Our GPL Compliance Work

- Stand Up For GPL!
- VMware Lawsuit: Press Release
- **FAQ on VMware Lawsuit**
- About GPL Compliance Project for Linux Developers
- copyleft.org

Frequently Asked Questions about Christoph Hellwig's VMware Lawsuit

Conservancy maintains this FAQ list regarding Christoph Hellwig's lawsuit against VMware in Germany over alleged GPL violations on Linux as a service to the Free Software community, and in particular, the copyleft community. Conservancy realizes this lawsuit generates many questions and interest from the community. Legal counsel (both Conservancy's own, and Christoph's lawyer, Till Jaeger) correctly advise us to limit our public comments regarding specific details of the case while litigation remains pending in court. Nevertheless, Conservancy, as a non-profit charity serving the public good, seeks to be as transparent as possible. If you have additional questions you'd like to see answered here, please email <info@sfconservancy.org>, but understand that we may often need to answer: "We cannot comment on this while litigation is pending".

Who is the Plaintiff in the lawsuit?

Christoph is one of most active developers of the Linux kernel. He has contributed 279,653 lines of code to the latest Linux 3.19 kernel, and thus ranks 20th among the 1,340 developers involved in that release. Christoph also ranks 4th among those who have reviewed third-party source code, and he has tirelessly corrected and commented on other developers' contributions.

Are the court documents released?

Not currently. Court proceedings are not public by default in Germany (unlike in the USA). Conservancy will continue to update this FAQ with information that Conservancy knows about the case. We would all also welcome an agreement with VMware whereby both sides would agree to publish all Court documents.

Who's funding this lawsuit?

Conservancy has engaged in a grant agreement with Christoph Hellwig for the purposes of pursuing this specific legal action in Germany. Conservancy is funding this legal action specifically as part of Conservancy's program activity in its GPL Compliance Project for Linux Developers.

Is this the Great Test Case of Combined / Derivative Works?

This case is specifically regarding a combined work that VMware allegedly created by combining their own code (“vmkernel”) with portions of Linux's code, which was licensed only under GPLv2. As such, this, to our knowledge, marks the first time an enforcement case is exclusively focused on this type of legal question relating to GPL. However, there are so many different ways to make combined and/or derivative works that are covered by GPL that no single case could possibly include all such issues.

Why must you file a lawsuit? Isn't there any other way to convince VMware to comply with GPL?

Neither Conservancy nor Christoph takes this action lightly nor without exhausting every other possible alternative first. This lawsuit is the outgrowth of years of effort to convince VMware to comply with GPL.

In October 2011, Conservancy received a GPL violation report on BusyBox for VMware's ESXi products. Conservancy opened the

matter in its usual, friendly, and non-confrontational way. Nevertheless, VMware immediately referred Conservancy to VMware's outside legal counsel in the USA, and Conservancy negotiated with VMware's legal counsel throughout late 2011, 2012 and 2013. We exchanged and reviewed CCS candidates, and admittedly, VMware made substantial and good efforts toward compliance on BusyBox. However, VMware still refused to fix a few minor and one major compliance problem that we discovered during the process. Namely, there was a major violation regarding Linux itself that ultimately became Christoph's key complaint in this lawsuit.

Meanwhile, when Conservancy realized in late 2012 there might be a major Linux violation still present in VMware's ESXi products, Conservancy representatives sought every industry contact we had for assistance, including those from trade associations, companies (both competitors and collaborators with VMware), and everyone else we could think of who might be able to help us proceed with friendly negotiations that would achieve compliance. While we cannot name publicly the people we asked for help to convince VMware to comply, they include some of the most notable executives, diplomats, and engineering managers in the Linux community. No one was able to assist Conservancy in convincing VMware to comply with the GPL. Then, in early 2014, VMware's outside legal counsel in the USA finally took a clear and hard line with Conservancy stating that they would not comply with the GPL on Linux and argued (in our view, incorrectly) that they were already in compliance.

