Chapter 6:

Damning with faint praise--

Take the right examples of free software and exploit them for everything

You might not believe me if I tell you that we decide what “cool” means, unless you look at the results in real life. We have the press working for us, we have open source doing what we need even as they believe they are fighting us. We can keep the press obedient in exchange for access, we can enjoy allegiance from open source from other deals that we make.

If our research shows that something is cool, we buy it or we produce it. One thing you can't do is make the same thing look cool forever. No matter how happy some customers are, or how much they rely on our software, we are still going to lead them away from one thing and towards another-- it isn't just because it's better; it's because if we let them rely on what they already have, we won't get to sell them anything new.

Now with cloudware, we control the access to the software. But it is still necessary for people to perceive that we are working to make improvements. The most convincing way to do this is to keep chasing what's new-- and dragging people from one thing to the next.

Once again we do this as friends, with a carrot of features and not just the stick of walled access and fees. We aren't just forcing people to pay, but leading them to want their subscriptions and feel good about them. We make promises about the new features being useful, but we also need the press and our marketing to make our products cool. No matter how much of a monopoly we work to maintain, there are other companies out there looking to create something before we can. We need to be sure we own and control it-- otherwise, it belongs to them.

Another great thing about what's new is that people don't know anything about it. We can write our own narrative about new products, and until enough people gain access and knowledge (and preferably first-hand experience) they can't say very much about the product that we can say twice as much about. New products let us run circles around people who claim to know about them-- they give us a unique advantage.

As for lack of familiarity, we can actually use that to increase interest rather than lose business. If the product is attractive enough, we can use the people who are familiar with it against the people who aren't, and make the latter look like they

are ignorant if they are unfamiliar with our product line. So the incentive to do business is to continue to appear knowledgeable. This tactic works notably well among enthusiasts and professionals.

If you ask people to define “cool,” they can't always come up with something concrete. But you don't have to be a cool person at all to know what definition of “cool” matters to us-- “cool” is what's new and what people want. That's as cool as we will ever need to be. That's what keeps us where we are, and everyone else where they are. And that's what we have to keep control of, if we want to stay where we are.

On the other side of the coin we have what's not cool: stuff that's not new, *or* stuff that people don't want. That's what we have to avoid offering-- if that's all we have, we need to buy or develop new things to offer.

A superior product is like a politician's speech-- the best way to sell a lie is to put a truth in it, so people assume the rest of it is also the truth. And when you want to sell a new product you can do the same thing: start with a feature people are desperate to have, and you can build a lot of garbage around it as long as the important features are satisfying enough.

It should be more than obvious that some of these features-- even some of the best features-- are going to be proprietary. So it becomes imperative if we are going to compete with and also infiltrate open source that we need to loosen the hold that free software has on the narrative.

A schism can be hashed out and resolved-- what we want is to widen it to a chasm and actually hand the reins of free software over to open source, so that all “open source” is forever a way to steer people towards our features.

Everything cool (that we care about) is new and wanted, and everything uncool is old and boring and standard. To keep churning out cooler, newer products (not always cool like Apple, sometimes just cool like an update with a few new features) we need to use our shills to show everyone how uncool the alternatives are. As long as we look cool and friendly, people will be reluctant to care that we need to go after our competition in this way. After all, cool is also about winning, and winning means someone loses somewhere.

By the time people are convinced that our competitors aren't cool, they won't want to side with them anymore and won't defend them from us.

We need to turn rich against poor, as mentioned two chapters ago-- and we need to turn inexperienced users against experienced ones, to prevent skeptics from handing down their stories to potential customers. It is essential to paint seasoned experts as gray and irrelevant, as has-beens who don't understand the genius of our new offerings. And it is essential to paint every tool we used to offer or never offered as outdated and obsolete.

When dealing with open source, the most important person to paint as a has-been is Richard Stallman. He and Gates are decades-long rivals, diametrically opposed to each other in their philosophies. But more relevant is that he leads the movement that opposes us-- we need to keep open source on our side, and lead them further away from free software.

Fortunately, Stallman and his followers are tightly-knit in their ideology. Attacking any of them is like attacking all of them-- we can play up their hacker style as social ineptitude, their adherence (where it exists) to standards and interoperability as a refusal to evolve, their playful culture as a refusal to grow up and be professional, and their self-reliance and independence as being non-team-players and even toxic masculinity.

Their hacker philosophy is about putting certain values first-- just as we use new features to get people to accept new flaws that we can promise to fix later (and then say that we have a greater commitment to security) and use open source to bring people to our exclusive software lines, we can use their values to steer the next generation of customers (and critics) towards a more corporate culture.

Any social values that we are saddled with keeping up appearances about in the workplace, we can instill through open source and then claim the rest are not putting enough emphasis on. Of course, some of these values are good values in and of themselves. But as much as we have “social value theater” in the workplace and have to play along, we can dump the same corporate culture onto anyone who will call it professionalism, and then say everyone else is just unprofessional and toxic.

In the short run we can use this against Stallman and his organization, but in the long run we can even use this to shackle Linus and gradually push him out the door. In our culture, it doesn't pay to be eccentric except when it makes us billions-- get with the program or get out. A leader that isn't making us money is a leader who has let us down, and we need to get rid of them as quickly as possible.

