

Teaching Wine The Samba Thing

Visions For Wine/Samba Integration

Kai Blin

**Wine Developer
Samba Team member**

Wine is a compatibility layer for running Windows programs on Linux/Unix without requiring a Windows license. Samba, as a file and print server for SMB/CIFS clients, provides network-aware features for Linux/Unix and Windows interoperability already.

The goal of the talk is to provide up-to-date information about the status of Wine's Samba integration and to show other developers who might want to hook into Samba with some ideas how to do this.

1 Introduction

Open Source development is a place in software development where everybody can work on his own ideas, and finally do things the right way. The NIH syndrome, "Not Invented here", is common in a lot of projects. So why bother with integration when one can write one's own version just as well? Of course, the obvious answers are that code reuse will save development time, bugs will only have to be fixed once, and the tried and true paradigm of rather doing one thing well than being mediocre to bad in a lot of things. This paper looks into ways to use Samba know-how for Wine.

1.1 Wine

Wine is a translation layer (a program loader) capable of running Windows applications on Linux and other POSIX compatible operating systems. Windows programs running in Wine act as native programs would, running without the performance or memory usage penalties of an emulator, with a similar look and feel to other applications on your desktop.

1.2 Samba

Samba is an Open Source/Free Software suite that provides seamless file and print services to SMB/CIFS clients. Samba is freely available, unlike other SMB/CIFS implementations, and allows for interoperability between Linux/Unix servers and Windows-based clients.

Samba is usable in a variety of roles in a CIFS-based network. It can be used as PDC (Primary Domain Controller), as a member file server and as a client computer.

1.3 Licensing

Both Samba and Wine are Open Source projects, using licenses from the Free Software Foundation. However, the current Samba development happens on code licensed under the GNU General Public License (GPL) version 3. Wine is licensed under the GNU Lesser (or Library) General Public License (LGPL) version 2.1. Those licenses are incompatible when trying to import code from Samba into Wine. While the version issue is easy to avoid (Wine's license says "version 2.1 or higher"), importing GPL code into LGPL code just is a no go. The licenses are designed that the other way around is easy, but GPL to LGPL is not possible.

2 Motivation

There are multiple reasons motivating the work on Wine/Samba integration. One is the potential of not having to re-implement functionality Samba already has in Wine. This is summed up in the following section. Another reason is that there are some functionalities that require a process listening on specific privileged ports. Wine should not be run as root for security reasons. Also, if Samba is already bound to those ports, Wine cannot bind there. Requiring that people either run Wine or Samba, but not both, if they want their Linux box to participate in a Windows centric environment is not a good solution either. Last but not least, once a defined interface for interacting with Samba is established, other projects will be able to take advantage from this, too.

2.1 Code Sharing

The code Samba can provide for Wine is mostly in the area of user management, which potentially is remote, as well as other code involving CIFS network interaction.

2.1.1 Local Security Authority

The Local Security Authority, or LSA in short, is the local checkpoint to authenticate and authorize users logging on to the system. In a domain or active directory environment, the LSA will have to query the log-on server using the SAMR pipe, this would ideally be handed over to Samba, which already does this for it's own work.

Wine currently does not handle user authentication at all. Apart from some of the test cases, there is no pressing need to get this working yet.

2.1.2 Netapi32

User and group management functions are exposed by the netapi32 interface. Handling both local and remote users and groups, this interface is the core of Windows user management. Samba supports this, of course. Wine so far has a very crude implementation of this that is only designed to make writing some test cases easier.

2.1.3 GENSEC / SSPI

GENSEC is an encryption layer that abstracts over the actual encryption method used on the connection, providing a unified interface for all supported encryption algorithms. Basically, this is similar to the GSSAPI standard proposed by the IETF in RFC 2744. However, it is designed to be compatible with a piece of software from Microsoft, called the Secure Service Provider Interface, SSPI in short, which is almost, but not completely wire-compatible to GSSAPI. Wine currently implements the NTLM provider of the SSPI wrapping calls to a Samba tool called `ntlm_auth`. This is not really extensible for other providers, so a different solution would be good.

