CCDC Greg’s Hardening Notes

* Make sure to run Apache under its own user
  + set shell to /sbin/nologin by “chsh –s /sbin/nologin <user>”
  + If creating user, home dir should be /var/www
  + Make sure the user cannot access anything else
  + The apache user should **not** have write access to /var/www (but it needs read access)
  + run ‘find / -user apache –not –path “/proc/\*”’ to find directories owned by the apache user
* Install mod\_security, enable, restart apache
  + (CentOS, RedHat) Pull the RBEL library by “yum install epel-release”
  + (CentOS, RedHat) Install mod\_security by running “yum install mod\_security”
  + (Debian/Ubuntu) Install mod\_security by running “apt-get install libapache2-mod-security”. May also be named libapache-mod-security or similar.
    - Guaranteed to be in repo “old-releases.ubuntu.com/ubuntu karmic universe” and later.
  + Default mod\_security rules are located in /usr/share/modsecurity-crs (possibly in /usr/share/doc/modsecurity/examples/rules)
    - add “Include /path/to/rules/\*.conf” to main apache config file to activate
  + (CentOS, RedHat) Automatically activate the base rule set by running “yum install mod\_security\_crs”
  + Find modsecurity.conf (in /etc/httpd/conf or /etc/modsecurity usually) and make sure it contains the line “SecRuleEngine On”. Change the SecRuleEngine line to that if it is not. If you can’t find the file, find it with the core rulesets above, and move it to a place where conf files for modules are read (e.g. mods-enabled).
* Install mod\_evasive, configure, and restart apache
  + (CentOS, RedHat) “yum install mod\_evasive” sets it up with a default ruleset
  + (Debian, Ubuntu) “apt-get install apache2-mod-evasive”
  + If it doesn’t include a config file, here’s a default config to be put in main config file
    - <IfModule evasive20\_module>
    - DOSHashTableSize 3097
    - DOSPageCount 2
    - DOSSiteCount 50
    - DOSPageInterval 1
    - DOSSiteInterval 1
    - DOSBlockingPeriod 60
    - DOSEmailNotify <someone@somewhere.com>
    - </IfModule>

(The following modify httpd.conf or apache2.conf, depending on configuration)

* Investigate which Apache modules are installed, disable ones not being used
  + **mod\_imap**, **mod\_include**, **mod\_info**, **mod\_status,** **mod\_userdir**, and **mod\_autoindex are modules that are usually not needed**
  + **Disable a module by commenting out its load line in the config file**
* Disable Apache’s ServerSignature
  + Change “ServerSignature On” to “ServerSignature Off”
  + Change the “ServerTokens” line to “ServerTokens Prod”
* Disable directory listing on accessible directories
  + Wherever there is a directory entry in httpd.conf, add “-Indexes” to the Options line, or add a dash in front of the Indexes entry if it already has one.
  + Add “Options –FollowSymLinks” to disable the following of symlinks.
* Secure the root directory
  + The root directory’s directory config should look like this:
    - <Directory / >
    - Options None
    - AllowOverride None
    - Order allow,deny
    - </Directory>
* Chroot Apache if you have time

**PHP**

* The php config file is usually at /etc/php.ini or at /etc/php5/… or some such
* Install suhosin
  + Attempt to install with “apt-get install php-suhosin”
    - If this fails, add the following to sources.list:
      * deb http://repo.suhosin.org/ ubuntu-trusty main
    - For this repo, install as “apt-get install php-suhosin-extension”
* Make changes to the config file
  + If the file includes “expose\_php = On”, change to “Off”, add “expose\_php = Off” if not present
  + Similarly, “display\_errors=Off” is important and “log\_errors=On”. Make sure “error\_log=/path/to/log” points somewhere useful
  + allow\_url\_fopen and allow\_url\_include should both be “Off”
  + Unless handling large POSTs, reduce post\_max\_size to something small, like 16K.
  + If hosting out of only one directory (e.g. /var/www/), set “open\_basedir = /var/www” to limit php file access to only that directory
  + Restart apache to make sure the changes propagated

