Non-technical security best-practices for open source projects

Greg Kroah-Hartman

gregkh@linuxfoundation.org

git.sr.ht/~gregkh/presentation-non-tech-security

Non-technical security best-practices for ~~open source~~ all projects

Greg Kroah-Hartman

gregkh@linuxfoundation.org

git.sr.ht/~gregkh/presentation-non-tech-security

About me

– Longtime Linux kernel developer (20+ years)
– Maintains Linux stable kernel releases
– Member of Linux security team
– Maintains some small Linux userspace packages
– Has helped develop 6 different Linux distributions

What this talk is NOT about

“Security Theater”

Infrastructure / Physical things

Language choices

What this talk is about

Documented things you can do better

Undocumented things that matter even more

Core Infrastructure Initiative Best Practices badge

https://bestpractices.coreinfrastructure.org/

bestpractices.coreinfrastructure.org

Lots of good things that all projects should do:

- Version numbering, Bug reporting, Release notes, Build systems, Test suites, Crypto practices, Code analysis, Documentation, Project oversight, Coding standards, Build systems, etc.

You need an OBVIOUS way to report “security” things

You need an OBVIOUS way to report “security” things

security@kernel.org

security@kernel.org

Created in 2005

7 day embargo limit, after we have a fix

No NDAs

No CVE assigments (**iwantacve.org**)

Drag responsible people in to fix bugs fast

www.kernel.org/doc/html/latest/admin-guide/security-bugs.html

security@kernel.org

We get “interesting” emails…

Majority of our work is saying “go report this over there.”

Average about one “real” security issue every week.

security@kernel.org

We love:

- Reports with instructions on how to reproduce
- Reports with fixes
- Reports from people who want to make Linux better

security@kernel.org

We hate:

- Reports of already public bugs
- Reports sent in Word files
- Reports sent in .pdf files
- Reports sent by “third party” organizations
- Requests for CVE assignments
- Companies that ignore us

Routing around **security@kernel.org**

Spectre / Meltdown Caused Intel to do their own thing

”This is different”

– Intel executive, now retired

Routing around **security@kernel.org**

Spectre / Meltdown Caused Intel to do their own thing

- Wanted to deal with companies, not community
- Siloed different companies with different information
- Caused multiple fixes to be created, all different
- Forgot that 75%+ Linux users are not company distros
- Did not allow developers to work together
- Caused a total mess

hardware-security@kernel.org

Created in 2019

Custom teams spun up for each issue

GPG/SMIME encrypted email

No NDA, but a “Memo of Understanding”

Embargo is negotiated based on hardware fix dates

www.kernel.org/doc/html/latest/process/embargoed-hardware-issues.html

Undocumented things that matter the most:

Undocumented things that matter the most:

Make it trivial to upgrade

Undocumented things that matter the most:

Make it trivial to upgrade

Do not make your users mad

Who are you?

Kernel developer
Library developer
Application developer

Kernel developers

Kernel developers

“Can not break userspace”

– The one rule of Linux kernel development

Kernel developers

We want users to always upgrade without worry.

“Users will not update if they are afraid you will break their current system.”

“Users will not ~~update~~ take your security fix if they are afraid you will break their current system.”

Kernel developers

Support existing features until there are no users

Kernel developers

Evolve system over time so no one notices

Kernel developers

Provide new ways of doing things

Kernel developers

Provide new ways of doing things
- eBPF, io_uring, realtime
- modern functionality on a POSIX system

Kernel developers

Create trust that people can rely on.

Kernel developers

We learned this the hard way

2.4 → 2.5 → 2.6 release lifecycle/nightmare

Kernel developers

We learned this the hard way

2.4 → 2.5 → 2.6 release lifecycle/nightmare

All releases are now stable, since December 2003

Library developers

Library developers

I pity you

Library developers

Hardest job of any developer

“You never know if an API is really useful, until you have too many people using it to ever be able to change it.”

Library developers

Can never break anything

Library developers

Fix “broken” apis by creating new ones

Library developers

Be strict in what you accept

Postal's law is NOT for APIs.

Library developers

Rusty's levels of workable APIs:

- blog post 1
- blog post 2
- original talk slides

Library developers

Keep supporting old apis for security issues

Provide guides for how to move to new apis

Library developers

Keep supporting old apis for security issues

Provide guides for how to move to new apis

- Documentation matters
- Do not rely on stackoverflow to get it right

Library developers

Evolution is addition only, never removal

`__attribute__((deprecated))` is evil

Library developers

Evolution is addition only, never removal

Major breaks are fresh starts, with no users

Library developers

Create trust so people use your code

Library developers

Create trust so people use your code

Never violate that trust on purpose

Application developers

Application developers

Most visible of all, easiest to criticize

Application developers

Like a traffic engineer, everyone has an opinion on where to place the road.

Application developers

Be very careful about taking away things that “work”

“The day my system broke”

– April 6, 2011

GNOME™

“The day GNOME developers learned they had real users.”

“The day GNOME developers broke their user’s trust.”

GNOME 3 lwn.net review

Article and comments are required reading

MATE and Cinnamon fill a real need

Application developers

Two acceptable ways to change

Application developers

Two acceptable ways to change

1) Evolve slowly so that no one notices

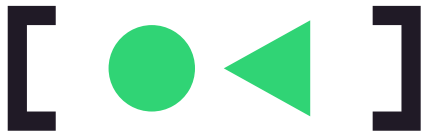

KDE

Application developers

Two acceptable ways to change

1) Evolve slowly so that no one notices

2) Provide a compelling reason to make the change



systemd

Application developers

Never break user’s trust, or they will leave

“If you make your users mad, they will not be your users anymore.”

“If you make your users mad, they will ~~not be your users anymore~~ run known-insecure software”

The most secure systems:

Make it trivial to upgrade

Do not make their users mad

git.sr.ht/~gregkh/presentation-non-tech-security