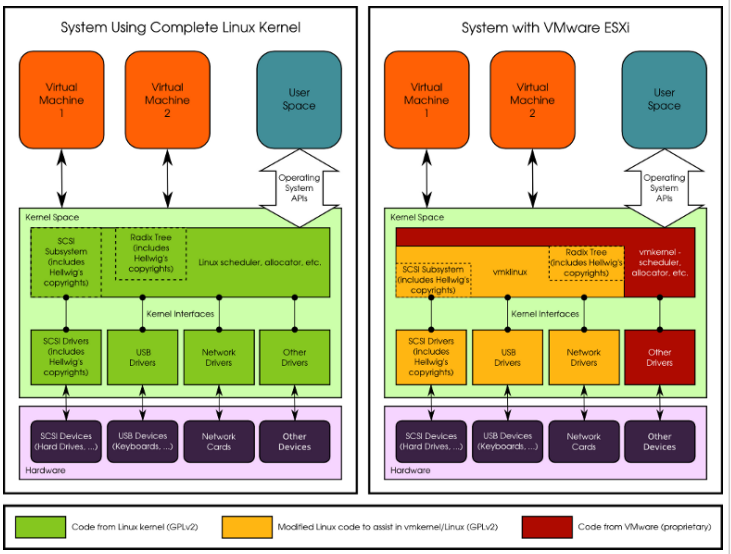
Conservancy in parallel informed Christoph fully of the details of the Linux violation on Christoph's copyrights, and based on Conservancy's findings, Christoph began his own investigation and confirmed Conservancy's compliance conclusions. Christoph then began his own enforcement effort with legal representation from Till Jaeger. Christoph has been unable to achieve compliance, either, through his negotiations in 2014. VMware's last offer was a proposal for a settlement agreement that VMware would only provide if Christoph signed an NDA, and Christoph chose (quite reasonably) not to sign an NDA merely to look at the settlement offer.

Thus, this lawsuit comes after years of negotiations by Conservancy to achieve compliance — negotiations that ended in an outright refusal by VMware's lawyers to comply. Those events were then followed by a year of work by Christoph and Till to achieve compliance in a separate action.

Simply put, Conservancy and Christoph fully exhausted every possible non-litigation strategy and tactic to convince VMware to do the right thing before filing this litigation.

Can you explain further how VMware incorporated code from Linux into their kernel?

Conservancy prepared this diagram to show the technical situation as we understand it. The diagram compares the technical architecture of a full, running Linux kernel with a full, running VMware kernel:



If you want to download the diagram, it's available in SVG (English), PNG (English), SVG (German), and PNG (German).

Can you explain further in words (rather than a picture) about the central component in ESXi that the lawsuit alleges violates the GPL?

The GPL violation at issue involves VMware's ESXi product. Conservancy independently reviewed ESXi 5.5 and its incomplete CCS release as part of our GPL enforcement efforts described above.

Conservancy's preliminary investigation indicated that the operating system kernel of VMware ESXi product consists of three key components:

- the proprietary component “vmkernel”, which is released in binary form only,
- the kernel module “vmklinux”, which contains modified Linux Code, and for which (at least some) source code is provided.
- other kernel modules with device drivers, most of which are modified Linux drivers, and for which (at least some) source code is provided.

Conservancy examined the incomplete CCS alongside the binary “vmkernel” component. Such examination indicates that functions in “vmkernel” do make function calls to Linux's kernel code in the usual way for a single program written in C.

Doesn't VMware's “shim layer” insulate them from GPL obligations and allow them to keep certain code in their kernel proprietary?

Many in the media have talked about the possibility that VMware might use some so-called “shim layer” between Linux code and VMware's proprietary code. While, for decades, there has been much talk of various mechanisms of GPL obligation avoidance, Conservancy believes that merely modifying technical details of a combination's construction does not typically influence the legal analysis in a combined or derivative work scenario.

