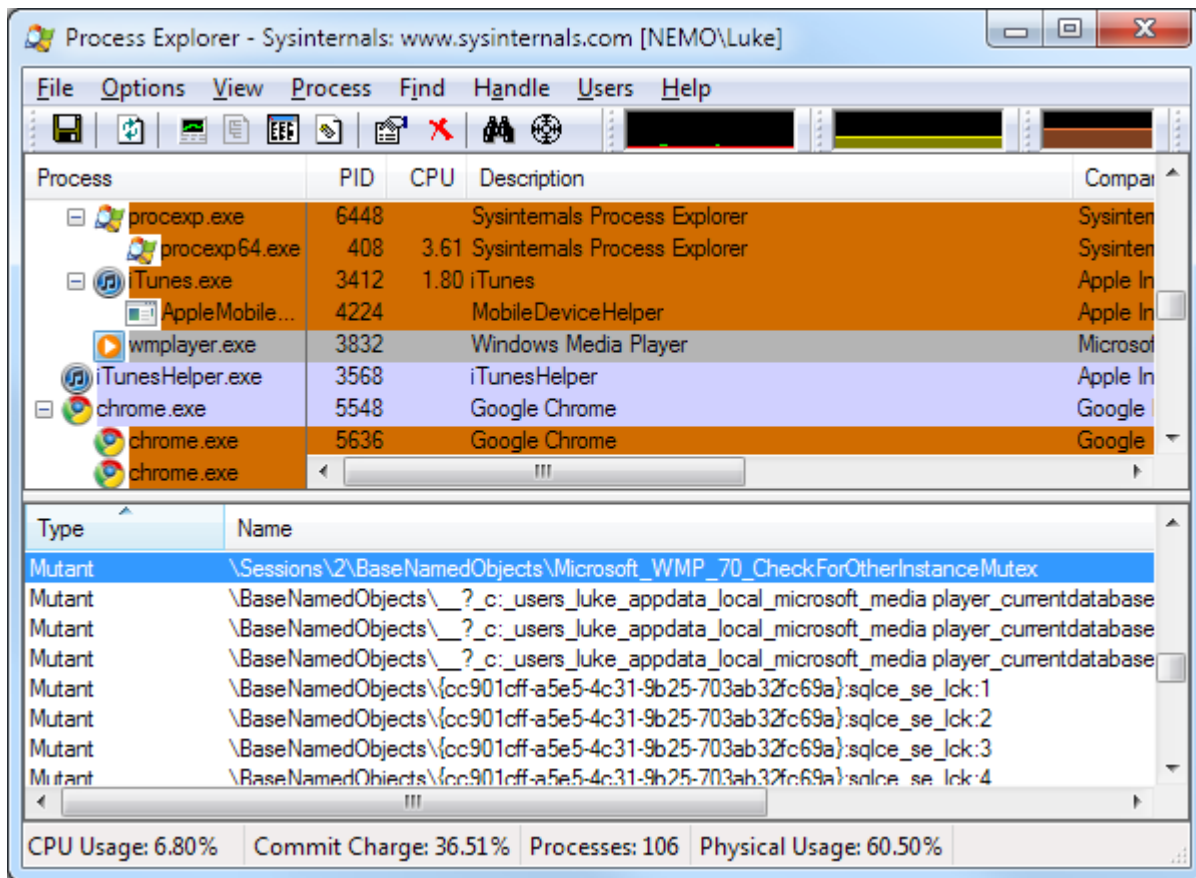


'Hacking' Windows Media Player

Have you ever noticed that it is possible to run only one instance of Windows Media Player? Attempting to run more than one instance will bring the current instance into focus. We are going to fix that.

Firstly, how does Windows Media Player know that there is already an instance of itself running? It does this by using a [Mutex](#). A [mutex](#) is a synchronization object, which is used to serialize access to a resource. We are able to see this in action by using [Sysinternals' Process Explorer](#).



After running Windows Media Player and Process Explorer, we can select the 'wmpplayer.exe' process and see various information about Media Player. What we are interested in, for the purposes of this article, is the Mutant highlighted here. This mutant is "Microsoft_WMP_70_CheckForOtherInstanceMutex" (Mutant is the internal name for a mutex). It is easy to determine this is the correct mutex (other than the obvious name), by right-clicking on the mutex and selecting 'Close Handle'. You will now be able to run another instance of Media Player. However, this instance, not already detecting the mutex, will create another mutex which will disallow anymore instances. It is possible via the same process to close that mutex, and run another instance, and so on.

We can also mess with Media Player by creating the mutex ourselves and Media Player will fail to run even a single instance, detecting the already existent mutex.

For example, to do this in C++:

```
#include <Windows.h>
#include <tchar.h>

int _tmain(int argc, _TCHAR* argv[])
{
    CreateMutex(NULL, TRUE, L"Microsoft_WMP_70_CheckForOtherInstanceMutex");

    system("PAUSE");
    return 0;
}
```

Whilst this console application is running it will have a hold of the mutex and Windows Media Player will not run, thinking that it is already running. You are able to check this by finding your application in Process Explorer.

So, now that we know how Windows Media Player goes about trying to keep itself a single instance application, we can stop this. Let's begin by opening wmplayer.exe in a disassembler, such as [Ida Pro](#). Retrieving debugging symbols from the Microsoft public symbol server makes navigating through the code a breeze. More information can be found at <http://support.microsoft.com/kb/311503>.

