SONY®

Issues with Open Source License Compliance in Consumer Electronics

Tim Bird

Principal Software Engineer

Sony Electronics

Abstract

Complying with the myriad licenses for software that are used in a modern consumer electronics device can be a complicated process. In this talk, Tim will discuss lessons learned from license compliance activities with Sony consumer electronics products. Tim will describe offers for source, software distribution, and rebuildability of provided source. The GPL license requires "complete and corresponding source". Tim will explain what this means, and how that interacts with things like secure product lockdown. The history of the GPL v2 license, and intent of Linux kernel community leaders will be presented.

Attendees should gain a better understanding of compliance requirements, and what issues to watch out for in managing the source code and requests for source for their embedded Linux products.

Agenda

- Legal disclaimer
- Aspects of Compliance
- License History
- Gotchas
- Recommendations

Disclaimer – I am not a lawyer

- The law is different than code
  - I’m a software developer, who has been around lawyers some
  - There are lots of unexpected things in law, that only people with legal experience know about
- Don’t trust what you read on the Internet, or what you hear from me
  - People make strong statements about what’s required, but it’s just their opinion
  - Very few legal court decisions interpreting Open Source licenses, GPLv2 in particular
    - Ambiguous language means different contributors can have different views of the same term under the same law (i.e. what is a “medium commonly used for software interchange”?)
    - Further problems: Open Source licenses have no choice of law so the same issue may decided differently depending on the location of the dispute (i.e. German law is different from US law)
  - Check with your own company’s legal counsel
- These are my views
  - I’m not speaking on behalf of Sony here, even though their logo is right up there in the corner

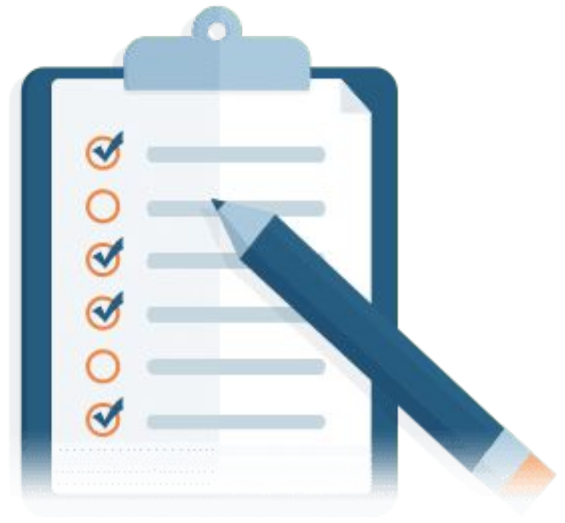
Pet Peeves

- People who say "It's trivial, just post the source"

- That works for (some) in-house developed products
  - Often even in-house products include components with their own embedded software
- Many modern products are developed by multiple vendors
  - Example: Mobile phone (Google, MediaTek/Qualcomm, Sony)
    - And others in the supply chain
- Some business units do black-box (white label) products
  - Example: Some old security cameras were just relabeled 3rd party products
- Some business units are just integrators of high-level components
  - Example: Sony B2B puts together A/V/IP packages for stadium suites

Aspects of license compliance

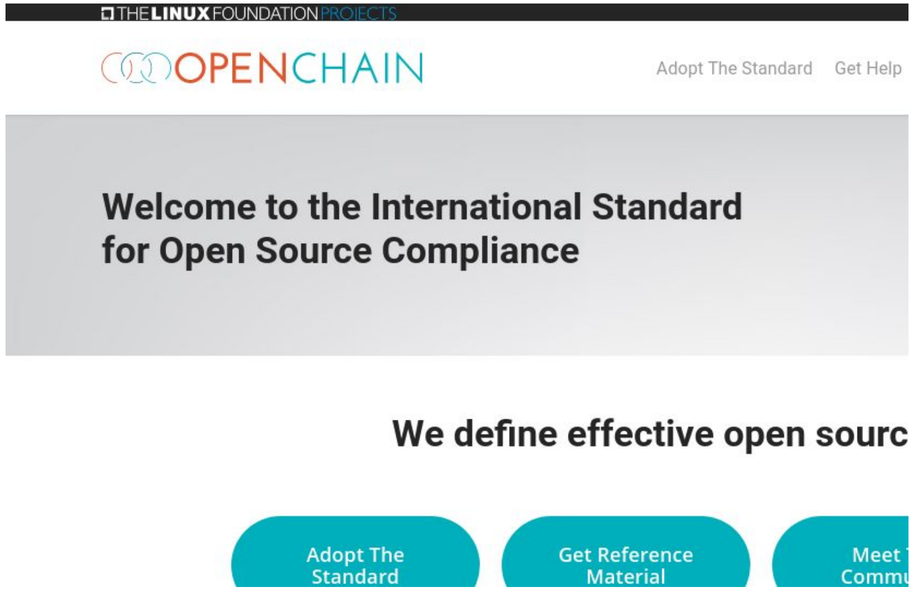
- Knowing what's in your product
- What you provide:
  - License attribution
  - Offer for source
  - Source code
- Providing source (only required for some licenses)
  - Duration
  - What's needed



