Question 1.a)

Do you consider that the release of source code of an implementation of the WSPP protocol specification would reveal technical details of protocol technology kept secret by Microsoft and embodied in the WSPP protocol specification?

Answer:

There is no doubt that releasing some source code of an implementation based on the WSPP protocol specification would necessarily reveal some internal information of the underlying technology of Microsoft protocols. This is however a very general statement and it must be better specified, and it will be in the next answers.

Question 1.b)

If your answer to question 1.a is affirmative: What kind of technical details about the working of:

i. Microsoft's protocol technology, or

ii. Microsoft's work group server operating system technology

could be revealed from the source code of such an implementation?

Answer

An implementation of the protocols published under the WSSP would implement most of the specifications. Samba would be published in source code form and thus would necessarily reveal much of the protocols involved. However, from reading the Samba source code, an engineer will not be able to reproduce the WSSP documents as such, because any protocol specification will leave many decisions about the particular implementation to the engineer. The Samba source code will only reveal the particular choice the Samba Team has taken, another engineer might come to a different solution from the same specifications.

There is no way to avoid a *limited* disclosure of internal information. It is impossible to assess now the magnitude of this phenomenon, because of too little information available. However, it is very unlikely that this information would be sufficient to allow a third party to build an alternative to Microsoft's operating system, as it would be limited to some very specific parts, almost irrelevant to the overall effort necessary to build a competing operating system solution and also because it would take way too much time to hit the market. The information published under the WSSP would only be useful to Samba Team, the only competitor with a suite of almost interoperable solutions available on the market.

Finally, it must be considered that there is a very different degree of revelation of the

“original” technology whether access is provided to the source code of the “original” implementation, to the licensed description of the protocols or to the source code of a different implementation written for another platform and based only on the licensed protocols description. Third hand informations are always poorer.

One example are algorithms. In our understanding of Microsoft's claims, some internal algorithms play a significant role in the overall design of their implementation. If these algorithms were to be disclosed, this could be done by allowing access to the original source code, something neither Microsoft **nor ourselves** desire, or by describing to a sufficient level what these algorithm do, what problem they address and what is the acceptable outcome of their use. In other words, sufficient information to implement in computer language **compatible** algorithms. Because the programmer can choose from a large variety of options, the odds are very scarce that two different programmers would use the same algorithm if presented with the same problem, and they would end up with two different solutions. It is possible that one is more efficient than the other, but nonetheless as long as they are compatible, this is an acceptable solution because this is exactly the normal competing practice in computer industry.

The question is if a third party could use the compatible algorithm to gather information as to the original and therefore ease the way to replicate the latter. The answer is "no". The compatible algorithm would give information on what kind of problems the same addresses, therefore it would give the kickstart to providing another **compatible** algorithm, maybe closer to the original one thanks to more research from other sources, but it would reveal nothing as to what the original algorithm was. Even this, however, would be nonsensical, because it would be more straightforward to use the compatible algorithm, which would be unrelated to Micorosft's original technology, and by no means would fall under copyright, secret or patent protection of the latter.

From other submissions, we have seen that Microsoft maintains that the correct practice of properly commenting the software could lead to reveal substantial information as to the accessed and secret information. We do not see the case. Commenting is a good practice in writing software, but the absence of comments, or a part thereof, does not prevent software from being compiled and work properly. In particular circumstances the individuals of Samba accessing the information could avoid comments, by issuing a “here I cannot say anything” statement that would address any concerns. Unlike what happens for proprietary software, there would be no problems in controlling what a single programmer does on a daily basis and what writes commenting his/her own code. Besides, the Samba Team does care of secrecy even without formal binding contracts with huge penalties, therefore there would be no higher danger in this respect than for other potential licensees, as this can be dealt with by contractual agreement.

Question 1.c)

If your answer to question 1.a is affirmative: Could, in principle, the information revealed by the source code of such an implementation, notwithstanding the necessary time and resources, also be gathered through lawful reverse engineering?

