acmqueue Quality Software Costs Money— Heartbleed Was Free

How to generate funding for FOSS

Poul-Henning Kamp

The world runs on free and open-source software, FOSS for short, and to some degree it has predictably infiltrated just about any software-based product anywhere in the world.

What's not to like about FOSS? Ready-to-run source code, ready to download, no license payments—just take it and run. There may be some fine print in the license to comply with but nothing too onerous or burdensome.

Your TV? It has a Linux or {Net|Free}BSD computer inside it. So does the copier-printer/multifunction machine at the office, as do the entertainment consoles in your car and in your kids' bedrooms. Code reuse has finally happened, but there is a footnote: you get what you pay for.

Earlier this year the OpenSSL Heartbleed bug laid waste to Internet security, and there are still hundreds of thousands of embedded devices of all kinds—probably your television among them—that have not been and will not ever be software-upgraded to fix it. The best way to prevent that from happening again is to avoid having bugs of that kind go undiscovered for several years, and the only way to avoid that is to have competent people paying attention to the software.

In the case of OpenSSL, it is painfully obvious that nobody was paying attention. There is a commercial entity called The OpenSSL Foundation, but as far as anyone can determine, it is a FIPS (Federal Information Processing Standard)-test consultancy and that is pretty much all it does.

According to an article in the *Wall Street Journal*,⁵ the OpenSSL Foundation received around \$2,000 a year in donations to maintain the OpenSSL code. That amounts to approximately 20 hours of attention per year from a good programmer to maintain north of a half-million lines of code. That certainly explains why the OpenSSL bug tracker was write-only and why the code is still overcomplicated by support for vintage operating systems such as VMS and 16-bit Windows.

Predictably, money started pouring in after the Heartbleed bug caused havoc: IT security is a game you can win only by losing first. So OpenSSL maintenance, testing, and quality assurance is funded now, and we're good, right? No, we are not! Many other FOSS projects (in addition to OpenSSL) are badly understaffed because they are underfunded—and we need to fix that.

WHY FOSS NEEDS MONEY

About the only thing GNU Project founder Richard Stallman and I can agree on when it comes to software freedom is that it's "Free as in free speech, not free beer."

I really hope the Heartbleed vulnerability helps bring home the message to other communities that FOSS does not materialize out of empty space; it is written by people. We love what we do, which is why I'm sitting here, way past midnight on a Saturday evening, writing about it; but we are also real people with kids, cars, mortgages, leaky roofs, sick pets, infirm parents, and all kinds of other perfectly normal worries.

The only way to improve the quality of FOSS is to make it possible for these perfectly normal

people to spend time on it. They need time to review patch submissions carefully, to write and run test cases, to respond to and fix bug reports, to code, and most of all, time just to think about the code and what should happen to it.

It would not even be close to morally defensible to ask these people to forgo time to play with their kids or walk their dogs in order to develop and maintain the software that drives the profit in other people's companies. The right way to go—the moral way to go—and by far the most productive way to go is to pay the developers so they can make a living from the software they love. And not just for a month or two: good programmers, like most everyone else, prefer stable jobs so they can concentrate on the job rather than wasting time chasing the next gig or the next sponsor.

HOW TO FUND FOSS

One way to fund FOSS is simply to hire the FOSS maintainers, with the understanding that they spend some amount of company time and resources on the project. Experience has shown that these people almost invariably have highly desirable brains, which employers love to throw at all sorts of interesting problems, and that tends to erode the “donated” company time. A lot of FOSS has been, and still is, developed and maintained this way, with or without written agreements or even company knowledge of this being the case.

A big thank you to those employers who do this deliberately!

These companies should add their support of FOSS to their lists of corporate social responsibilities, along with their sponsorships of local soccer teams and their funding of scholarships. They are doing something for the greater common good, which they should be proud of and get proper credit for.

Another way to fund FOSS is for software projects to set up foundations to collect money and hire developers. This is a relatively complex and expensive undertaking. You run straight into the murkiest corners of tax codes, so this approach is available only for larger projects. While several projects have gone this route, their ability to raise money varies significantly.

