Collection point for various ideas how we could make life more difficult for hackers - i.e. limit extent of a compromise. Examples:

- successful service compromise but cannot escalate to "root"
- successful "root" compromise but cannot install kernel-level rootkits
- successful "root" compromise but cannot capture new passwords
- successful "root" or service compromise but cannot reach the network

If an attempt is detected, we should log this - the next attempt might be successful. Unfortunately, patched Linux vulnerabilities tend not to log the attempt, i.e. the attacker gets to try as long as she likes without being detected. How best to report/detect such attempts is under discussion.. LinuxSecMonitoring.

SELinux

Fine-grained MAC, championed by NSA and Red Hat, based around the notion of "roles" (users and their processes) and "contexts" (files), with a set of rules that specifies which role can get access to what context, and how roles and contexts are allowed to change. A "policy" groups all of this together, and also specifies which context a file ought to have.

"On" (*enforcing*) by default, but protects only services for which a special "policy" exists (so-called "targeted" mode). In SLC4, the policy is monolithic, i.e. adding new services is hard (need to ship a new policy RPM, the thing changes regularly - maintenance headache). FC6/RHEL5 are supposed to have modular policies, i.e. services can bring along their own, much nicer... service-specific "policies" often have tuneable (boolean) parameters, e.g to selectively allow things the default policy would forbid.

Lots of documentation

- tons of (updated) man pages, including for the tuneables: `man -k selinux`
- NSA SELinux page
- NSA SELinux FAQ
- Fedora SELinux FAQ
- "unofficial" FAQ
- Red Hat SELinux Guide
- more info for SELinux on various distros
- Dan Walsh = Main developer's blog, lots of answers to real-life issues

Main challenge: prevent sysadmins from turning it off, either because they get a problem or are afraid they might do so later.. the thing is supposed to stop events from happening, after all.

- `ncm-selinux` exists to configure it on/off, tune parameters for Quattor machines. Not deployed anywhere (?)
- CERN-specific policies would have to be written for exposed network services
    - CASTOR: rfiod, rtcopyd
    - GridFTP
    - other Grid services
- requires file system support for extended attributes (this is where the context is stored), works on ext3, needs testing on XFS.
- turning SELinux on (on an existing machine) requires all (local) files to be labeled with their context. takes ages. **Not** required when going from *enforcing* to *permissive* and vice versa.

Hints: use `-Z` parameter to system utilities (such as `ls`, `ps`, `id`) to find out about SELinuxes idea of who you are, what a file is etc. Access problems get logged via syslog or **auditd** (in more detail).

Signed Kernel modules

Implemented by D.Howell and G.Kroah-Hartman in 2004, part of RHEL4. All SLC4 kernels have their own public key, "default" modules (those shipped in the kernel RPM) get signed on build. Private key is thrown away after build.

Initially, this caused some concern about DRM-style lockin ("this kernel will only run RedHat-signed modules.."), this seems to have died done this the mechanism isn't activated by default.

This is only part of a security solution, the kernel can be modified also via `/dev/mem`, `/dev/kmem` (see Zeppoo) and certain =ioctl=()s. And via bugs, i.e. unchecked parameters, errors on corrupted FSes etc.

May cause some issues on DAQ hardware where homegrown drivers need to be loaded. But could add a layer of security for CC machines and standard desktops.

Docs:

- LinuxJounal article and manual signing howto
- LKML discussion on if/why/how and original patch
- http://linuxfr.org/~ehoebadoag/15234.html (in french)

Use:

- turn on via `enforcemodulesig` on kernel command line (i.e. `/boot/grub/grub.conf`)
- in principle, multiple public keys in the kernel should be feasible - but needs a patch+recompile.

For some funny reason, the AFS module (usually standalone and therefore not signed) actually loads. Possibly because it does dirty tricks at startup (i.e. looks for syscall table, patch itself in..)? To check whether a module is signed (but not by whom), use

```
readelf -S tux.ko | grep module_sig
```

Understood: all the mechanics were in place on RHEL4.6, but the final check always granted access (fixed in 4.7, http://rhn.redhat.com/errata/RHBA-2007-0791.html)

Multiple signatures are required for this to be useful (otherwise no out-of-tree modules - we need several driver updates and OpenAFS) but can't be implemented in upstream RHEL because there is a `[..]patent on the process apparently held [b]y Microsoft.` (and internal discussion shot down a persistent signing key for Red Hat). IT#114551.