Knowing what’s in your product

- You MUST know what’s in your product to fulfill the license
- For a “big” product, expect to maintain a spreadsheet with thousands of rows
- Tools to scan
  - Black Duck
- Tools to manage manifests
  - Fossology
- Suppliers
  - Often a source of problems
  - Many suppliers do not take their responsibilities seriously
  - OpenChain workgroup focuses on supply-chain education and certification

OpenChain project


THE LINUX FOUNDATION PROJECTS

OPENCHAIN

Adopt The Standard Get Help

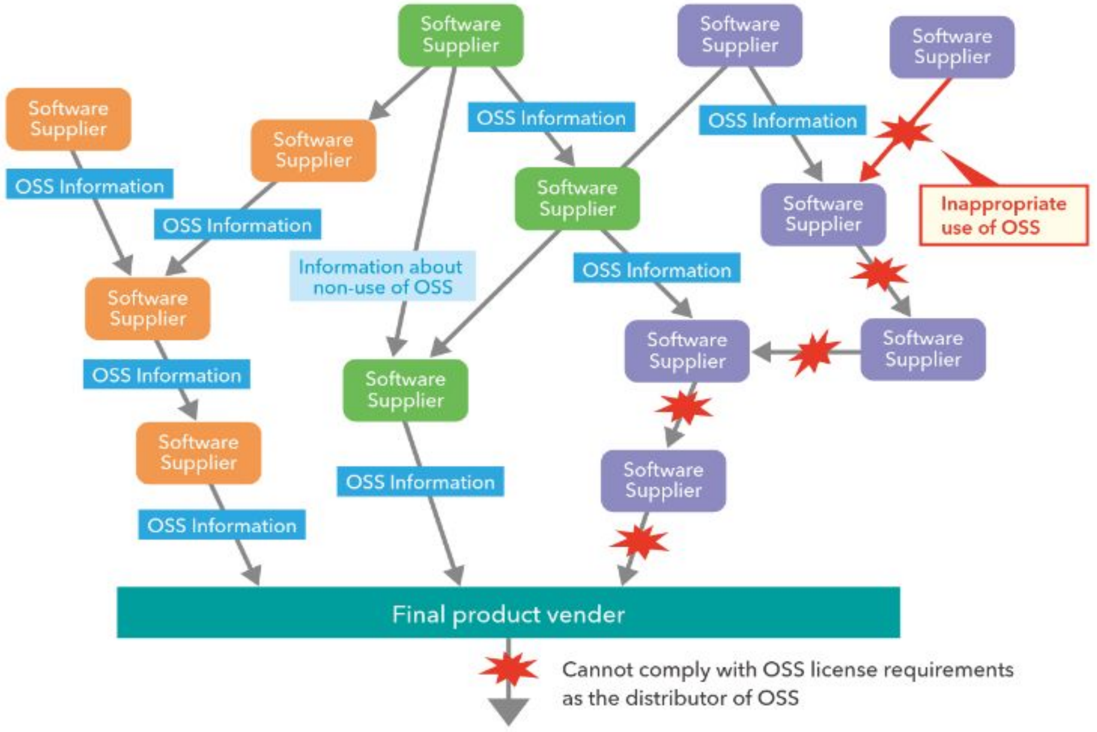
Welcome to the International Standard
for Open Source Compliance

We define effective open sourc

Adopt The Standard

Get Reference Material

Meet
Commu

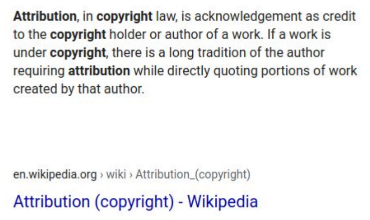

Software Supplier
OSS Information
Information about non-use of OSS
Inappropriate use of OSS
Final product vender
Cannot comply with OSS license requirements as the distributor of OSS

Knowing what's in your product

- Is useful to build the software from your supplier
  - To validate that the source provided matches the binaries in the component

Attribution

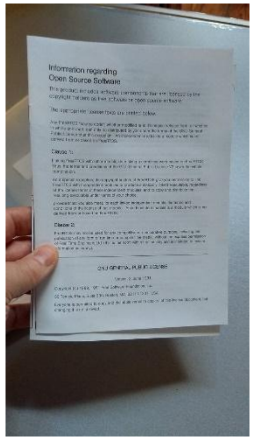
- Most licenses (even permissive ones like MIT and BSD) require that you mention someone somewhere
  - Some licenses only require attribution in the source, but most vendors include it in the legal information page with the product.
- Some licenses require that the license text accompany the product
- Where to put it:
  - Used to put things in paper manual
    - It’s gotten way too long for that



Attribution, in **copyright** law, is acknowledgement as credit to the **copyright** holder or author of a work. If a work is under **copyright**, there is a long tradition of the author requiring **attribution** while directly quoting portions of work created by that author.

