Safely Copylefted Cars: Reexamining GPLv3 Installation Information Requirements

Bradley Kuhn, Software Freedom Conservancy &
Behan Webster, Converse in Code Inc

Why Does Copyleft Exist?

- Admittedly, Copyleft was created by software freedom activists.
- Ultimately, GPL (both v2 and v3) are pursuing an activist agenda.

GPL has an activist agenda?!?!

Doesn't that mean it'll trick me into releasing my own code?!?

GPL is Pragmatic

- NO! The GPL, and its advocates, **do not want** to trick you into releasing code you don't want to release.
- GPL (both v2 and v3) are drafted for trade-offs between popularity (i.e., adoption) and assuring maximal software freedom (copy, share, modify, resell, reinstall) for downstream.

But "everyone" knows GPLv3 is much more restrictive than GPLv2, right?

GPLv3: “Just Different”

- GPLv2 is actually more restrictive in some ways.
- That’s why Software Freedom Conservancy promulgated “GPLv3-like termination” for GPLv2-works
- Concept now adopted upstream by many projects, including Linux in the “Linux Enforcement Statement”

Post-Hoc Perfection of GPLv2

- Many of the criticisms of GPLv3 were common about GPLv2 in the 1990s:
  - “can’t use in embedded products”
  - “difficult to comply with”
  - “confusing rules”
  - “will make my product insecure and unsafe”
- In fact, it’s just about learning the rules.

“Everyone knows you can’t use GPLv3 in embedded…”

With GPLv2 you must provide complete source code to downstream, but no installation details are required, right?

- Well, actually GPLv2 says you must include “scripts used to control installation”.

“Everyone knows you can’t use GPLv3 in embedded…”

But GPLv3 says you have to providing a way of installing modified GPLv3 software including your cryptographic keys for the device, right?

- GPLv3 does add a formal definition of “Installation Information”, but the definition is tightly bounded. Lets talk about this further…

Because Secure Boot

- Cryptographically locked down systems restrict what software can boot and execute on a platform
- Amongst other reasons, it is used to protect from hostile 3rd parties from installing malware

So the use of GPLv3 software means “no secure boot”, right?

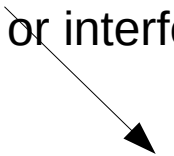
“Crypto-lock-down”

- Software freedom activists oppose crypto-lock-down as a policy matter.
- However, they want device owners to cryptographically trust & control their boot path.
- GPLv3 seeks to allow downstream user upgrade of the GPLv3’d software.

But that means you have to publish your signing keys, doesn’t it?

GPLv3 Installation Info

"Installation Information" for a User Product means any methods, procedures, *authorization keys*, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.



It doesn't say **your** keys.

Provide a way to install new firmware

- What if there was a way to provide a way to install modified firmware without having to provide your signing key?
- Some devices already support this...

The Tivo Paradox

- The Tivo Series 2 was likely **the first** User Product to exercise crypto-lock-down.
- Ironically, **even if Linux were GPLv3**, Tivo's method of crypto-lock-down **would likely comply** with GPLv3.

How is that possible?

The Tivo Paradox

- Tivo allows upgrade of Linux and other GPL'd software, by installing your own *unsigned* binary.
- Once you do, their **proprietary** user-space software that is not derived from / based on GPL'd software won't function, since the kernel checksum doesn't match.

Surely only bkuhn thinks though, right?

The Tivo Paradox

- *bkuhn checked this with RMS: even he agrees this mechanism complies with GPLv3.*

But, that thing about “continued functioning”?

GPLv3 Installation Info

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. *The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.*

It doesn't say *all* code on device!
It's *just* the GPLv3'd code
(and modified versions of it).

Tivo's proprietary application is moot: it's not a combined/derivative work with/from Linux, or bash, or (to our knowledge) any GPL'd package.

Unlocking a Google Phone

- Google Nexus/Pixel phones allow *`fastboot`* to unlock the boot-loader: allows 3rd party ROMs
- This process deletes all personal information and resets the phone to default settings
- Some Android phones blow a fuse or set a flag in the firmware that indicate software tampering
- The boot screen of the phone indicates whether the boot-loader is locked or not

Thought experiment: a Smart Thermostat

- Apply this to a simple example: a Linux-based “smart thermostat”
- Has cloud connectivity and a mobile app
- (Ignoring cloud security which is a separate issue)
- The manufacturer cryptographically locked down the thermostat.

Dev board mode

- So maybe, like an Android phone, user could put the thermostat into a “development board” state
- Allow the thermostat to transition into a mode which, like Tivo, allows upgrade of the kernel, bash, etc.
- It allows 3rd party user-space applications to run, but the proprietary user-space app won’t run on any kernel which isn’t signed with company’s private key.

Unlocking the boot-loader

- You could put the boot-loader into a state where it allows unsigned code to execute.
- Secure Boot & trusted boot chain would essentially be disabled when user modifies Linux, bash, etc.
- As a result no signing key is required to install new software.

But how will people know that the device has been “unlocked”?

Tamper Evidence

- Just like the phone, the boot-loader indicates at boot that the device is no longer in stock mode.
- One could also use HW to blink LEDs or force the GPU to always display a red box around the display or equivalent

But they'll have access to disrupt my cloud service, won't they?