Furthermore, the technical details of VMware's alleged GPL violation do not even mirror the typical scenarios that have usually been called “shim layers”. Conservancy's analysis of VMware's ESXi product, in fact, indicates that VMware rather flagrantly combined Linux code in their own kernel, and evidence seems to indicate the work as a whole was developed by modifying Linux code in tandem with modifications to “vmkernel” in a tightly coupled manner.

Is Conservancy proposing a “shim layer” as a viable solution for GPL compliance?

No, in fact, as we say above, Conservancy doesn't think the phrase “shim layer” has any meaning, despite regular use of that phrase in the media. Conservancy generally doubts there is any technological manipulation that changes the outcome of a combined/derivative work analysis.

Can you give a *specific* example, with code, showing how VMware combined Linux source code with their binary-only components?

There are numerous examples available that show this. The details of alleged infringement specifically relating to Hellwig's contributions to Linux are of course the main matter of the allegations in the litigation, and Conservancy released the diagram above to exemplify that issue. Conservancy continues to hope VMware will agree to make public all court documents as a matter of public good, since the court documents discuss the specifics of alleged infringement on Hellwig's copyrights.

However, Conservancy examined VMware's ESXi 5.5 product in detail even before Hellwig's enforcement action began. Below is one example among many where VMware's CCS was incomplete per GPLv2§2(c) and GPLv2§3(a). (One can verify these results by downloading and installing the binary and source packages for

VMware's ESXi 5.5 Update 2.) Note that this example below is not necessarily regarding Hellwig's copyrights; VMware incorporated Linux code copyrighted by many others as well into their kernel.

Example of “vmkernel”'s combination with Linux code

Our example begins with examination of the file called `vmkdrivers/src_92/vmklinux_92/vmware/linux_pci.c`, which can be found in the “Open Source” release for ESXi 5.5.0 Update 2 (5.5U2). A small excerpt from that file, found in the function `LinuxPCIDeviceRemoved()`, reads as follows:

```
#include <linux/pci.h>
[...]
/*
 * This function [...] is modelled after pci_remove_device, the 1
 * be called in a linux system.
 */
static void
LinuxPCIDeviceRemoved(vmk_PCIDevice vmkDev)
{
   LinuxPCIDevExt *pciDevExt;
   struct pci_dev *linuxDev;
[...]
  if (unlikely(
    vmk_PCIGetDeviceName(vmkDev, vmkDevName, sizeof(vmkDevName)-1
  {
      vmkDevName[0] = 0;
  }
[...]
VMKAPI_MODULE_CALL_VOID(pciDevExt->moduleID,
                        linuxDev->driver->remove,
                        linuxDev);
```

Combination of “vmkernel” code with “vmkdrivers”

The function, `vmk_PCIGetDeviceName()` must be defined, with an implementation, for this code above to work, or even compile. Inside `BLD/build/HEADERS/vmkapi-current-all-public/vmkernel64/release/device/vmkapi_pci_incompat.h`, found in the `vmkdrivers` package of ESXi 5.5U2, shows a function header definition for `vmk_PCIGetDeviceName()`. However, the source of its implementation is not provided there or anywhere in the source release.

Further evidence that the implementation of this function occurs elsewhere can by found by running `objdump -x` on the un-vmtar'ed `vmklinux_9` module. Note the following output in the “SYMBOL TABLE” section:

```
0000000000000000         *UND*  0000000000000000 vmk_PCIGetDevic
```

…and the following lines found in the “RELOCATION RECORDS FOR [.text]” section:

```
00000000000327ff R_X86_64_PC32     vmk_PCIGetDeviceName+0xffffff
0000000000035318 R_X86_64_PC32     vmk_PCIGetDeviceName+0xffffff
00000000000387e1 R_X86_64_PC32     vmk_PCIGetDeviceName+0xffffff
000000000003cf40 R_X86_64_PC32     vmk_PCIGetDeviceName+0xffffff
```

The above two properties both suggest that the `vmklinux_9` module requires: (a) a definition of the `vmk_PCIGetDeviceName()` function to operate, but (b) that function is not defined inside `vmklinux_9` itself.