en.wikipedia.org › wiki › Attribution_(copyright)

Attribution (copyright) - Wikipedia

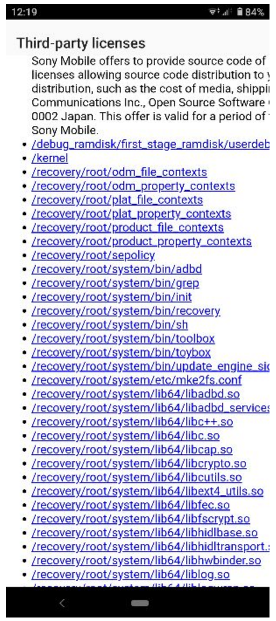


Attribution requirements, by license

- MIT – copyright and license with all copies of software
- BSD2 and 3 – copyright and license with source, and in documentation or other materials accompanying the binaries
- BSD4 – same as BSD2/3, *and* notice in advertising materials (ugh!)
- Apache2 – license and change notes with source, license and attributes with NOTICE text (in docs or display)
- GPLv2 – copyright and license and changelog with source, full source or written offer for source with binaries
- MPL2 – reference to license with source, offer for source and reference to license with binaries

Attribution – common practice

- For most consumer electronics devices with displays:
  - Put all copyrights and all licenses in a single page on the device accessible via a menu item
    - Xperia phone list of copyrights and licenses is about 400 phone screen pages long
    - See ‘About’, ‘Legal Information’ on your phone
- For products with no display and no paper docs:
  - Provide a web link on product or product packaging, and put all notices there
    - Have heard of inscribing a single web link on the product itself


12:19 84%

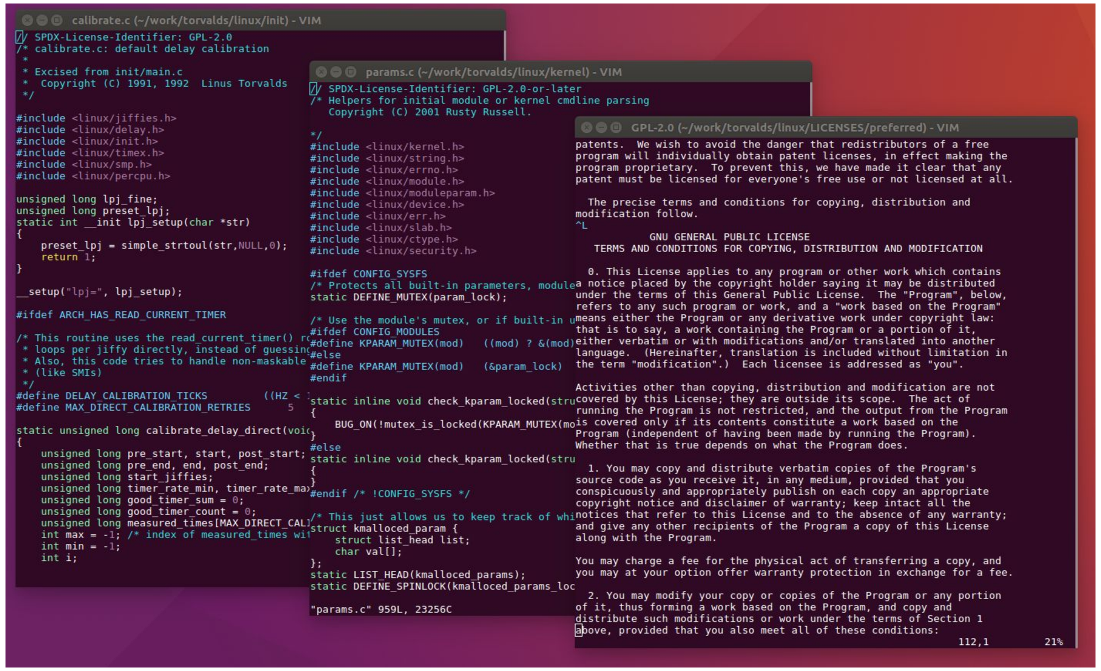
Third-party licenses

Sony Mobile offers to provide source code of
licenses allowing source code distribution to
distribution, such as the cost of media, shippi
Communications Inc., Open Source Software
0002 Japan. This offer is valid for a period of
Sony Mobile.

