How Not to Make Money in Open Source

Paul Brown / @paulrbrown
Multifarious, Inc.
Disclaimer

This is a negative talk in that it consists almost entirely of things that you should not do.

The idea for giving a negative talk was inspired by The Art of Demotivation by E.L. Kersten.
Problem Statement

You are an exhibitionistic but ascetic software engineer. You would like to create open source software but wish to avoid money and all of the hassles that might come with it.

More money => More problems
Risks

You may be exposed to any number of opportunities to make money with your project:

- commercial licensing
- support contracts
- consulting work
- sponsored development
- acqui-hire*
Attack Vectors

You must avoid some or all of:

- **Control and Ownership**: You can’t sell what you don’t own!
- **Completeness**: People like to buy things that solve their problems.
- **Digestibility**: People like to buy things they understand.

Let's talk defensive strategies.
Secret Bonus!

Many of these defensive techniques can be applied successfully even outside of open source!
Keep It Simple.

People are fundamentally lazy. If you don't give them a reason to call you, most of them won't.

Building small, high-quality, easily integrated components under business-friendly licenses is a good way to ensure silent appreciation.
Don't Offer Anything for Sale.

Thousand-dollar smart phone apps prove that some people will buy anything offered for sale on a web page, so don't offer anything for sale.

Simply not listing commercial offerings on your website can be enough to keep many people from asking and will help you avoid impulse buys from bored procurement officers.
Don't Have a Sales Process.

Even commercially successful open source projects may have a very low conversion rate from downloads to paying licenses (like 0.1% if they’re trying).

Failing to capture leads or being lazy with follow-up can ensure that your few potential customers slip right through the cracks.
Trick Other People Into Doing It.

If you develop useful software and a recognizable brand, you may find other people who are willing to profit from your software — even if they did absolutely nothing to create it.

You can even let them use your brand and your project mailing lists or IRC channel to promote their services. (Suckers!)
Give It All Away.

"Why buy the cow when you can get the milk for free?"

You can also give away the "cow" — make all of the enterprisey, complex features part of the freely available version.

You can foist paying customers onto other projects by charging for functionality that is available for free elsewhere.
Use a Weird or Whimsical License.

Corporate lawyers are relatively serious folks by nature, and using an offbeat license — even just adding a single line to a standard license — can be enough to throw them off.

Dual licensing and per-field-of-use licensing are also excellent choices.
Price Strangely.

Many commercially successful open source projects offer packaging and pricing similar to functionally related, closed source commercial software.

Choosing an offbeat pricing model can help keep customers from making comparisons.
Employment Agreements.

A well-written employment agreement can give your employer ownership of work you do in your off time, including your open source project.
Keep Your Source Messy.

Supporting software means that you have to be able to know when a bug is fixed and that no new bugs have been introduced.

Multiple committers with no release plan or branching scheme will help keep people guessing.

Infrequent releases and incomplete release notes also help.
Muddy Ownership.

Accepting contributions without contributor license agreements or similar structure can mean that you do not genuinely own the project past a point in time, ergo you cannot sell it or re-license it even by accident.
Decentralize Control.

“BDFL” sounds sexy, but so does plausible deniability.

Ceding control of your project to a collective can help ensure that you’re not able to align a feature-set or release schedule with customer requirements.
THANK YOU!