Answer

In theory this would be possible, at least for most of what is necessary, and it seems that this confirmed by Microsoft itself, who alleges that one way to easily achieve interoperability is reverse engineering. While this allegation is grossly wrong because the **inefficiency** of this process, which makes is practically impossible or useless competition-wise, it is true that by a full fledged lawful reverse engineering it would be possible to discover many – perhaps not all – details of the concerned technology. Samba is a reliable example of it. We do not doubt that, if given unlimited time and infinite highly skilled workforce, Samba would be now a lot more successful in providing a compatibility solution very close to Windows for non-Windows environments. The actual constraint is time and resources, because this process is highly inefficient, like trying to reconstruct a jar after having smashed it into pieces. Yet, while dedicating a lot of highly skilled work just to reconstruct the smashed jar, Samba has done a great job achieving an equivalent implementation of Windows NT4 (alas, six or seven years later).[1]

If Microsoft has engineered its implementation according to good practices, and not just to increase the entropy to a level that without the full information it is impossible to achieve full interoperability, largely the "missing pieces" could be achieved through a circular process of making certain assumptions and verifying them thoroughly enough by experiments. For instance, if the original software program expects that once the count reaches 31 the next number is 0, and the programmer of the alternative solution has understood that the count goes from 0 to 31 because a 2^5 value is exchanged, the latter can try to see whether the next value is "error" or is "0".[2]

1 Let us clarify better – and again – that what reduces the ability of Samba to efficiently compete with Microsoft Windows is not the lack of skilled developers or of management, as Samba 4.0 will stunningly reveal. What impairs the development of Samba is that it cannot make the right choices as to how work out the right solution, but it is forced by market constraints, and technological lock-ins, to follow the decisions of Microsoft. Microsoft has chosen and imposed the *de facto* standard, is in a position to change it at its choosing (and has done so many times, as it is going to do in the future), Samba must follow not because it is convinced that this is a better technology, but because it has no other options if it wants to maintain a compatible solution, because Microsoft has no desire – et pour cause! – to achieve any degree of interoperability. If Microsoft will be forced to disclose this kind of information, and would play fair ever after, we would see dramatic improvements on what the Samba Team can achieve, because it could dedicate resources to development, instead of trying to learn French in a French restaurant. Even Microsoft would then be allowed to derive from its competitor good suggestions as to how to improve its own protocols!

2 May we make an example related to "byte range locking". Jeremy Allison and Andrew Tridgell worked out the correct external behavior for byte range locking in SMB/CIFS clients by writing a test program called 'locktest'. They implemented this in Samba, and it passed any tests. Then while tweaking the test program Tridgell discovered a strange anomaly - Windows was behaving erratically for a particular lock, which became famous in the Samba team as the 'zero/zero' lock. It was determined that this erratic (and incorrect) behavior reveals a whole lot of information about how Windows handles byte range locks. It showed very clearly the specific tree structure that Windows uses to represent these locks internally. Samba uses quite a different (and much simpler!) structure to represent these locks. The two algorithms are compatible for all locks except the zero/zero lock, and Samba Team is almost certain that its algorithm is the best suited for that case (as the Windows algorithm is not deterministic from the point of view of the API user).
By seeing the algorithm chosen by Samba, a third person would simply not be in a position to achieve the same level of awareness than that obtained doing network analysis.
A second example is the algorithm for checking whether a user is allowed access to a file, the so called access_check() algorithm. Tridgell worked out the fine details of how access_check() must behave. Then, while testing this findigs in against Win2003, he discovered the latter had a bug in its access_check() code, which incidentally results in a security hole. That bug also happens to reveal something about how the Microsoft algorithm for access_check() works. Obviously the Samba Team

We concede, however, that there is a number of actions that Microsoft *could* implement to make it a lot harder, or even impossible, to continue doing network analysis as a method to achieve interoperability, such as preventing network sniffing with methods that cannot be turned off by the user. Yet this decision would be abusive itself, because it would be aimed at preventing interoperability and would have no security reasons whatsoever, therefore this possibility is not relevant.