- /debug_ramdisk/first_stage_ramdisk/userdeb
- /kernel
- /recovery/root/odm_file_contexts
- /recovery/root/odm_property_contexts
- /recovery/root/plat_file_contexts
- /recovery/root/plat_property_contexts
- /recovery/root/product_file_contexts
- /recovery/root/product_property_contexts
- /recovery/root/sepolicy
- /recovery/root/system/bin/adbd
- /recovery/root/system/bin/grep
- /recovery/root/system/bin/init
- /recovery/root/system/bin/recovery
- /recovery/root/system/bin/sh
- /recovery/root/system/bin/toolbox
- /recovery/root/system/bin/toybox
- /recovery/root/system/bin/update_engine_si
- /recovery/root/system/etc/mke2fs.conf
- /recovery/root/system/lib64/libadbd.so
- /recovery/root/system/lib64/libadbd_service
- /recovery/root/system/lib64/libc++.so
- /recovery/root/system/lib64/libc.so
- /recovery/root/system/lib64/libcap.so
- /recovery/root/system/lib64/libcrypto.so
- /recovery/root/system/lib64/libcutils.so
- /recovery/root/system/lib64/libext4_utils.so
- /recovery/root/system/lib64/libfec.so
- /recovery/root/system/lib64/libfscrypt.so
- /recovery/root/system/lib64/libhidlbase.so
- /recovery/root/system/lib64/libhidltransport.
- /recovery/root/system/lib64/libhwbinder.so
- /recovery/root/system/lib64/liblog.so

Offering source

- GPLv2 gives 2 options (for commercial products):
  - Provide source along with product
  - Provide offer for source


calibrate.c (~/work/torvalds/linux/init) - VIM
params.c (~/work/torvalds/linux/kernel) - VIM
GPL-2.0 (~/work/torvalds/linux/LICENSES/preferred) - VIM

```
// SPDX-License-Identifier: GPL-2.0
/* calibrate.c: default delay calibration
 *
 * Excised from init/main.c
 *  Copyright (C) 1991, 1992  Linus Torvalds
 */

#include <linux/jiffies.h>
#include <linux/delay.h>
#include <linux/init.h>
#include <linux/timex.h>
#include <linux/smp.h>
#include <linux/percpu.h>

unsigned long lpj_fine;
unsigned long preset_lpj;
static int __init lpj_setup(char *str)
{
    preset_lpj = simple_strtoul(str,NULL,0);
    return 1;
}

__setup("lpj=", lpj_setup);

#ifdef ARCH_HAS_READ_CURRENT_TIMER

/* This routine uses the read_current_timer() ro
 * loops per jiffy directly, instead of guessinc
 * Also, this code tries to handle non-maskable
 * (like SMIs)
 */
#define DELAY_CALIBRATION_TICKS          ((HZ < 1
#define MAX_DIRECT_CALIBRATION_RETRIES      5

static unsigned long calibrate_delay_direct(voic
{
    unsigned long pre_start, start, post_start;
    unsigned long pre_end, end, post_end;
    unsigned long start_jiffies;
    unsigned long timer_rate_min, timer_rate_max
    unsigned long good_timer_sum = 0;
    unsigned long good_timer_count = 0;
    unsigned long measured_times[MAX_DIRECT_CALI
    int max = -1; /* index of measured_times wit
    int min = -1;
    int i;
```

```
// SPDX-License-Identifier: GPL-2.0-or-later
/* Helpers for initial module or kernel cmdline parsing
   Copyright (C) 2001 Rusty Russell.

*/
#include <linux/kernel.h>
#include <linux/string.h>
#include <linux/errno.h>
#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/device.h>
#include <linux/err.h>
#include <linux/slab.h>
#include <linux/ctype.h>
#include <linux/security.h>

#ifdef CONFIG_SYSFS
/* Protects all built-in parameters, module
static DEFINE_MUTEX(param_lock);

/* Use the module's mutex, or if built-in u
#ifdef CONFIG_MODULES
#define KPARAM_MUTEX(mod)   ((mod) ? &(mod)
#else
#define KPARAM_MUTEX(mod)   (&param_lock)
#endif

static inline void check_kparam_locked(stru
{
    BUG_ON(!mutex_is_locked(KPARAM_MUTEX(mo
}
#else
static inline void check_kparam_locked(stru
{
}
#endif /* !CONFIG_SYSFS */

/* This just allows us to keep track of whi
struct kmalloced_param {
    struct list_head list;
    char val[];
};
static LIST_HEAD(kmalloced_params);
static DEFINE_SPINLOCK(kmalloced_params_loc

"params.c" 959L, 23256C
```

```
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and
modification follow.
^L
            GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
under the terms of this General Public License.  The "Program", below,
refers to any such program or work, and a "work based on the Program"
means either the Program or any derivative work under copyright law:
that is to say, a work containing the Program or a portion of it,
either verbatim or with modifications and/or translated into another
language.  (Hereinafter, translation is included without limitation in
the term "modification".)  Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.

  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.

You may charge a fee for the physical act of transferring a copy, and
you may at your option offer warranty protection in exchange for a fee.

  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:
                                                  112,1         21%
```

Providing source with the product

- GPLv2 says:
  - “Accompany it with the complete corresponding machine-readable source code ... on a medium customarily used for software interchange”
- Some options:
  - A CD
  - USB stick
  - Internal storage on the product itself

- Does solve some issues:
  - Don’t have to create a download site and keep it populated for a period of time
  - Don’t have to do physical fulfillment
  - Don’t have to give to any third party (only product recipients)
- Not used very often
  - Internal storage requires a product with accessible and plentiful storage
  - Presents problems for software updates (extra size, impractical to ship a USB stick with an update)