One can assume that the process of checking for other instances would occur in the start up routine. So jump to the PlayerEntry method. Having a brief look through this routine reveals some interesting code for example at offset 0x0C69 we see the call to the imported [CreateMutex](#) function, with the mutex name being '<Local\Microsoft_WMP_70_CheckForOtherInstance>'. Let's take a deeper look at things, following on the next page is the code flow, following with an explanation.



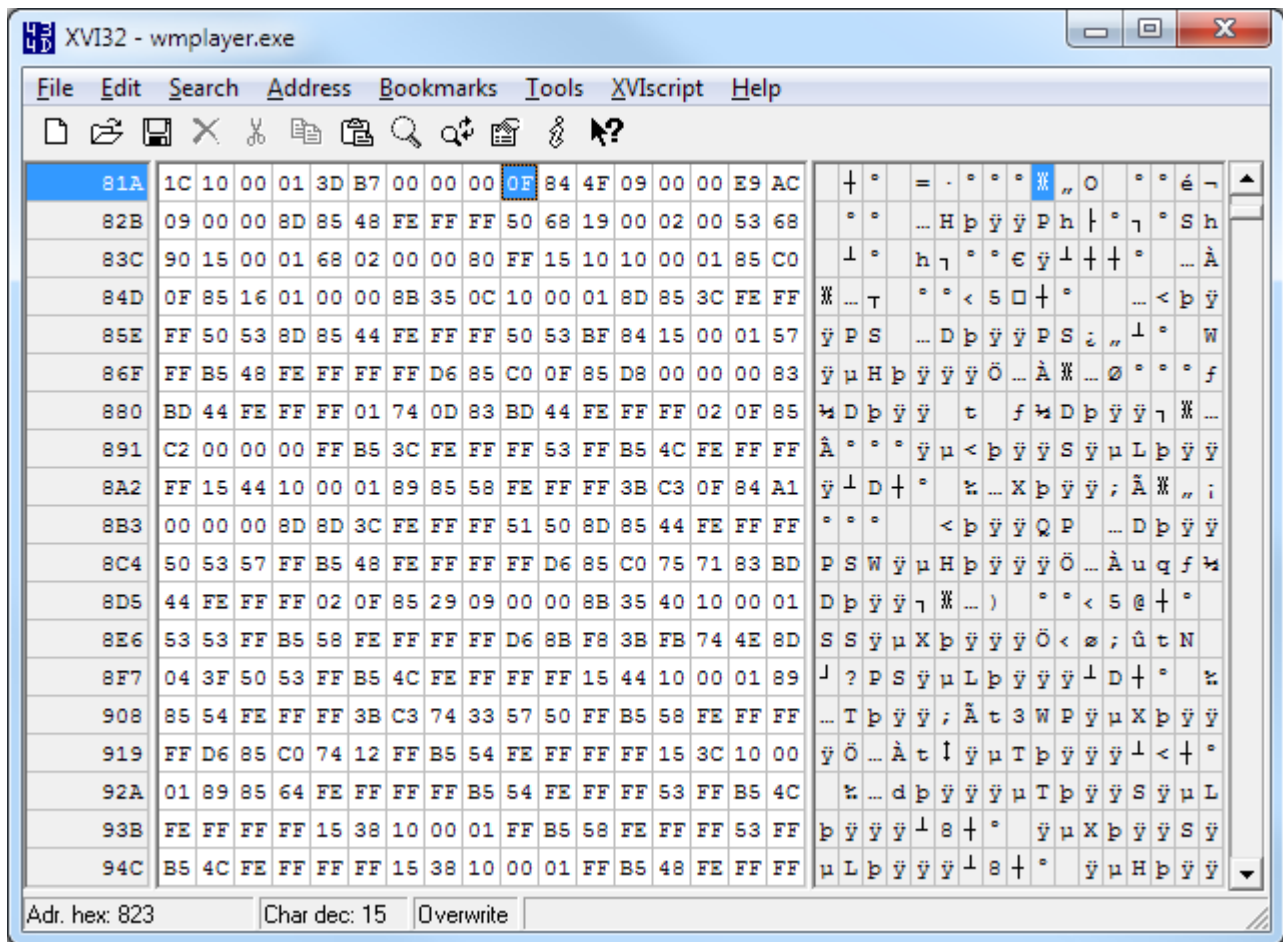
Starting with the start up routine (1), Windows Media Player calls the CreateMutex function to create the “Microsoft_WMP_70_CheckForOtherInstanceMutex” mutex. From here the code checks whether there was a returned value (2), and if so jumps to loc_CD1418 (3), otherwise, the code jumps to loc_CD1DCB to see if there was a permission issue with creating the mutex. The next section of code (3) [checks the last error code](#) and compares it with error code 183 (0xB7). Error code 183 is ERROR_ALREADY_EXISTS. So if there is a match, that means that the mutex already exists, and Media Player is already running. Upon finding a match, the code continues on to find the current instance

and bring that to the foreground to show to the user (4). If the error code does not match ERROR_ALREADY_EXISTS, then the code will continue through the start up procedures (5).

So, from what we have here, we can make remove one line of code and the single instance Media Player will be no more. Can you pick which line it is?

By removing the line in code block 3: 'jz loc_1001D78', the code will continue on to the normal start up procedure no matter what the outcome of the previous compare function, checking for the existence of the "Microsoft_WMP_70_CheckForOtherInstanceMutex" mutex.

OK, boot up your favourite hex editor, may I recommend [XVI32](#). The address we are looking for is 823, so let's jump to that:



To remove the code we replace it with [NOP \(No Operation\)](#), which for the [Intel family](#) is hex code 90.