The values of free software developers and the values of free software itself go hand in hand. We need to denigrate most of their software, along the same lines that we denigrate the people who create it. As said over and over, we need to rely on shills, fans, and “useful third parties” to denigrate these people and their work. We also need a “path forward” to our products. Whenever we outline our strategies to feed to our shills and the tech press, they need to paint free software and its authors as true gems-- from a bygone era.

“Yes, that was really great. But now, it's time to look to the future.”

The future is (always) us, and the products we want people to use. There's no rise in quarterly revenue for tried-and-trusted, except when it's merely the glue holding our new products together.

Relevant quotes from the Halloween documents:

“OSS poses a direct, short-term revenue and platform threat to Microsoft -- particularly in server space. Additionally, the intrinsic parallelism and free idea exchange in OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer mindshare threat.”

“However, other OSS process weaknesses provide an avenue for Microsoft to garner advantage in key feature areas such as architectural improvements (e.g. storage+), integration (e.g. schemas), ease-of-use, and organizational support.”

“OSS process vitality is directly tied to the Internet”

“The OSS process is unique in its participants' motivations and the resources that can be brought to bare down on problems. OSS, therefore, has some interesting, non-replicable assets which should be thoroughly understood.”

“Open source software has roots in the hobbyist and the scientific community and was typified by ad hoc exchange of source code by developers/users.”

“Credit for the first instance of modern, organized OSS is generally given to Richard Stallman of MIT. In late 1983, Stallman created the Free Software Foundation (FSF) -- http://www.gnu.ai.mit.edu/fsf/fsf.html -- with the goal of creating a free version of the UNIX operating system.”

“Commercial software development processes are hallmarked by organization around economic goals. However, since money is often not the (primary) motivation behind Open Source Software, understanding the nature of the threat posed requires a deep understanding of the process and motivation of Open Source development teams.”

“In other words, to understand how to compete against OSS, we must target a process rather than a company.”

“These individuals are more like hobbyists spending their free time / energy on OSS project development while maintaining other full time jobs. This has begun to change somewhat as commercial versions of the Linux OS have appeared.”

“Coordination of an OSS team is extremely dependent on Internet-native forms of collaboration.”

“OSS projects the size of Linux and Apache are only viable if a large enough community of highly skilled developers can be amassed to attack a problem.”

“Common goals are the equivalent of vision statements which permeate the distributed decision making for the entire development team. A single, clear directive (e.g. "recreate UNIX") is far more efficiently communicated and acted upon by a group than multiple, intangible ones”

“Because the entire Linux community has years of shared experience dealing with many other forms of UNIX, they are easily able to discern -- in a non-confrontational manner -- what worked and what didn't.”

“Having historical, 20:20 hindsight provides a strong, implicit structure. In more forward looking organizations, this structure is provided by strong, visionary leadership.”

“Raymond posits that developers are more likely to reuse code in a rigorous open source process than in a more traditional development environment because they are always guaranteed access to the entire source all the time.”

“Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.”

“Release early. Release often. And listen to your customers.”

“Because the developers are typically hobbyists, the ability to `fund' multiple, competing efforts is not an issue and the OSS process benefits from the ability to pick the best potential implementation out of the many produced.”

“Universities are some of the original proponents of OSS as a teaching tool.”

“Linus Torvalds is a celebrity in the Linux world and his decisions are considered final. By contrast, a similar celebrity leader did NOT exist for the BSD-derived efforts.”

“What are the core strengths of OSS products that Microsoft needs to be concerned with?”

“The single biggest constraint faced by any OSS project is finding enough developers interested in contributing their time towards the project. As an enabler, the Internet was absolutely necessary to bring together enough people for an Operating System scale project.”

“Like commercial software, the most viable single OSS project in many categories will, in the long run, kill competitive OSS projects and `acquire' their IQ assets. For example, Linux is killing BSD Unix and has absorbed most of its core ideas (as well as ideas in the commercial UNIXes).”

“One of the most interesting implications of viable OSS ecosystems is long-term credibility.”

“Long term credibility exists if there is no way you can be driven out of business in the near term. This forces change in how competitors deal with you.”

“a product/process is long-term credible if FUD tactics can not be used to combat it.”

“OSS systems are considered credible because the source code is available from potentially millions of places and individuals.”

“The likelihood that Apache will cease to exist is orders of magnitudes lower than the likelihood that WordPerfect, for example, will disappear. The disappearance of Apache is not tied to the disappearance of binaries (which are affected by purchasing shifts, etc.) but rather to the disappearance of source code and the knowledge base.“

“Inversely stated, customers know that Apache will be around 5 years from now -- provided there exists some minimal sustained interested from its user/development community.”

“The GPL and its aversion to code forking reassures customers that they aren't riding an evolutionary `dead-end' by subscribing to a particular commercial version of Linux.”

“The "evolutionary dead-end" is the core of the software FUD argument.”

From https://antitrust.slated.org/halloween/halloween1.html

About the author:

Ted MacReilly is a technologist and tech writer concerned with modern trends in software design and development. He does not work for Microsoft, Apple, or Google, but would like them to continue offering proprietary software and cloudware, without getting too cozy with free software developers.