

Bastille Linux: A Walkthrough

Jay Beale 2000-06-06

You use a "hardening program" to try to make your system as secure as possible, from the ground up. Generally, you deactivate unnecessary services and better the configurations of the ones you leave enabled. This is wildly effective, as it can eliminate many of the vulnerabilities that are common on Linux/Unix platforms. This article presents a walkthrough of Bastille Linux, a popular hardening program for Red Hat and Mandrake, available for free from Jon Lasser, Pete Watkins, myself, and the rest of the Bastille Linux project. This walkthrough won't be the kind of "paranoid" setup that I enjoy most, as that could remove too much functionality for the average reader. Don't worry - I'll explain what we'll break in each setting, how we'll break it, and how you can fix it. But first, a shameless plug: I'll let you know about the cool features in the newest Bastille version, which we've just released.

Cool Features in Bastille Linux 1.1:

- Bastille now runs on Non-Virgin systems!

Bastille is now much smarter, allowing it to run on systems that have been in use for a while. It uses pattern matching - based API routines and has a whole lot more of them. It also assumes a lot less about the state of the target machine.

- Bastille can now be run multiple times!

Again, Bastille is much more intelligent. It shouldn't make improvements that are already in place, whether they were made by you or by the program on a previous run.

- Undo functionality implemented!

Every single file change is backed up in an undo directory, with the directory structure preserved and permissions left alone. You can back out individual changes made by Bastille or simply back out everything altogether!

- Impotent (log only) mode added!

While the logging is a little rough, Bastille can log every action it would have taken on your system without actually taking those actions!

- Non - Red Hat / Mandrake support is much fuller now.

Bastille can now be more easily extended to other distributions of Linux, as we've removed most Red Hat specific content. Most of our modules will run on any distribution, as soon as we code the binary locations into our `ConfigureForDistro()` subroutine.

With all that said, please be assured that development is continuing. We're implementing even more cool functionality and we're all quite sure that Bastille is, and will continue to be, one of the most full-featured hardening programs out! Got a suggestion? E-mail me at jay@bastille-linux.org. With that said, let's get to the walkthrough!

Starting The Walkthrough:

At the time of this writing, Bastille 1.1.0 has been released. It might be a little rough around the edges, but if something doesn't work as planned, it can be easily undone or we can release a quick fix. I'm recommending this version specifically because the new architecture is much more featureful. Let's download, install and run Bastille on an x86-based Red Hat 6.x box. First, switch to console mode. While Bastille uses a GUI-like (curses-based) Text User Interface, it's still a console tool that runs best on a standard size 80x25 screen.

```
( Hit -- and log in as root )
```

```
# cd /root
```

```
# wget http://www.sourceforge.net/download.php/bastille-linux/Bastille-1.1.0.tgz
```

```
# tar -xzf Bastille-1.1.0.tgz
```

```
# cd /root/Bastille
```

```
# ./InteractiveBastille
```

Bastille starts with the four most powerful steps you can take to secure a Linux/Unix box:

- Apply a firewall (packet filter) to prevent access to possibly vulnerable services

- Apply system patches for all known security holes
- Perform a SUID - root Audit
- Deactivate or restrict unnecessary services

Let's step through these...

(Module: IPCHAINS)

Bastille starts by offering to configure a firewall. While this is not all that is required to secure a machine, firewalls can be great start. They function to block network traffic from hitting your machine's network daemons. Why not just secure the daemons or deactivate them? Actually, we'll take all three of these actions. Why not just do one or the other? The answer is a basic tenet of operating system security: Defense in Depth.

Defense In Depth

Protect each service or possible vulnerability through multiple means, so that if one fails, the remaining methods keep your machine from being compromised.

To keep this article from getting too deep too fast, we'll skip firewalling for now. Bastille has very good explanations in the IPCHAINS module - further, the default answers should be fairly safe. Unfortunately, this module is long enough to necessitate its own walkthrough. Look for a future article, Real Soon Now on this topic.