Remove cloud service authentication

- But also, once in “dev board” mode, your proprietary user-space application won’t run anyways. How will they even authenticate to your cloud service?
- Users’ realistic option is to write their own cloud server backend.

What if the wifi network device was forcibly disabled in dev board mode?

Disabling network hardware

- It would violate GPLv3 if you disabled the wifi hardware (or equivalent) of a non-disruptive device.
- GPLv3 gives narrow permissions for a forcible network ban

Wait, GPLv3 says I can fry the wifi chip if the end user disrupts my network?

GPLv3 §6 ¶6

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. *Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.*

If the dev-board mode device is disruptive,
you can unequivocally deny all network access.

But they'll have access to my proprietary user space app and can reverse engineer it, right?!?

Delete proprietary code

- It’s now a dev board, not a thermostat.
- You don’t need to leave your proprietary SW bits behind. Delete them if you want to.
- It will annoy software freedom activists but: nothing in GPLv3 says you can’t have encrypted non-GPL binaries that only decrypt **after** the crypto-lock-down boot is verified.
- This outcome is reasonable for software freedom: inspires users in Maker communities to “do their own” and still buy your hardware!

But this could be used to hack my thermostat!?!

One way transition

- Indeed if you allow a device to go into dev board mode and back again to stock, that **might** be manipulated to a malware path
- However, you **don’t** have to allow for reinstatement of a stock mode

GPLv3 §6 ¶6

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

This allows for a one way transition to dev board mode

So the transition to Dev board could be one way?
Is there no way back to stock firmware?

One way transition

- The transformation into a dev board could be permanent
- The warranty is void; you can permit stock recovery (or not).
- Make this **very clear** as people take their thermostat into dev board mode.

So what does this leave for the user, exactly?

Dev board

- Users have a little Linux dev board with Wifi,
- It does nothing other than run the GPL'd parts of the shipped software (+ user mods).
- User was informed that the original firmware is gone and can't come back.
- It probably won't even control the temperature anymore (until community devs write their own software to do so)

But surely automotive is different?

Automotive *is* DIFFERENT!

- We know that a car is not a thermostat.
- We know that a car has safety regulations.
- Yet, these same compliance principles apply.
- They just cause a slightly more goofy outcome.

Applied to Automotive

- Presumably the GPLv3 software is just part of low-level OS (e.g. bash running on the Linux kernel) of the In “Vehicle Infotainment unit’ or “Head Unit”.
- This unit mostly controls interior passenger related things, or displays car-related info
- Although IVIs don’t typically directly control the drive by wire systems, they are usually connected to them.
- Thus, the IVI is likely crypto-locked-down.

Making a car into a dev board

- Through some mechanism the idea is to put your IVI into an unlocked mode allowing new software to be installed

But, a “dev board” the size of a car?!?!

Barreling down the highway at 120 kph?!?!

Tamper evidence for the car

- The ECU authenticates the IVI at ignition (powerup) and determines the mode the car is in.
- If in “dev board” mode, lights can be flashed, dash instruments (if they exist) can be redlined, etc.

But they can still start the engine and drive the car can’t they?!?

We can’t allow that!

drive-by-wire access

- The ECU knows the crypto-lock-down is violated.
- After all, the car is now a car-shaped dev-board.
- Assuming that you didn't put GPLv3 software in the parking brake and the engine, just **lock down** the parking brake & refuse to start the engine.

But they have access to my proprietary IVI application now, don't they?

IVI Software

- At this point, it's the thermostat example.
- You can delete any of your proprietary SW when transitioning to dev board mode.
- There is no expectation that the IVI can still do what it did before becoming a dev board,
- As long as the modified user installed code can execute, you're in compliance with the GPLv3 in this case.

But what about taking it back into "car mode"?

Making it a car again

- You don’t actually need that giant hunk of metal to function as a car again.
- A one way transition is permissible.
- Remember, the warranty is void; stock recovery is at your option.
- *(Of course, you should **clearly communicate** this goofy outcome to your customers.)*

Is this really what the GPLv3 was trying to do? A 4-wheeled dev board?

A dev board with 4 wheels, stuck in the driveway forever.

- Is this really what GPLv3 intended? *Probably not!*
- GPLv3 admittedly envisioned a much wider body of GPLv3'd software.
- If your IVI system has only user-space GPLv3'd **applications**, not the operating system: that's a topic for another talk.
- But we've shown that you can put, say, GNU bash in your cars with lock-down infrastructure that you probably have today.

If This Doesn’t Work, You May Have Bigger Problems.

- No system is perfectly tamper proof.
- If **you** and certified mechanics can upgrade your IVI system, someone else (benevolent & nefarious) probably can, too.
- Any safe vehicle needs to be able to tell **instantly** on power-up whose OS it’s running and what that means for safety.
- And if you can do that already, you can comply with GPLv3’d OS software **today**.

YMMV

- We know these ideas don’t solve all use cases
- As always, you need to consult with your lawyers
- There are many regulations in various districts
- But this gives you a starting point with which to start the conversation & a safe path forward.

k

Thank you

- Questions?
  - (Also, we'll be around for the rest of the conference if you want to talk to us later)

Software Freedom Conservancy

- Bradley works for a US-based charity
- Become a supporter

https://sfconservancy.org/supporter

You can reach us at

- Behan Webster <behanw@converseincode.com>
- Bradley Kuhn <bkuhn@sfconservancy.org>