Lab 9

Due as a hard copy in class Tuesday, November 13th (e-mail is okay too if you can't turn it in by hard copy). Note that this is three weeks of lab so will count for 30 points. E-mail me with any questions you have and you can also e-mail me what you have before the deadline and I'll check it out and see if it looks right. You are expected to show up to lab this week, the next, and the week after, where Sheng and Bryan can help you with the x86 assembly and other questions you may have. You can work in groups of 2 or 3.

You should have received a file lab9assemblydump.txt. This is parts of the MyParty.A e-mail worm, which e-mailed itself as a www_myparty.com executable file (back in the DOS days executables were sometimes *.com instead of *.exe and Windows supports this). Basically, you execute the *.com file by clicking on it in your Outlook mail client and it e-mails itself to everyone in your address book.

The assignment is to reverse-engineer the things that it does with dates and times. From published reports: "If the computer date is not between January 25-29, 2002, or if the keyboard settings are set to Russian, the worm [exits]" (Google "W32.Myparty@mm" and a Symantec description is the first hit, then click on "technical details.")

In the file, the first column is the virtual address, the second is the machine code in hex, and the third is the disassembly which is x86 assembly, with most of the arguments (i.e. operands) being in hexadecimal. Google is a great tool for looking up what x86 instructions do, what Windows API function calls do, and converting to decimal (Google "0xABCD in decimal", for example).

You may want to research calender systems, it may save you a lot of painful reverse-engineering of what every single x86 instruction does. For example, certain Windows API calls return the date and time broken up into separate integers for day, month, year, hour, etc., while in Linux typically the data is a single integer counting seconds from 1970. 1970 is called the epoch in this case, other common epochs are 1900 and 1600 (because of the switch between the Julian and Gregorian calender in 1587, epochs before 1600 are tricky). There's a lot of code missing so you'll want to get a general sense of what's going on. (I was able to get this disassembly of the worm code by setting a breakpoint on calls to GetSystemTime and then letting the worm unpack itself.)

At one point the worm is getting it's own file attributes (date of last modification in particular) and doing something with that.

1. Give a general description of what each function does, e.g. "converts time from format X to format Y"

2. The published reports on MyParty.A said that the only constraint was that the current time was in a certain range. What other constraints are there related to date and time (there are at least two, one on the current date and time that is redundant i.e. always true if the published date check is true, and another on the file attibute)?

3. There are many conversions involving adjusting for time zone and converting between different formats for the date/time. Are they the most efficient way of doing things, or is there some redundant calculations that suggest the worm author was cutting and pasting code from other places? Explain.