Q: Would you like to run the ipchains script? [N]

A: N

(Module: Patch Download)

Bastille now tells us about downloading new patches. This is VERY important. Bastille can

minimize damage from vulnerable services through preventive measures, but it really is most effective to patch the security holes in the first place! Most operating systems (save OpenBSD) seem to release with vulnerable programs, some of which can be abused to gain root on your system. If you're still not patching, please remember the remote vulnerability present in Red Hat 6.0's BIND version... This step, however, is best done manually...

Q: Would you like to download and install updated rpms?

A: N

(Module: File Permissions)

Now we'll audit file permissions. If you're an advanced user, you may use the general permissions audit. This audit, adapted from a SANS document, restricts permissions on many binaries. Most people should choose N here.

Q: Would you like to set more restrictive permissions on the administrative utilities?

[N]

A: N

Now we get to that third "most powerful action" quoted from above: the SUID Audit. Bastille lets us strip the SUID bit on about half of the SUID programs on a Red Hat 6.x or Mandrake system. Since this is the non-paranoid run, we won't disable all of them...

We'll leave the SUID bit on mount/umount, so that ordinary users can still mount floppies and cdroms.

Q: Would you like to disable suid status for mount/umount?

A: N

We'll also leave ping and traceroute active, so ordinary users can test network connectivity. These are really not needed, but I am attempting to be less paranoid, in the name of user

convenience.

Q: Would you like to disable suid status for ping?

A: N

We'll strip the SUID bit from dump and restore, which are used for system backups. Root can take care of system backups.

Q: Would you like to disable SUID status for dump and restore?

A: Y

We'll also rip SUID status off of cardctl, which is used to configure PCMCIA services - you can answer N here if you're running a notebook.

Q: Would you like to disable SUID status for cardctl?

A: Y

We'll also rip SUID status off of at, because it has had a rich history of security problems. You can achieve all the same functionality with cron, so this is safe.

Q: Would you like to disable SUID status for at?

A: Y

We can safely tear SUID off of dos (dosemu). Dosemu is a DOS emulator for Linux and, as such, is very open to manipulation by the user. It also is used by few users - feel free to re-enable suid if you're planning on making this available to many users.

Q: Would you like to disable SUID status for DOSEMU?

A: Y

It is also very safe to remove the SUID bit from the news (UseNet) server control utilities, `inndstart` and `startinnfeed`. Most people reading this article aren't running a news server on their machines... Even if they were, only root should be starting the news server! Ordinary users can read news without needing to be able to start a news server, especially since most people read UseNet via their ISP's news server.

Q: Would you like to disable SUID status for news server tools?

A: Y

We'll leave the SUID bit on the printing utilities, though you should disable these if your machine won't be used for printing.

Q: Would you like to disable SUID status for printing utilities?

A: N

Now, we get to one of my favorite topics, the Berkeley r-tools, like `rcp`, `rlogin` and `rsh`. These harbingers of doom use IP addresses for authentication, which unfortunately is not a suitable method of authentication. Trust me here, turn these the heck off!

Q: Would you like to disable SUID status for the r-tools?

A: Y

We'll slip back to permissive mode now, and leave SUID on `usernetctl`. This utility is used by ordinary users to activate and deactivate network interfaces, like their dial-up modems. Feel free to leave this on...

Q: Would you like to disable SUID status for usernetctl?

A: N

Q: Would you like to disable SUID status for traceroute?A: N

Now, I promise there aren't so many questions in most of the other modules!

Module: Account Security

Bastille now moves to Account Security, which is not the last of the four "most powerful steps" that I outlined above. Don't panic! We'll come back to that one... Often a system cracker starts by stealing one of your user's accounts or by compromising a non-user account. The AccountSecurity module enforces some Best Practices and creates a few tricks, as we'll show below.

We'll create a second superuser account. You should use this for root activities and leave the root account untouched.

Q: Would you like to set up a second UID 0 account?