A higher level of information can be gathered by reverse engineering, i.e., decompiling (something that Samba Team *does not* do), the resulting implementation even in the event that it is released as object code only. Possibly this *could* be even easier than performing the same reverse engineering on the original Microsoft product, because of the better design of it, or because of the platform for which it has been compiled, or for the more advanced compiler and libraries used, or again because the single object code file is more specifically designed to accomplish a limited task therefore it results being smaller and simpler.

Question 1.d

If your answer to question 1.a is affirmative: Could the information revealed be used for other purposes than developing work group server operating systems? If yes, how likely is it that such information would be used for other purposes?

Answer

Again, it is impossible to asses how much the revealed information could be important to allow third parties to acquire some information to be used elsewhere. Nothing contradicts the assumptions that some of the technologies used to build a network operating system – we mean, the one resulting from proper implementation of WSPP – could be used for other purposes (actually, this is one of the goals of Free Software: that of allowing information to be spread beyond the limits of a single field of application, because it is this the way technology often progresses). The exercise of extending this notion to Microsoft's operating system is however highly speculative. This surely would not be the case for all examples we can think of, and we maintain that the burden of proof that this is likely to happen in specific and yet undisclosed cases lies on Microsoft, at least a burden of allegation exists.

For instance, if Microsoft has found a particular unpatented process to discover what server in a mixed environment is the best suited to be first accessed for downloading a printer driver to be installed on a client, there is no indication that this particular process could be used for, say, locating a computer to download music in a peer-to-peer environment over the Internet. However:

---

did not change the (working!) algorithm it had implemented beforehand to match the security hole in Win2003.
The Samba Team tends to aim for very simple, robust algorithms. It seems from our discoveries that Microsoft has different priorities (not surprising, as their OS is quite different from Linux/Unix) and as a result it develops quite different solutions to the same problems. We are rather confident by this and many other examples that the solutions found by Samba and those of Windows are rarely (if ever?) the same.

1. this process would not be immediately useful; chances are very high that it should be deeply modified and adapted to the different working environment, so that it would receive substantial innovation. This is something Microsoft claims it has done with LDAP, and to some extent it has done, but LDAP was already meant to provide directory services to public and private networks. The current example could be very specific to finding drivers: the more specific it is, the more unlikely it could be turned into something else;

2. this process could be not innovative at all, and/or be outperformed by something already existing;

3. There could be many alternatives more tailored to the purpose;

4. If the above would not apply because Microsoft had implemented an indeed new secret, very efficient and potentially all-purposes protocol, a special agreement could be found to avoid the implementation of the same process by the WSPP licensee, who would undertake to find a compatible process using other algorithms and solutions, with the help of the Trustee to ensure that not too much sensitive information are passed in the process;

The very vague example given above shows how many assumptions must be made to find what Microsoft alleges. Any theory requesting so many assumptions is in principle false according to well established criteria of logic. If some very limited cases were to spring up, they could be dealt with on a per case basis.

Moreover, as stated elsewhere, because what would be used by the Samba Team is strictly speaking the interoperability information, even without the need for special arrangements as envisaged by point 4 above Samba would most likely require different algorithms and implementations. Therefore we see little chances that the actual implementation of Microsoft would be anyway revealed and made usable for other purposes without further research and without the access to the original WSPP information. What would be gathered by accessing the source code of the implementation would be a very small portion of what would be necessary to use the information for non-interoperability purposes.

Question 2.a and 2.b)

According to the information available to the Commission Services, some software developers distribute proprietary software and software subject to an open source licence under separate licences, but layered together to form a single hybrid product “hybrid distribution”).

a) Do you use such “hybrid distribution”?

b) If your answer to question 2.a is affirmative: what are the reasons for using this distribution model and for which products do you use it?

Answer

We FSFE and Samba Team **do not, nor intend ever** distribute or promote or anyway endorse the use and distribution of proprietary software.

