CS4540 - Operating Systems
Assignment 1

Alex Dekau

September 7th, 2016

1 Microsoft Windows

1.1 Protection

Windows protects its users from malicious device drivers through a few different methods. One of the methods used by Microsoft Windows is driver signatures[1]. With Windows 10, only drivers that are signed from trusted vendors can be installed under normal settings. This can be bypassed however by disabling Driver Signature Enforcement[5].

1.2 How Windows Tells You How Safe and How Well a Driver Is Made

Windows tells you how well a Windows Driver matches a device using a specially designed ranking system. The rank is just an integer that is equal to or greater than zero, with the closer it is to zero meaning a better match[2]. The rank has three quantifiers: signature score, feature score, and the identifier score[2].

1.3 Driver Writing and Testing

To write a driver for Microsoft Windows, you can use the KMDF (Kernel Mode Driver Framework) or UMDF (User Mode Driver Framework)[6] through Visual Studio. To test the driver, build the source in Visual Studio and press the debug button. This will test it for the host machine. To test on a target machine, the driver must be compiled for the target operating system[6].

2 Mac OS X

2.1 Protection

Mac OS X 10.10 and higher uses nearly the same type of kernel mode driver safety that Microsoft Windows uses. Apple, instead, calls it System Integrity Protection (SIP)[7]. Essentially, Apple does not allow unsigned drivers to be installed on the machine. With SIP, the end user no longer has access to root, and can not install device drivers as super user[7].

2.2 How Mac OS X Tells You How Safe and How Well a Driver Is Made

I could not find any part of Mac OS X that specifically tells you if a Driver is safe or not, nor how well the driver communicates with the hardware. As for the safety aspect, I'd imagine it's probably not necessary to communicate to the user if the driver is safe to use or not since the driver's signature does that already. If the driver does not have a signature, it will not install without superuser permissions (requiring SIP to be disabled).

2.3 Driver Writing and Testing

Drivers are written for Mac OS X using XCode[8]. Apple has a standard development kit like Microsoft for driver development, called the I/O Kit. Depending on which APIs you use from the I/O kit, the driver being created will either exist in the user or the kernel space. Then the driver can start being written to communicate with the hardware using I/O Kit APIs such as the serial API for serial communication such as USB.

3 Linux

3.1 Protection

Linux's protection against installing malicious device drivers is pretty simple. It's that only the root user can load device drivers and other kernel code[9]. Linux as a whole, being a "power user's" operating system generally follows a "if you know the vendor and trust it, install it. If not, don't." philosophy. There is also inherent protection in the way that you have to go about installing a device driver. You have to specifically log in to the root user to do it, so malicious device drivers shouldn't be installed as a normal user by mistake or without your permission.

3.2 How Linux Tells You How Safe and How Well a Driver Is Made

There are ways to be able to know if a driver on Linux is safe or not, as well as know how well it works with the hardware. Since Linux is an open source operating system, many of the device drivers written for it are also open source. That means that the user, or other users, can check the source code of the driver and make sure nothing malicious is happening. The other way that you can know if a driver is safe or not is if it is in a trusted repository such as the distributor maintained ones for your Linux distribution. An example of that being the Ubuntu Software Center's maintained repositories (by Canonical).

3.3 Driver Writing and Testing

Linux kernel modules (device drivers) are written in C and compiled with the make command from the terminal (or other CLI)[10]. Just like other programs written in C, Linux drivers are written with includes to various standard library modules (such as linux/kernel.h and linux/module.h). Instead of using a main() function like in a normal C program, Linux drivers use init_module()[10]. You can debug the module by loading it into your own system or compiling it for a target system. You can also print things out only if the driver is in debug mode using *#ifdef DEBUG ... #endif*.

References

[1] https://technet.microsoft.com/en-us/library/cc776371(v=ws.10).aspx This link is a trustworthy source because it is from Microsoft, the creator of Microsoft Windows.

[2] https://msdn.microsoft.com/en-us/windows/hardware/drivers/install/how-setup-ranks-drivers–windows-vista-and-later- This link is a trustworthy source because it is from Microsoft, the creator of Microsoft Windows.

[3] http://www.techrepublic.com/blog/it-security/defend-against-kernel-malware/ This article was written by Tim Olzak and is trustworthy because Tim is a security researcher for the InfoSec Institute with over 30 years.

[4] https://msdn.microsoft.com/en-us/library/windows/hardware/ff554836(v=vs.85).aspx This link is a trustworthy source because it is from Microsoft, the creator of Microsoft Windows.

[5] http://www.howtogeek.com/167723/how-to-disable-driver-signature-verification-on-64-bit-windows-8.1-so-that-you-can-install-unsigned-drivers/ This article was written by Taylor Gibb, and is trustworthy simply because it is a how-to guide and not something that relies on certifications.

[6] https://msdn.microsoft.com/en-us/library/windows/hardware/ff554811(v=vs.85).aspx This link is a trustworthy source because it is from Microsoft, the creator of Microsoft Windows.

[7] http://www.infoworld.com/article/2988096/mac-os-x/sorry-unix-fans-os-x-el-capitan-kills-root.html This article was written by Paul Venezia. I believe it to be trustworthy because his information matches the Apple announcement found here: https://support.apple.com/en-us/HT204899

[8] https://developer.apple.com/library/mac/referencelibrary/GettingStarted/GS_HardwareDrivers/_index.html This link is a trustworthy source because it is from the Apple developer website.

[9] http://stackoverflow.com/questions/1565323/linux-kernel-modules-security-risk This link is a compilation of answers to a question about Linux kernel module security. The authors are various users offering an answer to the question.
I believe it is trustworthy because of the site's "reputation" system that other users can mark a user as reputable over time.

[10] http://www.tldp.org/LDP/lkmpg/2.6/html/lkmpg.html#AEN121 This page is written by Peter Salzman, and is section from the book "The Linux Kernel Module Programming Guide". I believe it's trustworthy as a published book.