The Apache Foundation is probably the largest of all these FOSS foundations. A few entities “adopt” smaller projects inside their fields of interest. This generally works OK but cannot easily be generalized to different areas.

An easier and cheaper way, the one advocated here, is simply to throw money at the developers, the way the FreeBSD and Varnish communities have done with me. Forget about tax deductions and all that; simply have the developer send your company an invoice every so often and stuff it into some account in the IT department labeled “misc. software licenses” or “expert assistance” or whatever will fly under the radar in your company. It takes only a dozen companies worldwide to keep a top-notch programmer's nose in the code of a FOSS project full time—and this is probably code that is likely a lot more critical to a company's operations than anybody really likes to think about.

Here's my story...

FREEBSD COMMUNITY FUNDING

My first crowdfunding of FOSS was in 2004 when I solicited the FreeBSD community for money³ so that I could devote three to six months of my time to the FreeBSD disk-I/O subsystem.

At the time I had spent 10 years as one of the central and key FreeBSD developers, so there were no questions about my ability or suitability for the task at hand. In 2004, however, crowdfunding

was not yet “in,” and I had to figure out how to do it myself. My parents raised me to believe that finances are a private matter, but I concluded that the only way you could reasonably ask strangers to throw money at you would be to run an open book where they could see what happened to the money. So I opened my books.

The next dilemma was my rate. Again, I had always perceived my rate to be a private matter between me and my customers. Because my rate is about half of what most people expect (as I won’t work for most projects but only on things I really *care* about), I was concerned that publishing my rate would undercut friends and colleagues in the FreeBSD project who made a living consulting. There was no way around it, however, so I published my rate, making every attempt to distinguish it from a regular consulting rate, and I never heard any complaints.

Having agonized over the exact text and then tested it on a couple of close friends in the FreeBSD project, I threw the proposal out there and waited. I had a perfectly safe fallback plan—you have to when you have two kids and a mortgage—but I really had no idea what would happen next.

Worst case, I would cause the mother of all bikesheds (<http://bikeshed.org/>²) to get thrown out of the FreeBSD project, and I would be widely denounced for my “ideological impurity” with respect to FOSS. Best case, I expected to get maybe one or two months funded.

The FreeBSD community responded overwhelmingly. My company has never sent as many invoices as it did in 2004, and my accountant nearly blew a fuse. Suddenly I found myself in a situation I had never even considered: how to stop people from sending me money. I had set up a PayPal account, and at least at that time, there was no way to prevent people from dropping money into it. In the end, I managed to yell loud enough, so I was overfunded by only a few percent. I believe that my attempt to deflect the surplus to the FreeBSD Foundation gave that group a little boost that year, too.

Today people who do crowdfunding have “stretch goals” to soak up overfunding, but in 2004, I was in uncharted waters; whatever happened, I wanted to contain any fallout from my experiment in a single fiscal year. I also wasn’t sure how it would work out in terms of the social dynamics of the project, so I didn’t want it to extend past half a year.

In total, I made 27,000 euros, which kept my kids fed and my bank happy for the six months that I worked on the project.

And work I did.

I’ve never had a harsher boss than during those six months, and it surprised me how much it stressed me. I felt like I was working on a stage with the entire FreeBSD project in the audience wondering if I would deliver the goods or not. As a result, the 187 donors certainly got their money’s worth, as most of that half year I worked 80-hour weeks, which made me decide not to continue, despite many donors indicating that they were perfectly willing to fund several more months.

VARNISH COMMUNITY FUNDING

Five years later, having developed Varnish HTTP Cache Version 1.0 for Norway’s *Verdens Gang* newspaper, and having exploded into a vacuum in Web-content delivery, I decided to give community funding a go again.