The definition can however be found in binary-only software provided in ESXi 5.5U2 — specifically, inside a file named `k.b00`, which is located in partition 5 on a disk where ESXi has been installed (or in the ESXi 5.5U2 installer ISO image). Running `file` after `gunzip` on this file yields “ELF 64-bit LSB shared object”. Meanwhile, `file k.b00` reports “gzip compressed data, was ‘vmvisor64-vmkernel.stripped’”. These findings strongly suggests this is an image of the “vmkernel” component. An `objdump -x` yields this “SYMBOL TABLE” section:

```
000041800036a408 g     F .text  0000000000000137 vmk_PCIGetDevic
```

… which indicated these binary file contains the function body for `vmk_PCIGetDeviceName`.

Furthermore, after detailed searching, Conservancy found no evidence that any other code (other than modified Linux code) makes calls to `vmk_PCIGetDeviceName`. This provides a strong indication that this function's primary purpose is to combine Linux code with

"vmkernel". Conservancy also found other functions where similar analysis yields similar results as above.

Linux's `struct pci` combined with `LinuxPCIDeviceRemoved()`

Having established the direct and close combination of `vmk_PCIGetDeviceName` and `LinuxPCIDeviceRemoved()`, focus now on the quoted code from `LinuxPCIDeviceRemoved()`. That code, note that one of the local variables is `struct pci_dev *linuxDev;`. A definition of `pci_dev` is found in `vmkdrivers/src_92/include/linux/pci.h` (which is `#include`'d above) reads:

```
struct pci_dev {
[...]
       struct pci_driver *driver;      /* which driver has alloca
[...]
truct pci_driver {
         char *name;
[...]
        void (*remove) (struct pci_dev *dev);   /* Device removed
[...]
#if defined(__VMKLNX__)
        /* 2008: Update from Linux source */
        u8              revision;       /* PCI revision, low byte
#endif /* defined(__VMKLNX__) */
  };
```

These structures, and based on those from Linux itself (a similar version of this file can be seen in Linux 2.6.24), and as can be seen above, have been modified to work with "vmkernel"

In `LinuxPCIDeviceRemoved()`, we saw a macro called with a variable, `linuxDev` which was of type `struct pci`. Thus, the combination of code from Linux's `pci.h` and VMware's `vmware/linux_pci.c` is very tightly coupled and interdependent.

`VMKAPI_MODULE_CALL_VOID` macro calls driver's code

The file `BLD/build/HEADERS/vmkapi-current-all-public/vmkernel64/release/base/vmkapi_module.h` contains the macro definition of `VMKAPI_MODULE_CALL_VOID`, which is quoted below (with debug lines removed):

```
#define VMKAPI_MODULE_CALL_VOID(moduleID, function, args...)  \
do {                                                          \
    vmk_ModInfoStack modStack;                                \
    vmk_ModulePushId(moduleID, function, &modStack);          \
    (function)(args);                                         \
    )                                                         \
    vmk_ModulePopId();                                        \
} while(0)
```

When the macro is expanded, it means that `(function)(args)` is actually expanded to `linuxDev->driver->remove(linuxDev)`. Therefore, we see `LinuxPCIDeviceRemoved()`, makes directs calls to a driver's remove() function, by combining with Linux's `struct pci`, and by VMware's introduction of this new calling code. Conservancy has confirmed many drivers from Linux are incorporated via these mechanisms; one specific example is discussed next.

Combination of the tg3 driver with "vmkernel"

VMware includes a file `vmkdrivers/src_9/drivers/net/tg3/tg3.c` in their source release. This file appears to be Linux's tg3 driver. It includes a definition of the `struct pci_dev` for this device, which reads:

```
static struct pci_driver tg3_driver = {
[...]
        .remove         = __devexit_p(tg3_remove_one),
```

Therefore, when the code in `LinuxPCIDeviceRemoved()` calls `linuxDev->driver->remove(linuxDev)`, the code ultimately called (in the case where a tg3 card is driven by the kernel) is `tg3_remove_one()`, which is found in `tg3.c` and comes directly from Linux.