Providing an offer for source

- GPLv2 says:
  - “Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, ... on a medium customarily used for software interchange”
- Most community members view a download site on the Internet as sufficient to fulfill this
- Lawyers believe that this should be interpreted as the medium customarily used for software interchange in 1991, when the license was written
  - ie Offering only an Internet download is not sufficient, offer of CD is required

Source offers – weirdness in Germany

- Licenses don’t include choice of law so decisions in US (and other countries) don’t apply to Germany
- License can be interpreted differently in different countries
  - One German court ruling says you MUST provide the option to obtain the software on physical media under GPLv2
    - So there’s legal precedent in Germany to require an offer to distribute physical media
  - German courts took cases seriously, including minutia about source offer
  - Certain features of German law were used by one litigant to gain a lot of money with a particular legal strategy
    - Read resource page at end of slides for more information
    - Bottom line: DON’T sign any compliance settlement agreement in Germany without reading up on the strategy and consulting with lawyers
- Ambiguity in license text means different interpretations by different contributors are possible even in the same country

Providing source - duration

- Must provide source for 3 years from time of <u>last shipment</u> of *binary*
- This is not the last shipment of product, but of the shipment of any firmware update for the product
  - Note – not the release of the firmware but the last time someone downloads it
- Usually not a problem. There's little incentive for companies to remove old source from a download site, so stuff hangs around a long time
  - Technically, you could remove the source for a firmware on the 3-year anniversary of when you took the firmware off your web site, but nobody does this

Providing source - observation

- If someone asks you for physical media, they are most likely a person with an interest in compliance enforcement
    - Handle requests promptly and carefully

What must source include?

- GPLv2 says
  - “For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable.”
- Term: Complete and corresponding source (or ‘CCS’ for short)
- GPLv2 also says:
  - “The source code for a work means the preferred form of the work for making modifications to it.”
  - You can NOT provide obfuscated code
    - Unfortunately, I have seen an example of this from a supplier

About Rebuilding

- Does the definition of CCS require that a recipient can reproduce binaries exactly?
  - I would argue no – reproducible builds has been a sore subject for decades
- Does it require distribution of compilers, linkers, IDEs, or proprietary tools?
  - No
  - This raises an interesting (and untested) facet of Open Source: Is it compliant to use proprietary tools to build Open Source software? Is it compliant to use tools unavailable to the general public?
- My view:
  - "Scripts used to control compilation and installation" means autotools, configure scripts, config files, and Makefiles, for most packages

SONY®

About Reinstalling

- Because the license requires “scripts used to control ... installation”, some people interpret that to mean that it must be possible to re-install the software on a particular device
  - The license does not actually say that
  - Neither the act nor target of installation is specified
- Industry practice (since 1991 and through today) fully supports use of GPL v2 software in ROM
  - Where the software can NOT be replaced

SONY

License history

- BSD licenses go back to 1988
  - BSD 3-clause created in 1999 (removed advertising clause from BSD4)
- GPL v1 developed in 1989
  - As a generalization of the EMACS license
    - History is pretty interesting. See http://www.free-soft.org/gpl_history/
- GPL v2 released in 1991, was adopted by Linus Torvalds shortly thereafter
- GPL v3 released in 2007, after a long discussion period
  - Linux kernel developers explicitly refrained from adopting GPL v3, for a number of reasons

Interpreting the license

- Overriding legal principle: Intent only matters if the license is unclear
- License doesn’t have it’s own “intent”
  - License is an expression of intent by the copyright owners of the software, not the author of the license (copyright owner could be a company)
  - Different software authors on a software project may have different intent
    - Who’s intent matters?
      - It can’t be just anyone – this way lies madness
- Linus Torvald’s intent for the kernel:
  - “tit for tat” – I give you source, you give me source back (and that’s it)
    - Well documented, numerous times

License source requirements – MY VIEW

- My view:
  - Give people the source – all the source, including Makefiles, configure scripts (output from autoconf, automake) and config files
- Customers should be able to build the software
  - No requirement to exactly reproduce your build
  - No requirement to install on a particular device
    - e.g. Customer can use any fixes you made to bash on their desktop or elsewhere
    - e.g. Can use any fixes for Qualcomm chipset on a phone of their own making
- Vendors are not required to produce hardware that can run binaries produced by 3rd parties such as customers
- Secure product lockdown is allowed by GPL v2

GPL v2 and GPL v3

- GPL v3 sought to impede lockdown, and changed the terms
    - Requires keys needed to install software for certain products
- This is why most CE vendors (including Sony) strongly avoid GPL v3
- Two views:
    - 1) This merely changes the terms to reflect the original intent
        - Some claim that the GPL v2 had this intent all along
    - 2) If GPL v2 actually incorporated this requirement, you wouldn't need to change the wording
        - Some people who used GPL v2 did NOT intend to disallow lockdown
- Many projects adopted software based on the wording of GPL v2 not GPL v3
    - Richard Stallman says it's evil to change the terms of the deal post hoc

