Implementing Windows Software Licensing Support in WINE for Open-Source Operating Systems

Ethan Graber (Honors Capstone)

ENGINEERING HONORS PROGRAM
UNIVERSITY OF MICHIGAN

Introduction: Windows Syscalls and the Wine Project.

When an application is running on a computer, it can request services (e.g. reading a file, opening a network connection) from the underlying operating system using a *system call* (or "syscall").

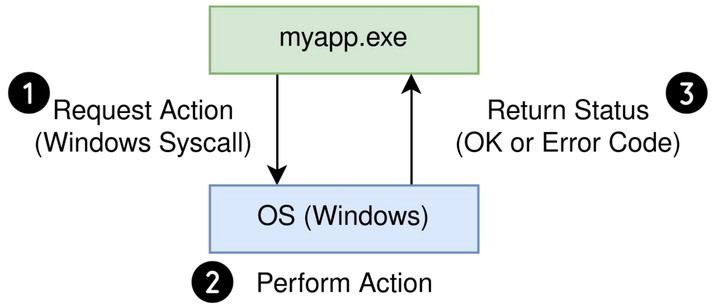


Figure 1: A Basic Overview of a Windows Syscall

Applications made for Windows cannot be run directly on other operating systems, primarily because of the differences in available APIs/syscalls ("Application Programming Interfaces", i.e. commands the system knows how to understand) across operating systems.

WINE is a popular piece of open-source software which aims to bridge the compatibility gap between operating systems by *translating* Windows Syscalls into calls to WINE function calls.

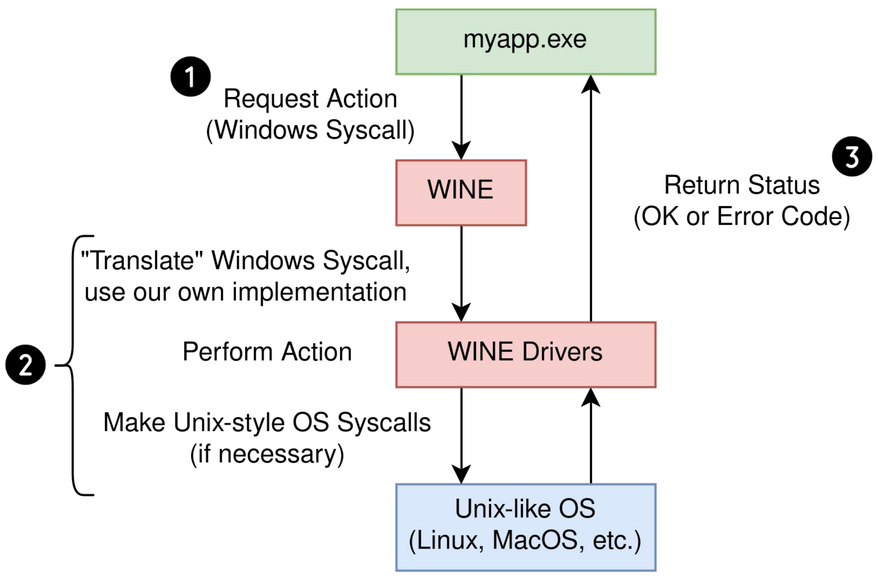


Figure 2: Overview of the same Windows Syscall made through WINE.

Adapted from:
wiki.winehq.org/Wine_Developer's_Guide/Architecture_Overview

Problem: Wine doesn't Implement *all* Windows APIs.

The Wine developers implement APIs in Wine as users or developers discover the APIs are needed. Applications that require a missing API will crash.

For example, the Microsoft Office 365 suite of products (Word, Excel, Powerpoint…) do not run properly through Wine, because Wine does not fully support the APIs that check whether the user owns a license to use the product.

This project attempts to add code to the Wine Project to implement the Software Licensing functionality found in Windows, and thus enable support for licensed applications like Microsoft Office 365.

