

Gentoo Hardened SELinux Testing

COLLABORATORS

| | | | |
|---------------|-------------------------------------------------------|-------------------|------------------|
| | <i>TITLE :</i> Gentoo Hardened SELinux Testing | | |
| <i>ACTION</i> | <i>NAME</i> | <i>DATE</i> | <i>SIGNATURE</i> |
| WRITTEN BY | Sven Vermeulen | February 13, 2011 | |

REVISION HISTORY

| | | | |
|---------------|-------------|--------------------|-------------|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

Contents

| | | |
|----------|---------------------------------------------|----------|
| 1 | Introduction | 1 |
| 1.1 | Testing Overview | 1 |
| 1.2 | Testing Architecture | 1 |
| 2 | Stage3 Hardened Installation | 1 |
| 2.1 | Steps | 2 |
| 2.2 | Deviations | 3 |
| 3 | Appendix: Facilitating Using BINHOST | 3 |
| 3.1 | Feeding the BINHOST | 3 |

1 Introduction

As an active contributor to the Gentoo Hardened project with regard to SELinux support (both documentation, policy development and ebuilds) I find it important to have a good testing approach. Thanks to virtualization efforts such as KVM, it is possible to generate a semi-automated testing environment, allowing me to test various setups and installations before considering contributions stable enough.

Of course, why using semi-automated testing rather than fully automated? Well, I'm probably going to automate more and more to the point it is fully automated, but for the time being, it is a semi-automated approach ;-)

1.1 Testing Overview

The following table shows the tests that I execute. Most tests are based on the documentation offered by Gentoo, which also means that I occasionally have to review the scripts to see if they still follow the instructions (as documentation changes over time).

| Test Name | Test Description |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Stage3 hardened installation | Perform a blank hardened installation, using the latest stage3-hardened-amd64 snapshot (in the autobuilds directory). Follow the instructions of the Gentoo Handbook closely. |
| SELinux conversion | Convert the hardened installation to a SELinux, following the instructions in the Gentoo Hardened SELinux handbook. Enable SELinux enforcing/strict mode and perform a few standard tests. |
| MySQL installation and testing | Install MySQL following the MySQL guide and perform a few standard tests. |

1.2 Testing Architecture

To enable semi-automated testing, I run a simple webserver which provides read access to:

- the host distfiles (URL: /gensetup/gentoo/distfiles)
- the stage3 tarball, portage snapshot and Linux kernel binary package (URL: /gensetup/gentoo)
- various binhosts for different installation types (URL: /gensetup/gentoo/binhost)

The tests all run inside KVM guests (with network access). The idea is as follows:

1. Perform a standard stage3 installation, call that "hardened-base".
2. Make a copy-on-write from "hardened-base" and use that one to further test hardened setups, call that "hardened"
3. Make a copy-on-write from "hardened-base" and use that one to test the SELinux installation instructions, call that "selinux-base"
4. Make a copy-on-write from "selinux-base" and use that one to further test various SELinux setups, call them "selinux"

2 Stage3 Hardened Installation

This test performs a hardened installation in a virtual environment. At the end, we should have a bootable virtual environment with a small but functional hardened system (with pax/grsecurity hardening active) inside.

2.1 Steps

The stage3 hardened installation uses a 20Gbyte qemu image (qcow2 format):

```
~$ qemu-img create -f qcow2 test.img 20G
```

This image is booted using the system rescue CD ISO:

```
~$ qemu-system-x86_64 --enable-kvm -net nic,model=virtio,macaddr=00:11:22:33:44:a0,vlan=0 \
-net vde,vlan=0 -drive file=test.img,if=virtio,boot=on -usb -usbdevice tablet -smp 2 \
-k nl-be -m 1024 -boot d -cdrom systemrescuecd-x86-2.0.0.iso
```

The virtual drive (vda) is formatted with a root partition (vda1, ext4), a swap partition and a physical group (vda3) for LVM2. A volume group is then created with usr, home, opt and var as logical volumes.

```
~# df -hT
Filesystem      Type      Size  Used Avail Use% Mounted on
/dev/vda1       ext4      2.0G  424M  1.5G  23% /
udev           tmpfs     10M   104K   9.9M   2% /dev
shm            tmpfs     498M    0   498M   0% /dev/shm
/dev/mapper/vg-usr  ext4      9.9G  2.2G   7.3G  23% /usr
/dev/mapper/vg-home ext4      2.0G   67M   1.9G   4% /home
/dev/mapper/vg-opt  ext4      2.5G   68M   2.3G   3% /opt
/dev/mapper/vg-var  ext4      2.5G  128M   2.3G   6% /var
tmpfs          tmpfs     498M    0   498M   0% /tmp
```

During the Gentoo installation, the following settings are used.

| Setting | Description |
|----------------------------------------------------|----------------------------------------------------------------------------------------------|
| Profile = hardened/linux/amd64 | The Portage profile selected during the installation is the "hardened/linux/amd64" profile. |
| USE = -ldap -gtk -xorg -ipv6 -pppd | The USE flags are modified slightly to keep the number of package dependencies to a minimum. |
| clock/TIMEZONE = Europe/Brussels | The timezone is set to Europe/Brussels |
| keymaps/KEYMAP = be-latin1 | The keymap is set to be-latin1 |
| net/config_eth0 = ("dhcp") | The network configuration uses DHCP |
| /etc/locale.gen only en_US (ISO-8859-15 and UTF-8) | The number of locales to be generated by glibc installation is limited to the en_US ones. |

Also, the following packages are installed:

- mdadm (as I hope to extend the installation later to use two virtual disks in software RAID-1 configuration)
- lvm2 (as the installation uses LVM2)
- syslog-ng (system logger), this one is also added to the default runlevel. During the installation, "**unset path**" is executed (otherwise the installation of syslog-ng fails).
- dhcpcd (DHCP client)
- grub (bootloader)
- layman (to add overlays)
- vim (personal favorite editor)
- git (for client)
- eix (portage package information caching)
- hardened-sources (kernel sources)

The kernel configuration/build itself is premade (occasionally, the Linux kernel build is done and a binary package is created which can then be reused across all virtual instances) so is not actually tested.

The method to automatically perform this installation is scripted and available from <http://github.com/sjvermeu/small.coding/gensetup>.

2.2 Deviations

A few deviations are in place from a standard installation. These deviations are occasionally cleared to ensure that no wrong assumptions are made.

- A local mirror is included to speed up the downloading of source packages. If a source package is absent on the Internet but available on the local mirror, we will not catch this error. To remediate this, the local mirror is sometimes disabled.

3 Appendix: Facilitating Using BINHOST

Portage has the ability to use BINHOSTs for fast deployments.

Within the test setup, BINHOST can be used to improve build speeds. However, it is important to occasionally clear the BINHOST so that the build process is tested too.

3.1 Feeding the BINHOST

A binhost should be based upon a Gentoo Portage profile + make.conf setting (including USE flags). The moment this changes, a new BINHOST needs to be set up.

To set up a BINHOST:

1. Perform an entire installation up to the point that it is either finished, or that the profile or make.conf changes.
2. Run `quickpkg --include-config=y --include-unmodified-config=y` for all packages (hint: use `/var/db/pkg` or, even better, `emerge.log` output)

```
quickpkg --include-config=y --include-unmodified-config=y $(grep "completed emerge.* to ↵  
/" /var/log/emerge.log | awk '{print "=$8}')
```

3. Upload the entire `/usr/portage/packages` location to the BINHOST

In the future, you can refer to this BINHOST using `PORTAGE_BINHOST`.