The GNU GPL, under which Samba is distributed, while allows the four freedoms (to use, study, modify and distribute the software, and any combination of the above), is a copyright license, and does not allow to Free Software to become proprietary. GPLed software must remain free – as in freedom –, so that the GNU GPL is a copyright license that ensure the freedoms, and that the freedoms are kept whenever the copyrighted work is reused, thus the definition of it as "copyleft". A hybrid product – that is to say a single program consisting in a Free Software part and an integrated proprietary part – would violate the copyright of the original authors and assignee of it, and this copyright has the same level of protection of Microsoft's. And Samba is not a "dominant" company: it would be odd, to say the least, if to put an end to a market abuse the only remaining competition of it would be forced to change the very license that has made its strength.

The examples provided **are not** hybrid products: they are self-standing software applications that make use of an underlying Free Software operating system. Real Player, Adobe Acrobat Reader, Novell Zen Works and NetMail are applications which exist also for other operating systems, and some are only **ported** to a GNU/Linux environment. This does not contradict the GNU GPL, because Freedom number 0 allows the use of the software for whatever purpose the user deems fit. An operating system is the system software responsible for the direct control and management of hardware and basic system operations. Its natural use is, inter alia, running applications and controlling the executions and interaction with them, via standard APIs. Contrary to what Microsoft's maintains against any evidence and common sense, a web browser which runs on the top of an operating system does not form a single piece with the underlying operating system more than a human being using a stair does form an hybrid entity with the stair.

Moreover, the GNU GPL does not prohibit to distribute proprietary programs together with the GPLed ones, as long as they are a distribution (a collection) of independent products. Conversely, the GNU GPL forbids to license derivative works under different conditions that the GNU GPL itself. The shipment of two different programs under different conditions is therefore **irrelevant** to our case, and we do not see how consequences can be drawn as to the viability of a "blackbox" solution.

Very different is however the example of the kernel drivers, and the purported example needs a little more explanation of how and for what specific and non replicable reasons, the usage of object-code only drivers has been possible, the so called "Linux exception".

Linux is the particular kernel that forms the GNU/Linux system. GNU/Linux is the most widespread version of GNU systems, and the most widespread version of a UNIX-like operating system. The other largest distributed UNIX-like operating system is BSD, and among BSD the most known is MacOSX, which is based on the Free Software project Darwin. One difference between GNU/Linux and BSD is the license: the BSD license is very basic and does not have "copyleft" or persistent effects, which allows for the use of BSD software to make proprietary software, including the use of proprietary modules to work with the BSD kernel. The same cannot be done with Linux, because Linux is under the GNU GPL. However, Linux is copyright of Mr. Linus Torvalds and

many other developers.

Since version 2.0 Linux changed from a monolithic kernel (a single self-contained file) to a modular kernel, which dynamically loads external modules on demand, thus reducing the size of the kernel itself and the memory used by it, with various advantages. The most likely piece of software to be compiled as modules instead as a part of the kernel are device drivers. Some hardware manufacturers found it advisable to start supporting GNU/Linux for their products, but did not want to distribute their device drivers in source code form, instead they preferred to ship only the object code. The GNU GPL, however, considers the use of external modules to be loaded by the main program as a part of a single program, notwithstanding if this is made by statically as opposed as dynamically linked software, and even plug-ins fall into this category. Between the options to keep the strict meaning of the GNU GPL, thus avoiding the use of proprietary modules, or to loosen it to enlarge the supported hardware list, Linux Kernel Developers chose to 'tolerate' proprietary kernel modules but allow symbols to be marked as only available to GPL modules.

It is evident from the discussion above that:

1. this is a special exception made by the copyright holders of a particular software program;

2. the exception is valid insofar the standard symbols are used, i.e., no special – even GPLed – modification of the kernel needs to be made to specifically load a particular proprietary module[3]; and

3. for the reason above the exception applies in a very limited field, and anyway at another level from the example at hands.