Gotchas

- Some nuances that can bite you:
- MPL 1.1 versus MPL 2.0
  - MPL 2.0 has CCS wording closer to GPL, MPL 1.1 does not
- Packages have switched from GPL v2 to GPL v3 (example: bash)
  - Have to pay attention to version if you’re avoiding GPL v3
  - Patches submitted to project might have ambiguous license
    - Example: shellshock patch for bash
      - Submitted simultaneously to 4.3 (gpl v3) and 3.2 (gpl v2) versions of bash
    - My view: It depends on a lot of factors
      - Who authored the patch? Was it submitted for both versions of the software simultaneously? Did someone backport the patch?
- Packages that are mostly one license, with exceptions:
  - util-linux-ng (mostly GPLv2, with some tools GPLv3)

Reinstallation and unlockability

- Unlockability not required by GPLv2, but let’s discuss it anyway
- Fully unlocked/unlockable
- Not unlockable
- Unlock and reinstall OSS only

SONY

Fully unlocked/unlockable

- Product can be unlocked by consumer, who can reinstall all software and retain all keys to stream content or connect to backend services
  - Not required by any open source license (including GPL v3)
  - Open Source licenses only cover that software, not proprietary bits
- Issues:
  - That's simply not supportable as a business proposition <u>for some products</u>
    - Example: video game consoles and network play
  - If you can unlock the product, you can break the rest of the encryption and get the content keys, or cheat in online gameplay (trust me on this one)
  - The reality is that many people involved in breaking the security of products are doing so solely to access content or backend services in an illicit manner
- Only works in a world of perfect, honest actors

Not unlockable

- Most CE companies lock down the product and don’t look back
- Security is often done in layers
  - Encrypted update images
  - Trusted execution rings, TPMs
  - Encrypted filesystems
  - Kernel, module, and application signing and verification

- A few counterexamples I’m aware of:
  - Some Sony Xperia phones
  - Some Google Pixel phones
- These prove it’s possible to make an unlockable product, with certain constraints

Unlock product and reinstall OSS only

- Product can be unlocked by consumer, but content keys and backend keys are removed
- Notes:
  - Allows consumers to install modified Open Source software licensed under GPLv2
  - Protects the interests of content industry (e.g. NetFlix, Amazon, etc.)
  - No obligation to make the proprietary software work with the modified Open Source software and may require removal or disabling of proprietary software on the device
    - Device may lose a lot of functionality, depending on stack
  - Customer demand is very low for this – it’s a hard sell to your business unit
  - Can be a one-way action
- Nice alternative if you can achieve it:
  - Support reinstallation of official firmware
    - With content keys and proprietary software
    - This is technically difficult. Some flash devices have write-once encryption keys. Once written, they can never be re-written.

What Tim recommends

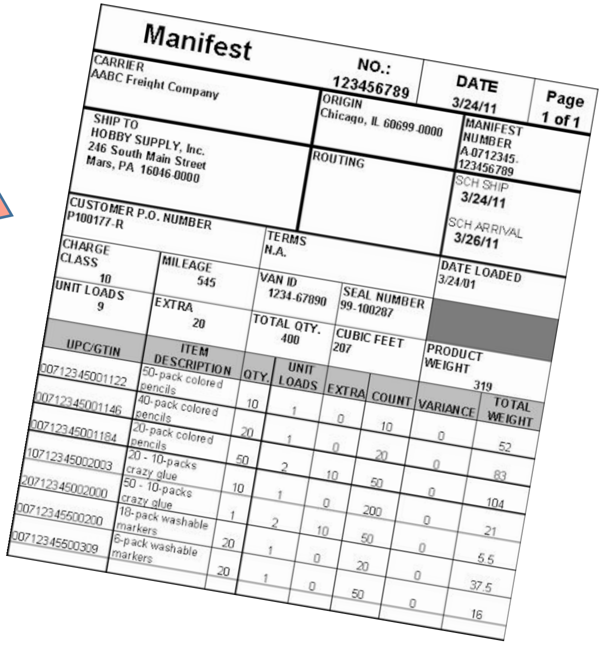
- Communicate with upstream suppliers
- Get a license manifest
- Build what you get from your supplier
- Lockdown or not?
    - Work with your company to decide your strategy on this issue since unlockability is not required under GPLv2

Recommendations - communicate

- Communicate with upstream suppliers, in advance
- Get your suppliers involved with OpenChain
- Several major vendors I’m aware of have had “oh crud” moments, where they found out a supplier did not disclose open source in some component
- Ideas:
  - Put penalties for failure to disclose open source software in procurement agreements with suppliers
  - Or, require permission in advance to deliver components with OSS

Recommendations – license manifest

- License manifest:
  - Should ask your supplier to provides a manifest of ALL the files in the device
  - Indicating the license for every file
- Don't do license tracking and source publication as an afterthought



