sites/ieee would have the MySQL table prefix site ieee , and the URL http://mint.ua.edu/ ieee/. The mkwpsite script is a quick way of setting a wordpress site up in case someone needs one. It can automatically fetch the latest WordPress release from wordpress.org or it can use a zipped release specified via command line. WordPress does have a multi-site "network" feature that allows one to run multiple WordPress sites from a single installation, so it may be good to check into this at some point and see if it would work for us.

Relevant files to this section:

- /home/sites
- /home/www/bin/mkwpsite
- /home/sites/conf/master-wp-config.php
- /usr/local/share/wordpress

Relevant commands to this section:

```
## Usage : mkwpsite <name> [<wordpress-zip >]
## Make site 'example' from latest code.
$ mkwpsite example

## Make site 'example' from a WordPress release zip 'wordpress-3.0.zip'
$ mkwpsite example wordpress-3.0.zip
```

Helpful links relevant to this section:

- Getting Started with WordPress
  http://codex.wordpress.org/Getting_Started_with_WordPress

4 CAF kiosks

The CAF kiosks are machines with no means of input aside from a touchscreen and a card reader used for checking out equipment. Some time ago, the CAF had serious problems with USB drives infecting the control computers with autorun viruses. To combat this, we put a series of shell scripts on the machines which will quickly (not necessarily completely) scan an inserted drive for viruses and report the result on-screen, quarantining files if necessary.

The machines reboot nightly and are more or less autonomous and attempt to update themselves and their associated scripts automatically. On boot, the machines auto-login to an 'av' user and execute Firefox in full-screen mode. Due to the lack of user input means, this is sufficient to keep most users (the ones without keyboard handy) from accessing other parts of the system. Also, on Gnome login, another scripted titled start-pollmount will execute.

Relevant commands to this section:

Gnome autostart file for `pollmount`
Goes in `~/.config/autostart/poll-mounts.desktop`
You can also create this in the normal people way by accessing the following:
System -> Preferences -> Startup Applications

```
[Desktop Entry]
Type=Application
Name=Poll-mounts
Exec=/home/av/bin/start-pollmount
Icon=system-run
Comment=Scan 4 viruses
Name[en_US]=Poll-mounts
Comment[en_US]=Scan 4 viruses
X-GNOME-Autostart-enabled=true
```

4.1 Virus-Scanner Scripts

The main virus-scanning script is titled `autoclean` and it works by exploiting characteristics typical of autorun viruses to perform its job more quickly than a full scan. Currently the scan heuristic entails these steps:

1. Check to see if there's an `autorun.inf` present. If so, open it. For each string to the right of an equals sign (=), check and see if that string refers to a file on the flash drive's file system. If so, quarantine it. Always delete the `autorun.inf` afterwards.
2. Remove any files ending in `.com` or `.pif`.
3. Remove directories at the root called 'Restore'.
4. Run a true virus scanner on the top-level directory and quarantine whatever it finds. This script by itself has been pretty reliable and hasn't been altered due to bugs in some time.

4.2 USB Drive Poller Script

The `poll-mounts` script runs all the time until killed for some reason. It's responsible for a lot of the GUI functions of the whole system (read: it makes use of `notify-send` to make the pretty messages come up). It depends on the `autoclean` script and as well another script called `check-for-usb`, which of course checks for mounted USB devices.

This script works by periodically copying `/proc/mounts` and running `diff` on two copies taken at different times to see if any drives have been added to the system. It dispatches `autoclean` when it detects a new drive added. A better way to do this might be using the event-based hooks of `udev`, but orchestrating that seemed like torture.

4.3 Automated Script Deployment

The machines will check for new versions of `poll-mounts`, `check-for-usb`, and `autoclean` via a `cron` job which runs the script `update-scripts`. If the scripts are updated, it kills all user processes for '`av`' user, restarting Gnome and forcing a reload of `poll-mounts`.

Currently they check for updates in `http://buzz.mint.ua.edu/deploy/viromotrons`. Any updates to the scripts in that web directory will be applied to the kiosks overnight.

Relevant files to this section:

- `/usr/local/bin` (main scripts)
- `/root/bin` (updater scripts)