A: Y

Bastille will now request permission to disable the Berkeley r-tools: rsh, rlogin, rcp by making PAM modifications, commenting out lines in inetd.conf, and removing all permissions on these binaries.

Q: May we take strong steps to disallow the dangerous r-protocols? [Y]

A: Y

We'll now elect to enforce password aging, to freeze unused accounts before they can be used to compromise your system.

Q: Would you like to enforce password aging? [Y]

A: Y

If you haven't created an ordinary user account, you should let Bastille do this for you. Otherwise, skip it like so:

Q: Would you like to create a non-root user account? [N]

A: N

We'll restrict most users from using cron - it represents a double risk: first, it runs as root and might be open to compromise. Second, an attacker can use cron to hide commands and run them outside of the interactive sessions that you'd be able to monitor easily.

Q: Would you like to restrict the use of cron to administrative accounts? [Y]

A: Y

We'll also go ahead, in the future, and assign a restricted/useless shell, like `/bin/false`, to all non-user accounts.

Module: Boot Security

Interestingly enough, in Red Hat 6.0, anyone who can get to a LILO prompt can have root on your system! Try this at the LILO prompt: type "linux single". You'll find yourself with a nice root shell! If we deactivate that, you can still type at the LILO prompt: `linux init=/bin/bash`. We can do something about this.

Q: Would you like to password-protect the LILO prompt? [N]

A: Y

Q: Enter LILO password, please. []

A: *put-in-your-own-password-please-or-you've-done-something really-dumb*

This solution actually prevents both of the exploits above, as it protects the LILO prompt. We can protect this prompt further by not giving an attacker any chance to even try a password...

Q: Would you like to reduce the LILO delay time to zero? [N]

A: N

We can write this lilo configuration to our hard drive or to a pre-made boot floppy, or both. Supposing that we boot from your hard drive only:

Q: Do you ever boot Linux from the hard drive? [Y]

A: Y

Q: Would you like to write the LILO changes to a boot floppy? [N]

A: N

We could disable CTRL-ALT-DEL rebooting, but this doesn't always make sense.

Q: Would you like to disable CTRL-ALT-DELETE rebooting? [N]

A: N

We can take one final step to prevent the "LILO: linux single" root-grab. Remember: Defense in Depth!

Q: Would you like to password protect single-user mode? [Y]

A: Y

Module: SecureInetd

OK, back to the fourth of the "four most powerful steps." Our last "powerful" step, restricting and deactivating unnecessary services, follows from another basic tenet of system security: Minimalism.

(Applied) Minimalism

Since crackers may discover an exploitable vulnerability in any service running with privilege, minimize both the number of these services and their levels of privilege.

This principal will save your butt over and over. A Red Hat 6.0 box running with "Everything" installed and every service active can be rooted remotely rather easily through, among other things, its Name Server. If you take the five minutes to turn this off, you win! If you had to leave it on, but you lower its level of privilege by setting it to run as an ordinary user, you win again. Otherwise, you can be rooted by the least experienced script kiddie.

We'll start the process of restricting unnecessary services by changing */etc/inetd.conf* and the TCP Wrappers */etc/hosts.allow* file. We'll disable telnet, ftp, pop, imap, rsh, rlogin, and talk. Pop and imap are mail protocols that should be disallowed unless you really do use them. Telnet, rlogin and rsh are all horribly insecure and can be replaced by ssh. Often, rcp and ftp can be replaced by the much safer scp. We remove talk simply because we don't need it.

Q: Would you like to modify inetd.conf and /etc/hosts.allow to optimize use of Wrappers? [Y]

A: Y

You can turn these back on, or allow them from required systems, by editing the `/etc/hosts.allow` file. Try: `man hosts_access`. Moving on, we'll choose to leave ssh open to the entire Internet, for now.

Q: Would you like to set sshd to accept connections only from a small list of IP addresses. [N]

A: N

Finally, we'll choose to add "Authorized Use Only" banners to the system. These are often required to successfully prosecute system crackers.