Manifest

CARRIER AABC Freight Company	NO.: 123456789	DATE 3/24/11	Page 1 of 1
SHIP TO HOBBY SUPPLY, Inc. 246 South Main Street Mars, PA 16046-0000	ORIGIN Chicago, IL 60699-0000	MANIFEST NUMBER A-0712345-123456789	
	ROUTING	SCH SHIP 3/24/11	
CUSTOMER P.O. NUMBER P100177-R		SCH ARRIVAL 3/26/11	
CHARGE CLASS 10	MILEAGE 545	TERMS N.A.	DATE LOADED 3/24/01
UNIT LOADS 9	EXTRA 20	VAN ID 1234-67890	SEAL NUMBER 99-100287
		TOTAL QTY. 400	CUBIC FEET 207
		PRODUCT WEIGHT 319	

UPC/GTIN	ITEM DESCRIPTION	QTY.	UNIT LOADS	EXTRA	COUNT	VARIANCE	TOTAL WEIGHT
00712345001122	50-pack colored pencils	10	1	0	10	0	52
00712345001146	40-pack colored pencils	20	1	0	20	0	83
00712345001184	20-pack colored pencils	50	2	10	50	0	104
10712345002003	20 - 10-packs crazy glue	10	1	0	200	0	21
20712345002000	50 - 10-packs crazy glue	1	2	10	50	0	5.5
00712345500200	18-pack washable markers	20	1	0	20	0	37.5
00712345500309	6-pack washable markers	20	1	0	50	0	16

SONY

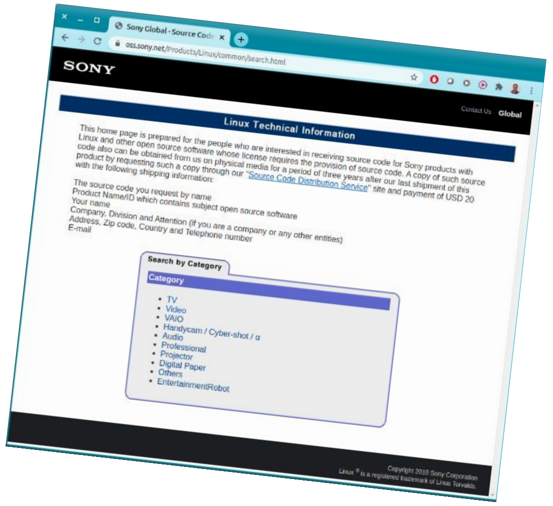
Recommendations – build and verify

- Build what you get from your supplier
- Ask that source be accompanied by a "compliance-build" script that does a start-to-finish build of all the open source software
- Use a docker container or VM to encapsulate the build environment
  - Build environments become obsolete and increasingly hard to reproduce
  - Dockerfile captures the host OS, toolchains and other programs used for the build, without distributing the toolchain binaries themselves
    - This avoids "compliance recursion"
- If the developer is an internal product team (not a supplier):
  - A separate internal group should be able to build what the team produces
- *This is aspirational*

Recommendation – source availability

- Provide an Internet download site
  - Make it easy to find and obtain source code for your product
  - Publicly available to all (not just customers)

- Also provide an offer for physical media
  - Should not get much traffic
  - Make sure that it works


Sony Global - Source Code
oss.sony.net/Products/Linux/common/search.html
SONY
Contact Us
Global

Linux Technical Information

This home page is prepared for the people who are interested in receiving source code for Sony products with Linux and other open source software whose license requires the provision of source code. A copy of such source code also can be obtained from us on physical media for a period of three years after our last shipment of this product by requesting such a copy through our "Source Code Distribution Service" site and payment of USD 20 with the following shipping information:

The source code you request by name
Product Name/ID which contains subject open source software
Your name
Company, Division and Attention (if you are a company or any other entities)
Address, Zip code, Country and Telephone number
E-mail

Search by Category

Category

- TV
- Video
- VAIO
- Handycam / Cyber-shot / α
- Audio
- Professional
- Projector
- Digital Paper
- Others
- EntertainmentRobot

Copyright 2018 Sony Corporation
Linux ® is a registered trademark of Linus Torvalds.

Recommendations - unlockability

- Unlocking your product is not required by GPLv2 (in my view)
  - But it makes community members and some customers happy
  - Due to the uncertainty, you should get legal advice
- If you support unlocking:
  - Provide a way for customers to unlock the product and install modified Open Source binaries licensed under GPLv2
    - If a product deals in protected content or services, you'll have to figure out how to disable access to those from modified Open Source software
      - This is more difficult than it sounds
      - It's easiest and safest to just remove proprietary software and keys
    - If unlocking involves permanently removing content protection keys or disabling proprietary applications, so be it
- If you don't support reinstallation, you must avoid GPL v3 software

Final statements

- I hope I haven’t scared you
- Compliance with the license is required
- It’s not trivial (depending on the circumstances), but it is achievable
- Due to legal uncertainty, each company must make its own decisions on how much risk to undertake
- Really should check out OpenChain project

Questions?