MySQL

* Run mysql under its own user
  + set shell to /sbin/nologin by “chsh –s /sbin/nologin <user>”
  + set owner for /var/www (or wherever is being served)
  + make sure the user cannot access anything else
  + run ‘find / -user mysql –not –path “/var/lib/mysql\*” –not –path “/proc/\*”’ to find directories owned by the mysql user
* Run mysql\_secure\_installation, which will walk through basic security config, like changing the root password and deleting unnecessary databases.
* Use the “mysqldump –u root –p password > mysql.bak” command to back up the database periodically
* Change config options in in /etc/my.cnf or /etc/mysql/my.cnf. Modifications are made under the [mysqld] header.
  + Make sure “skip-grant-tables” is **not** present
  + Add “safe-user-create=TRUE” if not there already
  + Add “skip-show-database” if not present
  + Add “skip-networking” to disable network listening and only connect using a UNIX socket. The socket is normally at /tmp/mysql.sock, but can be changes with the socket=/path/to/sock directive in both the client and server config files.
* Secure user permissions
  + User permissions are in the mysql.users table
  + Users are identified by “user@host” notation. Discover which user is the root user and which are being used by various applications
  + Delete any users not in use by any application. “DROP USER user@host” deletes a user
  + If any application is connecting as root, make a new user for that application (you broke it anyways when you changed the root password, so might want to do this first).
  + If a user has a wildcarded or inappropriate host name, change the host to be only the host that the user would log in from (or an expression for the hosts it would login from). Use IP addresses where possible. The root user should only be root@localhost.
  + You can control a user’s max concurrent connections with the “max\_user\_connections” value in the users table. (0 is unlimited)
  + Global privileges are stored in the users table as Boolean values (or enum(‘N’, Y’) for mariaDB). “File\_priv”, “Drop\_priv”, “Grant\_priv”, “Super\_priv”, “Show\_db\_priv”, and “Create\_user\_priv” should not be held by anyone besides root. In general, normal users should not have any global privs at all.
    - Remove privs by “Update mysql.users SET File\_priv = ‘N’, Drop\_priv = ‘N’, Grant\_priv = ‘N’, Super\_priv = ‘N’, Show\_db\_priv = ‘N’, Create\_user\_priv=’N’ WHERE User <> ‘root’;
  + For each user, enter “SHOW GRANTS for user@host” to show the grants on each table for a user.
  + Revoke all grants that are not INSERT, UPDATE, SELECT, or SHOW\_VIEW (possibly keep DELETE too or EXECUTE if there are stored routines). Grants should be as narrow as possible (ideally per-table, but per-database is good too). Revoke grants with “REVOKE priv1, priv2,… ON table1, table2 FROM db1.table1, \*.\*, …;”.
  + To revoke everything for a user, use “REVOKE ALL PRIVELEGES, GRANT OPTION FROM user1, user2,…”
  + If you created new users for applications, GRANT them relevant privs using “GRANT INSERT, UPDATE, SELECT ON db1.table1, \*.\*,… TO user1, user2,…;”

**Postgres**

* Switch to the “postgres” (or other admin user) on Linux to access the database
  + Immediately change its password to something else
* List all users by typing “select \* from pg\_shadow;” into the postgres console
  + Delete all unnecessary users and find the user(s) that your applications are using
  + If you need to create a user use “CREATE ROLE user1 WITH LOGIN ENCRYPTED PASSWORD mypassword”
    - Name the new role the same as the system user that the application uses
  + Always revoke public schema privs with “REVOKE ALL PRIVILEGES ON SCHEMA PUBLIC FROM user1;”
  + Use the “\dp” command in postgres to view tables and priveleges
  + As in MySQL, REVOKE all privs that are not select, update, or insert
  + Other user config options are as in MySQL
* Secure access permissions in pg\_hba.conf
  + Entries are of the form “local DATABASE USER METHOD [OPTION] ” or “host DATABASE USER IP-ADDRESS/CIDR-MASK METHOD [OPTION]”
    - The local option is for a UNIX socket, and the other is for general hosts (including 0.0.0.0
    - The postgres user should only ever be allowed to connect through local to all dbs
    - METHOD should always be “md5” or whatever the authentication is, except for postgres, which can be “trust”
  + Limit host access only to those hosts that need it. If all connections are through UNIX sockets, only use the “local” entry

**IIS**

* Download Microsoft’s IIS lockdown if you’re running a version of IIS before 7.0. You might want to do it anyways afterwards, but do this first if this is the case.
  + If prompted, also configure URLScan if using an old version
* Enable dynamic IP restrictions
  + Make sure the “IP Security” or “IP and Domain Restrictions” module is installed
  + Go to “IP Address and Domain Restrictions” from the main machine page
  + Click “Edit Dynamic Restriction Settings” on the right
  + Adjust settings to liking (the defaults are okay as long as they are enabled).
* Under “Application Pools”, make sure each pool runs under “Application Pool Identity”
* Set up request filtering
  + Double-click on “Request Filtering” on the main machine page
  + Click the “Edit Feature Settings” button on the right
    - Make sure high-bit characters and double-escaping are not allowed
    - Make sure other settings are sane
  + Click on the “rules” tab and click “add new filtering rule”
    - Name it “SQL\_Injection” or something
    - Scan both URLs and queries
    - Apply to .asp and .aspx files
    - Filter the keywords “select”, “convert”, “update”, and “union”
  + Click on the “URL” tab
    - Deny the sequences “./” and “..”
  + Click on the “HTTP Verbs” tab
    - Add a new rule that disallows the “OPTIONS” verb
* Enable logging
  + Double-click on “Logging” on the main machine page
  + Make sure that logging is enabled
  + Click on the “View Log Files” button on the right-hand side
    - Determine the currently-active log file
    - Open a shell window and run “type –wait <logfile>” and keep the command window open to monitor the log file

**Windows**

* Run the Security Configuration Wizard
  + Turn off all roles and features which aren’t expressly needed
    - Any remote service is bad (unless you need it for something
    - There should probably be practically no enabled options
  + Choose to disable all services not specified
  + Probably keep the auto-detected firewall settings
    - Disable anything super-sketchy looking though
  + Set registry policies
    - Only check domain accounts when asked how to connect to other computers
    - Disallow as much inbound connection as possible
  + Change audit policy to audit successful and unsuccessful activities
  + Name the policy “CCDC” or something and apply it to the server immediately
* Open Event Viewer and look at the “Security” log to see security audit logs
* If have time, look at Windows Security Compliance Manager