Q: Would you like to make "Authorized Use" banners? [Y]

A: Y

Module: DisableUserTools

Following the principle of Minimalism, we can now set permissions to only allow root to use the compiler. If you need users to run this, please **don't** choose Y here.

Q: Would you like to disable the compiler? [N]

A: Y

Module: ConfigureMiscPAM

Sidetracking from the "minimalism" stuff, we make some modifications that should make it difficult for any one user, including the "nobody", "web" or "ftp" users, to abuse system resources to cause a Denial of Service (DoS) attack.

Q: Would you like to put limits on system resource usage? [Y]

A: Y

Q: Should we restrict console access to a small group of user accounts? [N]

A: N

Module: Logging

We'll add some additional logging to the system, creating special logs for kernel messages and severe error messages and, furthermore, logging important information to two virtual TTY's. After we're done, you'll use `-` and `-` to view these and `-` to get back to an X server.

Q: Would you like to add additional logging? [Y]

A: Y

Unless you have a remote logging host, we'll answer the next question like so:

Q: Do you have a remote logging host? [N]

A: N

Process accounting logs *every* process as it starts. If you've got the disk space and CPU time to spare, or are very paranoid, this can be very useful. Most people shouldn't touch this.

Q: Would you like to set up process accounting? [N]

A: N

Module: MiscellaneousDaemons

We now get directly back to our Minimalism-motivated process. We'll turn off every system daemon that you don't need. In this walkthrough, we turn most everything off. You can turn it back on with the command `chkconfig on`.

Q: Would you like to disable apmd? [Y]

A: Y

Q: Would you like to deactivate NFS and Samba? [Y]

A: Y

Q: Would you like to disable atd? [Y]

A: Y

Q: Would you like to disable PCMCIA services? [Y]

A: Y

Q: Would you like to disable the DHCP daemon? [Y]

A: Y

Q: Would you like to disable GPM? [Y]

A: Y

Q: Would you like to disable the news server daemon? [Y]

A: Y

Q: Would you like to deactivate the routing daemons? [Y]

A: Y

Q: Would you like to deactivate NIS server and client programs? [Y]

A: Y

Q: *Would you like to disable SNMPD? [Y]*

A: Y

Module: Sendmail

Continuing to follow the principle of Minimalism, we can now deactivate sendmail's "listen mode." It will still be available to send mail off the system, but won't receive any from the network. If you need to receive mail via sendmail, choose Y:

Q: *Do you want to leave sendmail running in daemon mode? [Y]*

A: N

To disable automated sendmail altogether, we'd choose N for the next question.

Q: *Would you like to run sendmail via cron to process the queue? [N]*

A: Y

Finally, we'll disable some sendmail commands that are commonly used to gain information about your system for cracking or spamming.

Q: *Would you like to disable the VRFY and EXPN sendmail commands? [Y]*

A: Y

Module: RemoteAccess

You should install Secure Shell (ssh) on every system that needs remote access, though you may want to do it manually.

Q: Would you like to download and install ssh? [N]

A: Y

Module: DNS

You might notice a strange practice in the remaining modules: we tighten services that we turn off. We recommend you do this because you never know when you might have to turn them back on. Paranoid, yes, but from experience, a sound practice.

We secure DNS initially by running it as an ordinary user and restricting (chroot-ing) it to a small subset of the filesystem. This turns the recent BIND (DNS) exploit from a root-grab into DNS server Denial of Service (DoS). Actually, we (and the guys at SANS) implemented and suggested this months before the exploit was found.

Q: Would you like to chroot named and set it to run as a non-root user? [N]

A: Y

Q: Would you like to deactivate named, at least for now? [Y]

A: Y

You'll have to get used to two small changes in the way you admin your DNS server when you do this - read the Bastille explanations carefully.

Module: Apache

Unless you'll be using your web server immediately, we'll turn it off for now. Reactivate it later with *chkconfig httpd on*.

