

Who Does That Server Really Serve?

GNU philosophy

This essay was originally published in the online edition of the *Boston Review*, on 8 March 2010, under the title “What Does That Server Really Serve?”

This document is part of GNU philosophy, the GNU Project’s exhaustive collection of articles and essays about free software and related matters.

Copyright (C) 2010 Richard Stallman

Verbatim copying and distribution of this entire document are permitted worldwide, without royalty, in any medium, provided this notice is preserved.

Who Does That Server Really Serve?

Background: How Proprietary Software Takes Away Your Freedom

Digital technology can give you freedom; it can also take your freedom away. The first threat to our control over our computing came from *proprietary software*: software that the users cannot control because the owner (a company such as Apple or Microsoft) controls it. The owner often takes advantage of this unjust power by inserting malicious features such as spyware, back doors, and Digital Restrictions Management (DRM) (referred to as “Digital Rights Management” in their propaganda).

Our solution to this problem is developing *free software* and rejecting proprietary software. Free software means that you, as a user, have four essential freedoms: (0) to run the program as you wish, (1) to study and change the source code so it does what you wish, (2) to redistribute exact copies, and (3) to redistribute copies of your modified versions. (See *The Free Software Definition*.)

With free software, we, the users, take back control of our computing. Proprietary software still exists, but we can exclude it from our lives and many of us have done so. However, we now face a new threat to our control over our computing: Software as a Service. For our freedom’s sake, we have to reject that too.

How Software as a Service Takes Away Your Freedom

Software as a Service (SaaS) means that someone sets up a network server that does certain computing tasks—running spreadsheets, word processing, translating text into another language, etc.—then invites users to do their computing on that server. Users send their data to the server, which does their computing on the data thus provided, then sends the results back or acts on them directly.

These servers wrest control from the users even more inexorably than proprietary software. With proprietary software, users typically get an executable file but not the source code. That makes it hard for programmers to study the code that is running, so it’s hard to determine what the program really does, and hard to change it.

With SaaS, the users do not have even the executable file: it is on the server, where the users can’t see or touch it. Thus it is impossible for them to ascertain what it really does, and impossible to change it.

Furthermore, SaaS automatically leads to harmful consequences equivalent to the malicious features of certain proprietary software. For instance, some proprietary programs are “spyware”: the program sends out data about users’ computing activities. Microsoft Windows sends information about users’ activities to Microsoft. Windows Media Player and RealPlayer report what each user watches or listens to.

Unlike proprietary software, SaaS does not require covert code to obtain the user’s data. Instead, users must send their data to the server in order to use it. This has the same effect as spyware: the server operator gets the data. He gets it with no special effort, by the nature of SaaS.

Some proprietary programs can mistreat users under remote command. For instance, Windows has a back door with which Microsoft can forcibly change any software on the

machine. The Amazon Kindle e-book reader (whose name suggests it's intended to burn people's books) has an Orwellian back door that Amazon used in 2009 to remotely delete Kindle copies of Orwell's books *1984* and *Animal Farm* which the users had purchased from Amazon.¹

SaaS inherently gives the server operator the power to change the software in use, or the users' data being operated on. Once again, no special code is needed to do this.

Thus, SaaS is equivalent to total spyware and a gaping wide back door, and gives the server operator unjust power over the user. We can't accept that.

Untangling the SaaS Issue from the Proprietary Software Issue

SaaS and proprietary software lead to similar harmful results, but the causal mechanisms are different. With proprietary software, the cause is that you have and use a copy which is difficult or illegal to change. With SaaS, the cause is that you use a copy you don't have.

These two issues are often confused, and not only by accident. Web developers use the vague term "web application" to lump the server software together with programs run on your machine in your browser. Some web pages install nontrivial or even large JavaScript programs temporarily into your browser without informing you. When these JavaScript programs are nonfree, they are as bad as any other nonfree software. Here, however, we are concerned with the problem of the server software itself.

Many free software supporters assume that the problem of SaaS will be solved by developing free software for servers. For the server operator's sake, the programs on the server had better be free; if they are proprietary, their owners have power over the server. That's unfair to the operator, and doesn't help you at all. But if the programs on the server are free, that doesn't protect you *as the server's user* from the effects of SaaS. They give freedom to the operator, but not to you.

Releasing the server software source code does benefit the community: suitably skilled users can set up similar servers, perhaps changing the software. But none of these servers would give you control over computing you do on it, unless it's *your* server. The rest would all be SaaS. SaaS always subjects you to the power of the server operator, and the only remedy is, *Don't use SaaS!* Don't use someone else's server to do your own computing on data provided by you.

Distinguishing SaaS from Other Network Services

Does condemning SaaS mean rejecting all network servers? Not at all. Most servers do not raise this issue, because the job you do with them isn't your own computing except in a trivial sense.