We do not enter into the discussion if this choice was correct or useful – although we have high reservations on it – but this decision of Linux Kernel Developers shows itself that, again, there is no possibility to draw consequences from this example.

Question 2.c)

Is it technically feasible to use hybrid distribution for proprietary implementation of the WSPP protocol specification?

Answer

No, It is possible, but unfeasible.

The purported examples of hybrid software are programs that run on the top of something conceived to run programs, or to load external modules, therefore naturally and logically separated from the underlying software or by the main part of the kernel to perform specific tasks.

3 See http://www.atnf.csiro.au/people/rgooch/linux/docs/licensing.txt for more details from Mr. Torvalds' voice and also http://www.tux.org/lkml/#s1-18 and http://www.tux.org/lkml/#s1-19

With network protocols, it is different. Again, in theory, nothing would prevent the use of a particular software solution to perform one network task and another part to perform a different one, for instance, one requesting the address and public certificate of the server acting as authentication server, and another issuing the request for authentication to it via the public certificate of the so chosen authentication server. If these process would have been separated and clearcut defined, nothing would prevent also the client parts to be exchanging these services via a standard API (for instance, a Kerberos and a LDAP clients). Actually Samba can use OpenLDAP as directory services and the MIT Kerberos services in a GNU/Linux environment. These two components could be one proprietary and one Free Software. Similarly they could be replaced by similar software applications, as this is natural for the UNIX architecture: small programs that perform limited tasks very efficiently and rely on others for different tasks.

But we are not speaking of neatly designed UNIX network services communicating via well-defined and public interfaces, but an “interconnected web of services” commingled together and conceived to work as a monolithic, one would say clumsy, mixture of protocols with little separation and little if any tolerance as to the order of requests. This is not a choice of Samba, rather the opposite, and is the only way the protocols can be used to obtain valid answers and interoperate with Windows. According to Samba experience this is dissimilar from any competing solution. And it is allowed only because Windows controls (or assumes to control) both sides of the network (by this meaning either the server and the client, and two or more servers for server-to-server communication). In other words, Windows assumes that no interoperability is present or will be allowed! The allegation that the Windows network operating system is “tightly coupled” is, in this, not far from the truth, if we mean “it has poor interfaces, designed to be the least interoperable with different solutions”.

The classic concrete example here is password changes. They can happen over at least LDAP, Kerberos, CIFS and DCE/RPC. Windows clients expect that once a password is changed over one protocol **all** other components of an Active Directory implementation know about the new password without any delay, so all protocols need to access the very same database.

The above means that also the Samba client and server must be designed to work the protocols out as a single protocol. The WSPP therefore should be implemented with a clear and all-encompassing view. If we considered the option of building a proprietary part of Samba, this could not, by any means, be done by Samba (see answer to Question 4). This means that, in case the WSPP were not to allow Free Software distribution of the implementation, we would have only two possible scenarios:

1. A “hybrid” solution, one part made of an extension of the present Samba solution; and the other distributed with proprietary license, made by external developers;

2. An all-proprietary solution, made from scratch, something that the MCPP show unfeasible.

We are going to explain under answer to Question 4 why also solution number 1 is totally unacceptable, and it would be even worse than the present situation.

Question 3

According to the information available to the Commission, Services, various technical solutions could be used to implement hybrid distribution models with a Linux-based operating system. These are:

i. binary-only loadable Linux Kernel modules (i.e., software which on request adds functionality to the kernel but is only distributed in binary form);

ii. binary-only software loaded and executed in the "user space" of the kernel;

iii. binary only software which is "dynamically linked" to the kernel;

iv. the use of an intermediate "shim" that acts as an interface between the kernel and a user space program that is distributed in binary form.

a) Please specify whether any of these four solutions could be used by you to develop and distribute a proprietary implementation of the WSPP protocol specifications together with a Linux based operating system?

b) Could you please outline in detail the technical advantages and drawbacks that each of the above-mentioned solutions would entail.