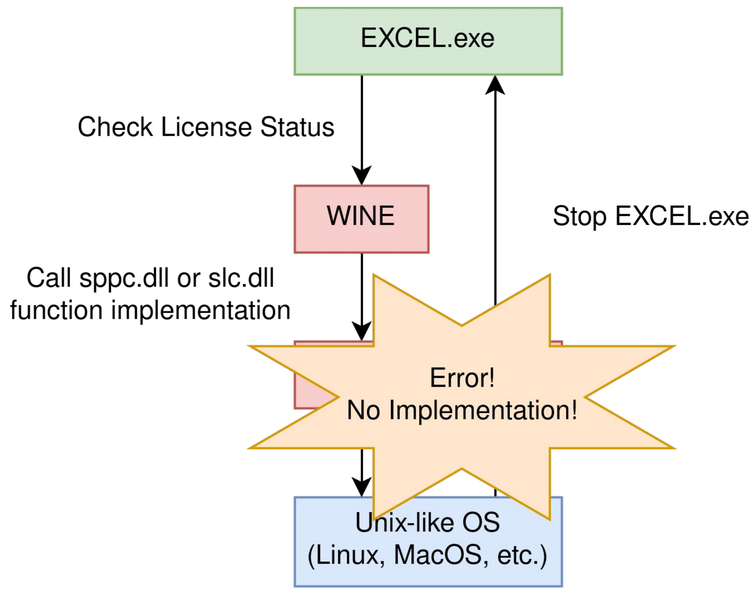


Figure 3: Excel fails an API call to check if it's licensed to run

Challenges: Clean-Room Reverse Engineering and a crippling lack of Documentation.

The Wine Project follows the *clean-room* design method for implementing any code that mimics functionality provided by Windows. In practice, this means that Wine developers **cannot** reference any Windows code (directly or decompiled) when writing Wine code. This would be considered a copyright violation.

Coupled with the lack of existing documentation about how the Windows Software Licensing APIs are meant to behave, this makes implementing bug-free functionality extremely difficult. Functionality details need to be guessed, and an incorrect or incompatible guess leads to a crash.

Solution: Iteratively implement missing functionality.

Whenever a program running through WINE encounters an unimplemented API, Wine will terminate the program immediately. This makes it impossible to determine the full set of APIs an application will call until the previous crashing call is fixed or marked as "unnecessary" (Many functions are redundant for the purposes of compatibility).

Wine API implementations don't need to (and should not) match the exact *implementation* of the Windows API, so long as the *functionality* remains the same. I was able to drastically simplify the license storage process by leveraging Wine's implementation of the Windows registry as a simple and secure key-value store. This is almost certainly not how Windows handles storing licenses, but it allows Wine to store and retrieve licenses exactly as the application would expect.

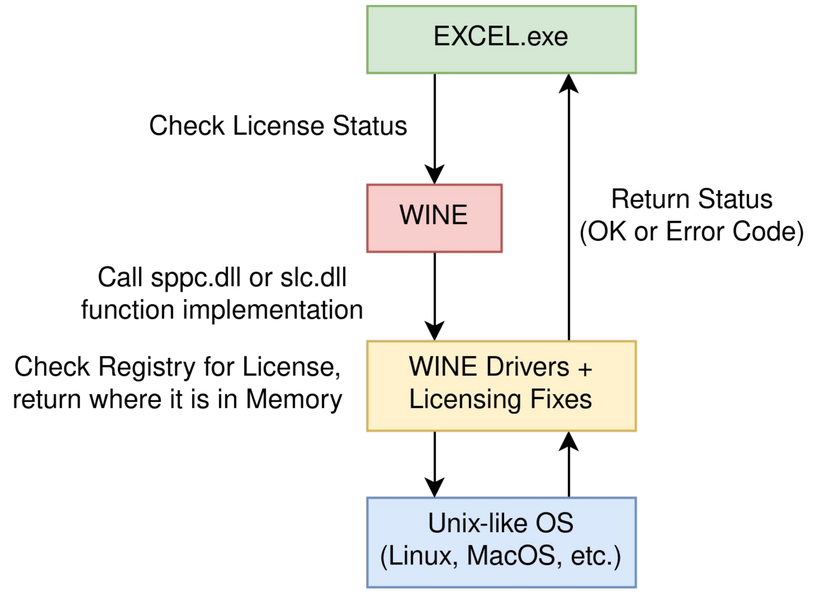


Figure 4: Excel is able to retrieve the license for later verification

Future Extensions / Next Steps

In the time it took me to make the modifications, there have been about ~1,300 other changes to the Wine source code. I will merge those changes with my changes and verify that everything continues to work as expected.

Finally, my changes will be submitted for approval by the Wine project maintainers. If approved, they will become included in the next official release of Wine.