The original purpose of web servers wasn't to do computing for you, it was to publish information for you to access. Even today this is what most web sites do, and it doesn't pose the SaaS problem, because accessing someone's published information isn't a matter of doing your own computing. Neither is publishing your own materials via a blog site or a microblogging service such as Twitter or identi.ca. The same goes for communication not meant to be private, such as chat groups. Social networking can extend into SaaS; however,

¹ Brad Stone, "Amazon Erases Orwell Books from Kindle," *New York Times*, 17 July 2009, sec. B1, <http://nytimes.com/2009/07/18/technology/companies/18amazon.html>.

at root it is just a method of communication and publication, not SaaS. If you use the service for minor editing of what you're going to communicate, that is not a significant issue.

Services such as search engines collect data from around the web and let you examine it. Looking through their collection of data isn't your own computing in the usual sense—you didn't provide that collection—so using such a service to search the web is not SaaS. (However, using someone else's search engine to implement a search facility for your own site *is* SaaS.)

E-commerce is not SaaS, because the computing isn't solely yours; rather, it is done jointly for you and another party. So there's no particular reason why you alone should expect to control that computing. The real issue in e-commerce is whether you trust the other party with your money and personal information.

Using a joint project's servers isn't SaaS because the computing you do in this way isn't yours personally. For instance, if you edit pages on Wikipedia, you are not doing your own computing; rather, you are collaborating in Wikipedia's computing.

Wikipedia controls its own servers, but groups can face the problem of SaaS if they do their group activities on someone else's server. Fortunately, development hosting sites such as Savannah and SourceForge don't pose the SaaS problem, because what groups do there is mainly publication and public communication, rather than their own private computing.

Multiplayer games are a group activity carried out on someone else's server, which makes them SaaS. But where the data involved is just the state of play and the score, the worst wrong the operator might commit is favoritism. You might well ignore that risk, since it seems unlikely and very little is at stake. On the other hand, when the game becomes more than just a game, the issue changes.

Which online services are SaaS? Google Docs is a clear example. Its basic activity is editing, and Google encourages people to use it for their own editing; this is SaaS. It offers the added feature of collaborative editing, but adding participants doesn't alter the fact that editing on the server is SaaS. (In addition, Google Docs is unacceptable because it installs a large nonfree JavaScript program into the users' browsers.) If using a service for communication or collaboration requires doing substantial parts of your own computing with it too, that computing is SaaS even if the communication is not.

Some sites offer multiple services, and if one is not SaaS, another may be SaaS. For instance, the main service of Facebook is social networking, and that is not SaaS; however, it supports third-party applications, some of which may be SaaS. Flickr's main service is distributing photos, which is not SaaS, but it also has features for editing photos, which is SaaS.

Some sites whose main service is publication and communication extend it with "contact management": keeping track of people you have relationships with. Sending mail to those people for you is not SaaS, but keeping track of your dealings with them, if substantial, is SaaS.

If a service is not SaaS, that does not mean it is OK. There are other bad things a service can do. For instance, Facebook distributes video in Flash, which pressures users to run nonfree software, and it gives users a misleading impression of privacy. Those are important issues too, but this article's concern is the issue of SaaS.

The IT industry discourages users from considering these distinctions. That's what the buzzword "cloud computing" is for. This term is so nebulous that it could refer to almost any use of the Internet. It includes SaaS and it includes nearly everything else. The term only lends itself to uselessly broad statements.

The real meaning of "cloud computing" is to suggest a devil-may-care approach towards your computing. It says, "Don't ask questions, just trust every business without hesitation. Don't worry about who controls your computing or who holds your data. Don't check for a hook hidden inside our service before you swallow it." In other words, "Think like a sucker." I prefer to avoid the term.

Dealing with the SaaS Problem

Only a small fraction of all web sites do SaaS; most don't raise the issue. But what should we do about the ones that raise it?

For the simple case, where you are doing your own computing on data in your own hands, the solution is simple: use your own copy of a free software application. Do your text editing with your copy of a free text editor such as GNU Emacs or a free word processor. Do your photo editing with your copy of free software such as GIMP.

But what about collaborating with other individuals? It may be hard to do this at present without using a server. If you use one, don't trust a server run by a company. A mere contract as a customer is no protection unless you could detect a breach and could really sue, and the company probably writes its contracts to permit a broad range of abuses. Police can subpoena your data from the company with less basis than required to subpoena them from you, supposing the company doesn't volunteer them like the US phone companies that illegally wiretapped their customers for Bush. If you must use a server, use a server whose operators give you a basis for trust beyond a mere commercial relationship.

However, on a longer time scale, we can create alternatives to using servers. For instance, we can create a peer-to-peer program through which collaborators can share data encrypted. The free software community should develop distributed peer-to-peer replacements for important "web applications." It may be wise to release them under GNU Affero GPL, since they are likely candidates for being converted into server-based programs by someone else. The GNU Project is looking for volunteers to work on such replacements. We also invite other free software projects to consider this issue in their design.

In the meantime, if a company invites you to use its server to do your own computing tasks, don't yield; don't use SaaS. Don't buy or install "thin clients," which are simply computers so weak they make you do the real work on a server, unless you're going to use them with *your* server. Use a real computer and keep your data there. Do your work with your own copy of a free program, for your freedom's sake.