Answer

The reference to the Kernel is something we would not consider at all: Samba is by no means a kernel module and will never be. So only alternative ii is relevant at all, but not with Samba.

Solution i. and iii. are substantially the same. Solution i., ii and iii. would imply to enter into the scenario number 2) above. They would indeed require to build an alternative solution that would **entirely** replace Samba, trashing almost 15 years of development and restarting everything from scratch. The point is not whether there can be an interoperability solution to run on the top of GNU/Linux, but an interoperability solution that would work and be commercially and technically available in the short term (as one said, on the long term we would be all dead anyway), i.e., in the two or three year timescale. This would be plainly impossible for any of the potential players. Microsoft itself has developed the current technology for more than ten years, and struggled for many years before reaching a decent implementation, being in the position to pull handles on all sides of the problem.

Any investment on proprietary solutions must rely on a long-term repayment with very little margins (a positive effect of competition) and with the uncertainty of a possible termination of the WSPP licenses, subject to possible reversal of the Decision in court or to reevaluation of the Commission at any stage. This only to achieve a level similar to the present Samba. And most of the expert programmers able to bring this project to a working solution are either with Microsoft or with Samba. Building an entire GNU/Linux solution (but the problem extends also to Darwin/MacOSX, which uses Samba as well,

and to other minor players), would take at least two or three years operating on a quite large project. Of course neither the Samba Team nor the FSFE are by any means interested in this. "*Good night and good luck*" to all wishing to follow this path. The MCPP in the USA has shown how little interesting this is: nothing relevant has been produced in years.

The solution under iv. is difficult to understand. The only way a "shim" could be conceived is to relay some services through a proprietary interface towards Windows, and through internal interfaces towards Samba, which would perform the rest of the tasks. If the "shimmed" program would have to be a proprietary one, there would be no need for a shim, as it could use directly the "WSPP" interfaces under current license terms, and Samba (as well as the entire Free Software) would have no role at all even here. The shim itself would include, not exclude, the problems envisaged by Microsoft, therefore it could not be Free Software, according to Microsoft's reservations and objections. Actually, Samba is such a "shim" to any application program that requests network services (printing, retrieving and storing files, etc...), and these application programs do not have to perform any network activity, therefore there is no way Microsoft could be legitimately concerned by the fact they are Free Software or proprietary applications, as they have no network interfaces at all.

See answer to question 4 for further discussion.

Question 4

Please specify whether it would be possible for Samba to develop implementations of the WSPP protocol specifications under a subcontract with vendors who themselves would distribute such implementations with a Linux kernel using hybrid distribution techniques. Please specify whether the development of such implementations would be subject to the (GNU) GPL. Please specify whether Samba would consider developing such implementations under a license different from the (GNU) GPL

Answer

The kernel has no role whatsoever with Samba. Samba is compiled for a variety of operating systems, GNU/Linux being just one of them. We do not see why the fact of Samba, or any alternative solution, being distributed along with any sort of kernel would have a role in the present discussion. As said before, **the point is not Linux**, or GNU/Linux, **but Samba**. And for FSFE, GNU/Linux with a proprietary Samba-alike solution is almost as uninteresting as a Windows native solution. For Samba, a GNU/Linux solution of that kind is even meaningless. Finally, the point is not distribution, but freedom.

The only thing we know for sure is that if Samba would have to rely on a particular piece of kernel it would lose much of its portability to other platforms. If this piece of software was proprietary, all the program would depend stricto sensu on it, and therefore the choice to use Samba with other platforms would depend on external providers. Samba must therefore remain a self-contained suite of applications otherwise its market value will significantly be lowered.

We have explained in previous submission to the Commission why using kernel-space applications would be very detrimental also from a security and stability perspective not only of Samba, but of the overall operating system as well. Microsoft has changed its design of Windows exactly to bring the network part of the operating system outside the kernel space, Samba has never struggled the same way because it had better design, it would be plainly stupid to go in the opposite direction.

This reason excludes solutions i. and iii. above from the realm of possibility.