Q: Would you like to deactivate the Apache web server? [Y]

A: Y

If you only need the webserver to test web pages that you're working on locally, we can bind it to a your local interface. We could also bind it to only one interface (like your ethernet card, but not your PPP link.) I assume here that your web server must be viewable by the entire internet.

Q: Would you like to bind the web server to listen only to the localhost? [N]

A: N

Q: Would you like to bind the web server to a particular interface? [N]

A: N

We have more choices than just deactivating Apache. While Apache has some nice features, we'd prefer to disable features we aren't directly using. Apache can follow symbolic links, so that if one of your users makes a link from his web directory to /, Apache can show any user-viewable file to the entire Internet! We'll turn this off:

Q: Would you like to deactivate the following of symbolic links? [Y]

A: Y

Server side includes aren't used by most casual users. They can be rather dangerous.

Q: Would you like to deactivate server-side includes? [Y]

A: Y

CGI scripts incredibly useful on servers - simultaneously, badly written CGI scripts represent one of the most common methods of system compromise today. Decide carefully: will you be writing, downloading or buying CGI scripts? Many conscious sites run them on only some

servers and then only after auditing each CGI script for problems.

Q: Would you like to disable CGI scripts, at least for now? [Y]

A: Y

Lastly, we think about indices. In the absence of a index.html file, Apache will list all files in the current directory. These aren't as bad as symbolic links, since they don't enable an attacker to see files outside the web directories.

Q: Would you like to disable indexes? [N]

A: N

Module: Printing

If your box isn't being used to print, we can disable the printing daemon and strip SUID from lpr and lprm, like so:

Q: Would you like to disable printing? [N]

A: Y

Module: FTP

Last module! Anybody tired yet? We now can prune back the access of the FTP daemon, wu-ftpd. From a security perspective, FTP is a very problematic protocol. Further, wu-ftpd has had a number of security alerts lately. Most security-conscious people try to avoid running an ftpd like the plague and you might want to follow their example. Even if you can, let's prune this daemon.

Of all possible configurations, the worst is one that allows anonymous upload. Luckily, this is not the default configuration! Still, Red Hat's default configuration allows user/password access as well as anonymous download. You can deactivate one or both of these. Here, we deactivate

only user ftp access:

Q: Would you like to disable user privileges on the FTP daemon? [N]

A: Y

Q: Would you like to disable anonymous download? [N]

A: N

Wrap-Up:

OK, that's it. We've made our choices in the front end, which should now create a configuration file. Let's implement these choices like so:

(Exit from the Credits screen by pressing Tab)

```
# ./BackEnd.pl
```

Your changes should now be implemented. You'll find your machine to be slightly less functional, but only in ways that you chose. You'll find backup copies of every configuration file changed in `/root/Bastille/undo/backup`. Reboot your machine and notice the changes. An attacker will find far fewer avenues of attack against your machine, depending on what options you picked.

*Jay Beale is the Lead Developer of the Bastille Linux Project (<http://www.bastille-linux.org>). He is the author of several articles on Linux/Un*x security, along with the upcoming book "Securing Linux the Bastille Way," to be published by Addison Wesley. At his day job, Jay is a security admin working on Solaris and Linux boxes.*

Relevant Links

[FOCUS-Linux Mail List](#)

SecurityFocus

[Installing Linux](#)

Peter Merrick

[Securing Linux](#)

Dale Coddington

[Securing Linux Pt II](#)

Dale Coddington

[Linux and IPsec](#)

Rafael Coninck Teigao

[Linux Security Tools](#)

Jonathan Day

[Building a Linux Bunker: Basic Firewalling](#)

Rafael Coninck Teigao

[Intrusion Detection on Linux](#)

David "Del" Elson

[Bastille Linux Walkthrough](#)

Jay Beale

[Snort Installation and Basic Usage](#)

Dale Coddington

[Snort Installation and Basic Usage Pt. II](#)

Dale Coddington

[Privacy Statement](#)

Copyright 2006, SecurityFocus