(Note: `__devexit_p` is a straightforward macro found in `vmkdrivers/src_92/include/linux/init.h` (which also comes from Linux) that will simply expand to its first argument in this case.)

VMware distribution of binary version of `tg3.c`

VMware furthermore distributes a modified version of `tg.c` in binary

form. This can be found in `usr/lib/vmware/vmkmod/tg3`, which is extracted by un-vmtar'ing the file `net_tg3.v00` (found on the ESXi 5.5U2 installer ISO image). Conservancy has confirmed that file is a compiled version of `tg3.c`

Conclusions

Given this evidence and related contextual clues, the only logical conclusions are:

- `vmklinux_9`, a binary object, dynamically links with the binary objects: `k.b00` and `tg3` (the driver built from `tg3.c`'s source). These three binary objects together form a single running binary (likely along with many other binary objects as well).
- That single running binary contains code licensed under the GPLv2 — namely the code derived from `tg3.c` and `pci.h`. Thus, the single running binary may be distributed in binary form only under permissions provided under GPLv2 — in particular GPLv2§2 and GPLv2§3.
- GPLv2§3(a–b) requires that "complete corresponding machine-readable source code" must accompany binary distributions such as these. GPLv2§3 further states that "for an executable work, complete source code means all the source code for all modules it contains".
- The binary work in question contains modules from `k.b00`, `vmlinux_9` and `tg3`.
- VMware did not provide source code for any modules found in `k.b00`.
- Therefore, VMware failed to comply with the GPLv2, as such compliance requires source code (or an offer therefor) for the material in `k.b00`.

The above is but one piece of evidence among many, but hopefully it helps to explain some of the “combined work” violations found in VMware's ESXi product.

How can I verify Conservancy's technical findings above?

The binary and source packages mentioned above are available on VMware's website. These packages contain the previously-mentioned `linux_pci.c`, `vmkapi_pci_incompat.h`, and `k.b00` files, as well as `vmklinux_9` and the source code that builds the latter.

To obtain the source components, follow these steps (no login is required):

1. Visit https://my.vmware.com/web/vmware/details?downloadGroup=ESXI55U2_OSS&productId=353.
2. Click the “Download” button beside the text that reads “Open Source Code for VMware vSphere ESXi 5.5 Update 2”.
3. Confirm that the SHA-1 hash matches the published one (d121634668a137ec808b63679fd941cef9a59715), found under “Read More” on that web page.
4. Mount (or otherwise open) the downloaded `VMware-ESX-550U2-ODP.iso`.
5. Extract `vmkdrivers/src_92/vmklinux_92/vmware/linux_pci.c` and `BLD/build/HEADERS/vmkapi-current-all-public/vmkernel64/release/device/vmkapi_pci_incompat.h` from `vmkdrivers-gpl/vmkdrivers-gpl.tgz` with tar and gzip.
6. Generate `vmklinux_9` by following the steps in `vmkdrivers-gpl/BUILD.txt` in the ISO. (Note: `vmklinux_9` is also available pre-built on a running ESXi system; see below for instructions on how to access it).
7. You may need the “Supporting Toolchain packages for VMware vSphere ESXi 5.5.0 Update 2” file from the above download page to complete the build — upon downloading you will find it is named `VMware-TOOLCHAIN-550u2-ODP.iso` and has a SHA-1 hash of f679e81ffb2f92729917bbc64c2d541cf75b5b94.

To obtain the binary components, follow these steps (a login is required):

1. Register for an account at https://my.vmware.com/web/vmware/registration.
2. Click the “Activate Now” link in the follow-up email. Enter the password used at registration time. Click “Continue”.

