Why lock down the kernel?

Matthew Garrett
<mjg59@google.com>

What are we trying to do?

- There’s a barrier between users and root
- Should there be a barrier between root and the kernel?

Some prior art

- CONFIG_MODULE_SIG_FORCE
  - Root can only load modules that are appropriately signed
- CONFIG_STRICT_DEVMEM
  - Root can’t access arbitrary physical RAM
- Linux security modules
  - Root can be arbitrarily restricted

Why would we want to go further?

- Various security presumptions are based on the kernel being trustworthy
- In common configurations, users are effectively root
- Changing user behaviour to prevent them from touching root is impractical

When is a trustworthy kernel helpful?

- Audit policy
- Binary measurement and signature enforcement
- Trustworthy LSM policy

Not a novel idea

- BSD Securelevel blocked various types of access
- Added to NetBSD in 1994
- Suggested for Linux by Alan Cox in 1998
- Rejected by Linus

What about capabilities?

- Each process has a set of capabilities
- Some capabilities are required to do certain kernel stuff
- Add a new capability to control sensitive kernel access
- For secure kernel configs, just drop that capability by default

Unfortunately, as you probably already know, reality

- For security purposes, software drops capabilities
- Software drops all capabilities except the ones it needs
- Adding lockdown via capabilities adds a new capability to drop
- Apps either drop it (don’t work) or don’t drop it (insecure)
- Can’t re-use an existing capability without breaking userspace
- Capabilities: just say no

Why not an LSM?

- Not everything you want cleanly fits into the LSM approach
- Hard to retain compatibility with existing policy

What was the motivation?

- If root can replace the on-disk kernel, root/kernel boundary isn’t interesting
- UEFI Secure Boot meant that infrastructure for verified boot was widespread
- Wider verified boot ecosystem was moving in this direction

What is verified boot

- Firmware validates the signature on a bootloader
- Bootloader makes a determination about what it’s willing to boot

What is a bootloader

- Code that’s able to read a file
- Code that can relocate that file in physical RAM
- Code that can jump into that file in ring 0

The kernel is a bootloader

- Kexec is literally this
- (kexec also traditionally let you load arbitrary code into the kernel)

A bootloader that boots a bootloader

- If a signed bootloader boots a signed kernel that will boot unsigned code…
- Verified boot chain is broken
- insmod kexec
- Strong root/kernel barrier prevents this

Patchset initially posted in 2012

- Used capabilities approach
- Some arguments, little progress
- Replacement for kexec wasn’t ready at the time

Patchset posted again in 2013

- Some argument, little progress
- Linus objects quite strongly to a related patch
- Really quite strongly

Patchset posted in 2014

- Renamed in response to conversation at Linux Security Summit (with Linus)
- Tentative agreement
- Tremendous bikeshedding
- I find a new job

Time passes

- David Howells cleans up and extends patchset
- …and audits every module parameter
- (Thank you, David)

Where are we now

- Patchset reposted in 2018
- It… does not go well
- Lots of argument about one central thing

Conflation between security policy and boot policy

- Not all environments that want this patchset have secure boot
- Not all environments that have secure boot want this patchset
- When do we want this, and when don’t we?

How is the patchset enabled?

- Parameter in bootparam structure
- Kernel command line argument
- (Optionally) by default if on UEFI and Secure Boot enabled

Why provide this default?

- Use cases where lockdown provides real benefit without secure boot are rare
- Use cases where secure boot provides real benefit without lockdown are rare
- Default on UEFI Secure Boot, default off otherwise gives users easy control

What if firmware vendors don’t provide an option?

- Still waiting for someone to show me a system that fits this category…
- sudo mokutil --disable-validation

Patchset is useful even without default policy

- Even requiring a command line option or sysctl write is a benefit
- Maybe we should merge it without the default policy patch?
- Every mainstream distribution is shipping a variant of this at the moment