2.1.4 libsmbclient

libsmbclient handles the client side of SMB/CIFS networking, so browsing for shares and printers, accessing remote shares using smb:// style URIs, and NETBIOS name resolution. Wine's winefile.exe, a file explorer for use in Wine, would benefit from being able to access network shares, too. A couple of GPL licensed file managers already use this, Wine is stopped by licensing issues.

2.2 RPC over SMB

RPC over SMB is the thing that makes Distributed COM (DCOM) distributed.

2.2.1 Endpoint Mapper

COM objects register their interfaces to the Service Control Manager (SCM). Other COM objects query the SCM to find interfaces they need to perform their job. In order to make this process transparent over the network, SCMs talk to remote SCMs to figure out if an interface is available on a remote host. The endpoint mapper stores where those interfaces can be accessed. Samba 4 already handles this, but does not provide a public interface for it. Wine also has an endpoint mapper, but that only takes local COM objects into account.

2.2.2 RPC Over SMB Named Pipes

There are multiple methods of using DCE/RPC, Wine only supports DCE/RPC over TCP/IP so far. Some operations are only possible via RPC over SMB using SMB named pipes. Examples include accessing a remote registry or remotely managing system services.

2.3 Benefits To Other Projects

The obvious benefits are that there will be a defined and used interface that other projects with similar needs (or a subset thereof) can use. This interface will be tested with regression tests both from the Wine and from the Samba side, so bit-rotting is not a problem either.

3 Implementation Ideas

Over the course of years, multiple suggestions have been brought up on how the interaction can be done. The following sections discuss the more popular and/or practical approaches.

3.1 License Changes

Both Wine and Samba are Open Source software, so sharing source code should be easy, right? Not quite. Wine is licensed under the GNU Lesser General Public License (LGPL), Samba is licensed under the GNU General Public License (GPL). While the LGPL is designed to be

upgradable to an equivalent-versioned GPL license, making inclusion into GPL code easier, this is not true for the opposite direction. GPL code cannot be included into LGPL code.

Now, one suggestion that comes up on mailing lists once in a while is that this can be easily solved by converting Wine to the GPL. Obviously, that is not an option. In order to allow porting of Win32 applications to Linux using winelib, it is desirable to not force people to convert their Win32 application to the GPL. This would be necessary if winelib was GPL-licensed.

3.2 Stdio Over Pipes

This is a solution that is currently in use to provide the low level bit pushing for the NTLM Secure Service Provider (SSP) via `ntlm_auth`. However, adding functionality in such an interface is hard. Wine handles encryption and cryptographic signatures on NTLM itself because there is no good way to hand this off to the `ntlm_auth` tool. This results in code duplication, as Samba of course also has code to handle signing and sealing via NTLM.

Adding further security providers as Kerberos or SPNEGO will not be possible in any reasonable effort either. So while this is a technology that works now, it is not a viable way to continue on.

3.3 DBUS

DBUS is a message channel developed by *freedesktop.org* in order to allow desktop programs to communicate with each other, as well as providing a system channel that provides information about state changes (e.g. "network connectivity lost"). However, apart from the system channel, communication is only possible between processes of one user, and as Samba usually runs as root, this is not a likely scenario. This is due to the missing security concept on DBUS.

3.4 DCE/RPC

The best option currently available seems to be local DCE/RPC connections, e.g. over a Unix named pipe. As both Samba and Wine have a code generator that understands the Interface Definition Languages (IDL), creating a protocol for this should be straightforward. Some interfaces will have to be developed to provide as-of-yet private Samba functionality via the RPC interface, as well as calling those interfaces from the Wine side. DCE/RPC supports authentication, so it does not suffer from the main problem DBUS posed for Samba/Wine interaction.

Conclusions

Even in the Open Source world, cooperation can be hard. However, there are methods to resolve these problems in a way that all participating projects can work with. Cooperation between Samba and Wine will make Linux computers more useable in a Windows-centric environment and is certainly an important step towards making Linux more widespread as a desktop operating system.