3. Visit https://my.vmware.com/web/vmware/evalcenter?p=free-esxi5.
4. Click “Register” (under the text that reads “You have not registered for this product”).
5. Enter the number of servers you plan to install on (e.g., 1). Click “Continue”.
6. If the “VMware vSphere Hypervisor 5.5 Update 2 – Binaries” section is not expanded, click the plus sign next to it.
7. Click the “Manually Download” link that's beside “ESXi 5.5 Update 2 ISO image (Includes VMware Tools)”.
8. Confirm that the SHA-1 hash matches the published one (9475938b51cafc86c8b17d09f2493cb6b4fae927).
9. Mount (or open via some other means) the downloaded `VMware-VMvisor-Installer-5.5.0.update02-2068190.x86_64.iso`.
10. Find the `k.b00` file in the root directory. Extract it using `zcat k.b00 > vmvisor64-vmkernel` (or a similar command). Repeat the steps described above using `objdump -x vmvisor64-vmkernel`.
11. To retrieve `vmklinux_9` you will need to install ESXi on your system by booting the ISO and following the instructions. Once booted, you can then enable SSH access using “Customize System/View Logs -> Troubleshooting Options -> Enable SSH”. Login to the system with SSH and then run `find /vmfs -name misc_dri.v00 -print`. On the resulting file, run `zcat misc_dri.v00 > misc_dri.vmtar` then `vmtar -x misc_dri.vmtar -o misc_dri.tar`. You can then extract `misc_dri.tar` using the usual `tar` to extract `usr/lib/vmware/vmkmod/vmklinux_9`. The `misc_dri.v00` file is also available next to `k.b00` in the root directory of the ISO (mentioned above), but the `vmtar` command itself is only available when logged into an ESXi system. `vmtar` can be found at `bin/vmtar` inside `sb.v00` on the ISO, but one needs `vmtar` to open `sb.v00`, similar to `misc_dri.v00` above.

Note that VMware may present you with EULAs and ToS when you download software from VMware's website. Conservancy strongly suggests that you review these terms in great detail with the assistance of your own legal counsel before downloading the software and/or engaging in the process that Conservancy discusses above.

Have others issued statements of support about this action?

Various individuals and groups have publicly stated their support for Conservancy's and Hellwig's actions in this matter. They include:

- APRIL
- Free Software Foundation
- Free Software Foundation Europe
- GNOME Foundation
- Open Source Initiative
- The Samba Team
- The SWIG Project
- Dave Airlie, Linux Developer
- Matthew Garrett, Linux Developer
- Grant Likely, Linux Kernel Engineer
- Michal Nazarewicz, Linux Developer
- Luis R. Rodriguez (aka mcgrof), Linux Developer
- Wolfram Sang, Linux Developer
- Josh Triplett, Linux Developer
- Rik van Riel, Linux Developer

I see FSF's statement of support, but why isn't FSF enforcing in this case?

While FSF are the authors and license steward of the GNU GPL, it's up to the copyright holder to enforce GPL. VMware created an operating system by combining parts of the kernel named Linux with their own proprietary code, and then added BusyBox to provide the userspace operating system components. As such, ESXi is not a traditional GNU/Linux system. FSF has many copyrights of its own, but these are almost exclusively on various parts of the GNU system, not on the kernel, Linux. As such, FSF probably does not have copyright interests available to directly enforce the GPL regarding the primary issue in this case.

I care about copyleft and the GPL. How can I help?

Conservancy needs your immediate financial support to proceed with

this litigation. Litigation costs are unpredictable, and this lawsuit may take years to resolve. Conservancy is prepared to fund this case through its conclusion, but we can only do so with *your* support. If you are an individual who supports copyleft and wants to see it defended, please donate now. And, if you make a public statement of support, please email the URL to <info@sfconservancy.org>, as we'd like to include representative selection of supportive statements above.

Why is the case in Germany?

Copyright infringement claims can be brought anywhere that distribution of the copyrighted works occur. VMware distributes ESXi throughout the world, but Germany is close to Christoph's home and his lawyer was available to do the litigation work there. Finally, historically, Mr. Jaeger's cases in Germany have usually achieved worldwide compliance on the products at issue in those cases.