Signed (ELF) binaries

http://disec.sourceforge.net/ , description at http://disec.sourceforge.net/docs/digsig.pdf

DigSig is a kernel module that only execs **signed** binaries and libraries. Not feasible for machines running user-supplied code (but could perhaps be extended to validate only code running as uid=0/"root")?. Will store the signature in the ELF file, thereby invalidating any RPM/tripwire checksums... Unclear how this reacts to non-ELF code (scripts). Does not protect against a signed binary doing something stupid, i.e. being exploited - `[..]if malicious code is signed, all security benefits are lost.[..]` probably should be amended ".. and buggy ..".

Conclusion: not useful for general CERN machines.

Firewalling (iptables)

We have a two-layer firewall configuration - CERN-wide and machine-local via iptables. SLC4 comes with a default firewall that is rather restrictive for incoming traffic - but some services turn it off :-(. No egress filtering yet (to prevent backdoors talking back to their masters), though, and no central logging (Note: R.Wartel has central firewall loggin for certain Grid services). Certain kernel-level rootkit will be reachable despite a local firewall being in place, by intercepting traffic before the firewall sees them. Analysing firewall logs for intrusions is a pain, hard to automate and hard to prevent being totally swamped.

disable automatic module loading

The Linux kernel will autoload modules if the corresponding special device is being accessed. It will invoke a userspace utility (usually `sbin/modprobe`) to do so, this is configureable via `/proc/sys/kernel/modprobe`. Disabling the mechanism was a first-line defense against the ptrace exploit, and provided nice logging of the attempt. Every little bit helps. This workaround can be undone at runtime. Most useful for machines that have a defined HW configuration at the end of the boot process, not good for desktops with new hardware appearing out of the blue (USB sticks etc).

#CAP_SYS_MODULE A similar but more permanent (i.e. until reboot) result should be achievable by dropping the CAP_SYS_MODULE capability

```
#echo 0xFFFEFFFF &gt /proc/sys/kernel/cap-bound
```

per http://lists.freebsd.org/pipermail/posix1e/2000-June/000317.html we also would need to drop CAP_SYS_RAWIO to prevent direct memory manipulations via `/dev/mem`. Which might cause other trouble, but perhaps not on farm machines.

See `lockdown` script/RPM.

disable any module loading

as per above #CAP_SYS_MODULE trick, or by redirecting the kernel entry point (`init_module`), after a machine has reached production state (all modules loaded) and before untrusted users/services come up

See `lockdown` script/RPM.

`/dev/mem` protections

See thread at Kerneltrap why this is required - largely BIOS mem area access (`dmidecode` included, this is used in production for monitoring).

RHEL has some protections against unlimited `/dev/mem` access but these seem to be ineffective (RootKitChecker#ZeppooKernelMod); the `phalanx-b6` rootkit README says

```
I might also add that the people at Fedora have made some sort of
protection against these things, you aren't supposed to access
stuff over a 1M limit or so in /dev/mem. Unfortunately they are high
on cocaine and didn't implement it very well.
```

and happily writes to the `/dev/mem` FD after mmap'ing it..

Would need to identify `/dev/mem` users (read and/or write) on production machines, then see whether we can turn it off completely, allow only read access, or need to 'fix' the Red Hat patch by restricting to non-kernel memory areas.. (update: this fix has been deployed on RHEL5 and will come to RHEL4)

See `lockdown` script/RPM.

KIDS/StMichael/SMM/IBM IMA

StMichael (old=2000) : http://www.kernelhacking.com/rodrigo/docs/StMichael/

KIDS:
http://www.ekoparty.com.ar/archive/2007/ekoparty07%20-%20Rodrigo%20Branco%20-%20KIDS.pdf

http://www.csc.ncsu.edu/faculty/jiang/pubs/RAID08_NICKLE.pdf

IBM IMA: http://domino.research.ibm.com/comm/research_people.nsf/pages/sailer.ima.html

FIXME - check.

This topic: LinuxSupport > LinuxHardening
Topic revision: r3 - 2008-11-13 - JanIven