28 Steps on how to harden your Linux server

If you run your own Linux server here are some tips on server hardening, liberally stolen from the CFS security GUI script for cPanel/WHM.

If you run your own Linux server here are some tips on server hardening, liberally stolen from the CFS security GUI script for cPanel/WHM, that I have become only too familiar with since yesterday:

1. On your firewall (you do have one don't you?) check the incoming MySQL port and if 3306 is open, close it. If this port is left open it can pose both a security and server abuse threat since not only can hackers attempt to break into MySQL, any user can host their SQL database on your server and access it from another host and so (ab)use your server resources
2. Check /tmp permissions. /tmp should be chmod 1777
3. Check /tmp ownership /tmp should be owned by root:root
4. Check /etc/cron.daily/logrotate for /tmp noexec workaround. Due to a bug in logrotate if /tmp is mounted with the noexec option, you need to have logrotate use a different temporary directory. If you don't do this syslog may not restart correctly and will write to the wrong (older) log files. See here for a way to do this
5. Check /var/tmp permissions. /var/tmp should be chmod 1777
6. Check /var/tmp ownership. /var/tmp should be owned by root:root
7. Check /var/tmp is mounted as a filesystem. /var/tmp should either be symlinked to /tmp or mounted as a filesystem
8. Check /var/tmp is mounted noexec,nosuid. /var/tmp isn't mounted with the noexec,nosuid options (currently: none). You should consider adding a mountpoint into /etc/fstab for /var/tmp with those options
9. Check /usr/tmp permissions. /usr/tmp should be chmod 1777
10. Check /usr/tmp ownership. /usr/tmp should be owned by root:root
11. Check /usr/tmp is mounted as a filesystem or is a symlink to /tmp. /usr/tmp should either be symlinked to /tmp or mounted as a filesystem

    Check /etc/resolv.conf for localhost entry. You should not specify 127.0.0.1 or localhost as a nameserver in /etc/resolv.conf - use the servers main IP address instead

12. Check /etc/named.conf for recursion restrictions. If you have a local DNS server running but do not have any recursion restrictions set in /etc/named.conf this is a security and performance risk and you should look at restricting recursive lookups to the local IP addresses only. Unrestricted recursive lookups are as good as a DDoS attack against your system. They will eat up all your system resources
13. Check server runlevel. For a secure server environment you should only run the server at runlevel 3. You can fix this by editing /etc/inittab and changing the initdefault line to:
    id:3:initdefault: and then rebooting the server
14. Check nobody cron. You have a nobody cron log file - you should check that this has not been created by an exploit
15. Check Operating System support. Make certain that your OS version is still supported by the manufacturer and that upgrades continue to be available
16. Check SSHv1 is disabled. You should disable SSHv1 by editing /etc/ssh/sshd_config and setting: Protocol 2 (remove the hash # from in front of the line and edit out the 1.1)
17. Check SSH on non-standard port. Moving SSH to a non-standard port avoids basic SSH port scans. Edit /etc/ssh/sshd_config and setting: Port nnnn Where nnnn is a port of your choosing. **Don't forget to open the port in the firewall first!**
18. Check SSH PasswordAuthentication. For ultimate SSH security, you might want to consider disabling PasswordAuthentication and only allow access using PubkeyAuthentication. For more information read this article and this article
19. Check telnet port 23 is not in use. Close this port in your firewall. Telnet is an insecure protocol and you should disable the telnet daemon if it is running

20. Check shell resource limits. You should enable shell resource limits to prevent shell users from consuming server resources - DOS exploits typically do this. If you are using cPanel/WHM set Shell Fork Bomb Protection.
21. Disable all instances of IRC - BitchX, bnc, eggdrop, generic-sniffers, guardservices, ircd, psyBNC, ptlink. If you are using WHM you can do this in the Background Process Killer.
22. Check apache for mod_security if not installed install it from source
23. Check apache for mod_evasive. You should install the mod_evasive apache module from source to help prevent DOS attacks against apache. Note that this module breaks FrontPage functionality
24. Check apache for RLimitCPU. You should set a value RLimitCPU to prevent runaway scripts from consuming server resources - DOS exploits can typically do this.
25. Check apache for RLimitMEM. You should set a value RLimitMEM to prevent runaway scripts from consuming server resources - DOS exploits can typically do this
26. Check php for enable_dl. You should modify /usr/local/lib/php.ini and set:
    enable_dl = off This prevents users from loading php modules that affect everyone on the server. Note that if use dynamic libraries, such as ioncube, you will have to load them directly in php.ini1
27. Check php for disable_functions. You should modify /usr/local/lib/php.ini and disable commonly abused php functions, e.g.:
    disable_functions = show_source, system, shell_exec, passthru, exec, phpinfo, popen, proc_open Some client web scripts may break with some of these functions disabled, so you may have to remove them from this list
28. Check phpsuexec. To reduce the risk of hackers accessing all sites on the server from a compromised PHP web script, you should enable phpsuexec when you build apache/php. Note that there are side effects when enabling phpsuexec on a server and you should be aware of these before enabling it

Reprint with permission of the author