The suggestion made here seems to conduct to an hybrid solution in the user space (one part of Samba, one part proprietary). As said before, not the software science, but the way Microsoft has conceived its network operating systems, makes impossible to write separate parts working together via APIs. Only this kind of separate parts would have a chance to be developed independently, because each one would take care of certain sharply defined specifications, and them alone. Being this impossible by design, due to the interlocked concept of the WSPP protocols, only a very integrated[4] solution is feasible. But this integrated solution would be **subject to the GNU GPL**, because it would be a very single product, unless precautions are taken.

The concept of separation must be introduced. It is indeed possible to split a single program into different pieces, by separating them and having them to communicate through interfaces. The point is that there are natural and unnatural interfaces, which translates into “efficient and inefficient ones”. The communication between Samba and the proprietary part should be performed largely by unnatural interfaces. **Interfaces** designed by **legal** rather than technical considerations, thus **inefficient** and **unnecessary**, therefore raising the **complexity** of an already very complex project **by one order of magnitude**. This itself is unacceptable.

A very challenging task would be that of defining these interfaces if the Samba Team is excluded from the WSPP information. This would be the case because the access to such information would prevent them to work on Free Software, for the simple reason of avoiding to be charged of illicitly using the information. A Free Software developer can start developing proprietary solutions, as long as the copyright held by third parties is preserved, a proprietary software developer cannot work on Free Software without the permit of its former or present employers in the same field or without the risk of being accused of stealing or revealing secret information. This asymmetry exists also in the present case. Once proprietary developer, forever proprietary developer.

The result for the Samba Team would be not better than the present situation, it would be worse and worse. This time, instead to be faced with “natural” network interfaces, it would have to struggle with “internal” interfaces, blindly relying, for them, on what would be proposed by the “proprietary team”.

Not that the “proprietary” team would be in a better position. Let us suppose that Microsoft would change its protocols: the team would have to change the proprietary interfaces without counting on a larger community, and possibly also the “Samba” ones without the cooperation of the Samba people, and the Samba people would have only to wait and start doing their part when the proprietary team would communicate what their decision is on the particular subject. Even worse, our experience from the existing proprietary modules of the kernel says that their overall pace of development is way slower

4 Integrated between the black-box and Samba

than the Free Software ones.

But the most unacceptable outcome is that a substantial part of what is presently Samba would be a **proprietary** program, and the rest would **depend** on it flatly: this would be the dream of Microsoft, and a big achievement from its point of view, it would not even resemble a measure to restore competition. Instead to face the only real competition, i.e., Free Software, Microsoft could "concentrate" on a particular proprietary software manufacturer, which would be the weakest link of the chain. And even the most recent history of this case shows that Microsoft has enough arguments and incentives to "solve privately" the issues with its competitors that are strong enough to be a threat but not out of reach of what money can buy or of what other threats can achieve otherwise. The history proves that in many fields, the one at hands included, the only vaccination to preserve live competition is Free Software: once excluded this, the major problems are solved.

Wouldn't this be funny? The solution to restore competition, in Microsoft view, would be to cripple its competitors by *pro*-competitive measures even worse than by the free development of anticompetitive market actions!

Samba **would and could never accept**, nor pursue, such solution. Samba **has no interest** in "subcontracting" or being subcontracted of anything, because anyone is already invited to contribute to the Samba project. Samba **has no intention to change the license of Samba**, i.e., **Samba is and will remain GNU GPL licensed software**. The only different license acceptable, but for limited purposes and in exceptional circumstances, would be a compatible license, such as the GNU LGPL, but this would be only for newly developed software, and this would not address the "Free Software issue", i.e., the need to publicly release the source code.

Again, we would regard it as a very peculiar solution to solve the situation created by an abusive conduct: that of imposing a change in the business model of the competitor in the weakest position instead to press on the abusive one. It would be tantamount to request bankers to leave their vaults open and to sack their security personnel in order to avoid the risk of armed robberies.