Wiser from experience, I structured the VML (Varnish Moral License)⁴ to tackle the issues that had caused me grief the first time around: contact first, then send money, not the other way around; also focus on fewer, larger sponsors, rather than individuals sending me 10 or 15 euros, or even in one

case one euro, which lingered in an unused PayPal account. I run even more-open books this time, and on the VML Web pages you can see how many hours I have worked, along with a terse one-line description of what I did for every single day I've been working under the VML since 2010.

I also decided to be honest with myself and my donors: one hour of work was one hour of work—nobody would benefit from me dying from stress. In practice it doesn't quite work like that: there is plenty of thinking in the shower, as well as e-mails and IRC (Internet Relay Chat) answers at all hours of the day and night, and a lot of "just checking a detail" that happens off the clock because I like my job, and nothing could stop me anyway.

This is the funding model I want to "sell" for other FOSS projects, because it works. In each of the years 2010, 2011, and 2013 I worked around 950 hours on Varnish (funded by the community) in addition to work I did for my other customers. In 2012 I worked only 589 hours, because I was building a prototype computer cluster to do adaptive optics real-time calculations for the European Southern Observatory's Extremely Large Telescope¹ (there was just no way I could say no to that contract).

In 2014 I have hours available to do even more Varnish work, but despite my not-so-subtle hints, the current outlook is still for only 800 hours to be funded. I'm crossing my fingers that more sponsors will appear now that Varnish Version 4 has been released (nudge, nudge, wink, wink — He said knowingly).

The VML is not an ideal funding model in the sense that with a single exception, none of the big corporations that deliver massive amounts of HTTP traffic with Varnish has participated. A number of smaller concerns have kept the project alive and kicking. In a smaller company the CEO, or at least the CTO, is more likely to know what FOSS products the company uses, and quite likely keeps abreast of developments and announcements. In a larger organization, on the other hand, bigger issues drown out such "details" as a fundraising plea before they rise to a level where a decision can be made, or where the fear that the marketing department must be involved spikes the idea. I can vividly imagine how Dilbert would fail to convince his PHB (pointy-haired boss) that the company should give money away because it is using some free software.

The good news is that the PHB doesn't have to understand: if a dozen midsize companies each decide to spend \$500 every month, that would do wonders for any piece of FOSS—if the money makes it into the hands of the right person(s). And there's that little detail: finding the Right Person.

There is no way to shortcut that: each FOSS project, each sponsoring company, each potential maintainer will have to look one another in the eye and decide if they think this is worth a shot or not. There will be failures and disappointments—that is unavoidable—but if we do not fund good people to maintain critical FOSS projects, there will be no end to the Heartbleed.

REFERENCES

1. European Southern Observatory. The European Extremely Large Telescope; <http://www.eso.org/public/teles-instr/e-elt/>.
2. Kamp, P.-H. 1999. Why should I care what color the bikeshed is; <http://bikeshed.org/>.
3. Kamp, P.-H. 2004. Fundraising for FreeBSD development; <http://people.freebsd.org/~phk/funding.html>.
4. Kamp, P.-H. The Varnish Moral License; <http://phk.freebsd.dk/VML>.
5. Yadron, D. 2014. After Heartbleed bug, a race to plug Internet hole. *Wall Street Journal* (April 9);

<http://online.wsj.com/news/articles/SB10001424052702303873604579491350251315132> (login required).

LOVE IT, HATE IT? LET US KNOW

feedback@queue.acm.org

POUL-HENNING KAMP (phk@FreeBSD.org) is one of the primary developers of the FreeBSD operating system, which he has worked on from the very beginning. He is widely unknown for his MD5-based password scrambler, which protects the passwords on Cisco routers, Juniper routers, and Linux and BSD systems. Some people have noticed that he wrote a memory allocator, a device file system, and a disk-encryption method that is actually usable. Kamp lives in Denmark with his wife, son, daughter, about a dozen FreeBSD computers, and one of the world's most precise NTP (Network Time Protocol) clocks. He makes a living as an independent contractor doing all sorts of stuff with computers and networks.

© 2014 ACM 1542-7730/14/0600 \$10.00