SONY®

SONY

Supporting information

- Statements by:
  - Linus Torvalds
  - Richard Stallman
  - SFC
  - About certain cases in Germany

Linus Torvalds on social contract with Linux

- From https://youtu.be/PaKIZ7gJlRU (video from Debconf 2014)
  - (at 0:27 in the video): "My argument for liking version 2, and I still think version 2 is a great license, was that 'I give you source code, you give me your changes back, and we're even.' Right? That's my take on GPL version 2. It's that simple.”
  - (at 5:30): "I don't like locked down hardware, but at the same time that was never the social contract I intended with Linux.“
  - (at 8:10): “You don't have to use Linux. If you do use Linux the only thing I ask for is source code back. And then there's all the other verbiage in the GPL version 2 about exact details and those aren't important. And that was always my standpoint."

Linus Torvalds about source reciprocity (only)

- from https://lkml.org/lkml/2007/6/14/259
- "The BSD license is not doing tit-for-tat. It doesn't give me anything back. I don't believe in that kind of model. So I'd not use it for my projects.”
- “The GPLv2 has a good balance. It encourages tit-for-tat, and it makes sure that the software is kept free. And it doesn't try to force anything else, or play politics. The only thing you have to believe in is "tit-for-tat".
- “The GPLv3 goes too far. It's no longer "tit-for-tat", it's "our software is worth _soo_ much, that we want to force you to behave well, or you cannot use it"

Linus Torvalds on modifying a particular device

- From https://lkml.org/lkml/2007/6/13/390
  - "But I think the whole thing is totally misguided, because the fact is, the GPLv2 doesn't talk about "in place" or "on the same hardware"."
- From https://lkml.org/lkml/2007/6/15/336
  - "So clearly, the whole "modify in place" argument is simply *wrong*. It cannot *possibly* be a valid reading of the GPLv2! When the GPLv2 talks about "legal permissions to copy, distribute and/or modify" the software, it does *not* mean that you have to have the ability to modify it in place!"

Richard Stallman on GPLv2 and requiring device unlocking

- From See http://gplv3.fsf.org/wiki/index.php/FAQ_Update#How_does_the_.22any_later_version.22_clause_change_when_GPL_version_3_is_released.3F (quote extracted on 2020-10-09)
  - “As a user, I cannot demand that TiVo give me the authorization codes if they ship software that was released under the "GPLv2 or later" license. It "would be totally evil," Richard Stallman said, for a Free Software license to create new requirements for TiVo as time goes on. TiVo chose to distribute the code under GPLv2, and so they are not bound by the new requirement [in GPLv3].”

Richard Stallman on Free Software in ROM

- From https://www.fsf.org/news/freebios.html
  - "The ethical issues of free software arise because users obtain programs and install them in computers; they don't really apply to hidden embedded computers, or the BIOS burned in a ROM, or the microcode inside a processor chip, or the firmware that is wired into a processor in an I/O device. In aspects that relate to their design, those things are software; but as regards copying and modification, they may as well be hardware. The BIOS in ROM was, indeed, not a problem."

SFC on GPL enforcement work

- From https://sfconservancy.org/news/2020/oct/01/new-copyleft-strategy-launched-with-ARDC-grant/
  - “Our new initiative features:
    - Litigation to enforce against license violators that do not voluntarily comply in a timely manner
    - Coordinating the development of alternative firmware for devices where none currently exists
    - …”

SONY

Notes on Patrick McCardy litigation in Germany

- https://www.zdnet.com/article/linux-beats-internal-legal-threat/
  - By Steven J. Vaughan-Nichols, ZDNet
- https://opensource.com/article/17/8/patrick-mchardy-and-copyright-profiteering
  - By Heather Meeker
- http://kroah.com/log/blog/2017/10/16/linux-kernel-community-enforcement-statement/
  - by Greg Kroah-Hartman, Leading Linux kernel developer
- https://laforge.gnumonks.org/blog/20180307-mchardy-gpl/
  - by Harald Welte
- https://lwn.net/Articles/721458/
  - by Jake Edge, LWN.net

Makefiles are install scripts

- In 1991, the default build environment was self-hosted
  - You built and installed software on the machine it was going to run on
    - There were exceptions for cross-building, of course
  - Autoconf, automake, Makefiles were made to deal with difference in development environments and targets
  - By default, “make install” was the install mechanism – to the local machine
- Mid-90’s – the rise of the packager
  - debs and rpms (ipkg in embedded world)
  - Source rpm of a package fulfills licence for that package.
    - rpm tool is out of license scope.
- ‘make install’ (and ‘make modules_install’ for the kernel)
  - Create an interface between the package install script and the system install mechanism
- Kernel is different, since it always required different install mechanisms
  - Even here, there is a standard: /boot and /lib/modules/{uname –r}/...
- Don’t conflate a package install with a system install mechanism