Caleb Slater
ECE 4540
Assignment 1

Part One

Windows

The preferred method of driver writing on Windows is to write a Universal Windows Driver using the User-Mode Driver Framework (3). This is implemented using Visual Studio and the Windows Driver Kit (2). Once the driver is made it is ran through the Windows Driver Verifier. This program detects unauthorized function calls that can damage the OS and tests to verify there aren't any memory leaks (1). I got my information directly from Microsoft so my information is valid since I can't know more than the manufacturer.

References:

1. N. (n.d.). Driver Verifier. Retrieved September 10, 2017, from https://docs.microsoft.com/en-us/windows-hardware/drivers/devtest/driver-verifier
2. N. (n.d.). Getting started with Windows drivers. Retrieved September 10, 2017, from https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/
3. How to write your first USB client driver (KMDF). (n.d.). Retrieved September 10, 2017, from https://msdn.microsoft.com/en-us/library/windows/hardware/hh706187(v=vs.85).aspx

Linux

The preferred method of driver writing on Linux is to decide on using the user space or kernal space with access to the devices which will be created as virtual mounted files (1). Once the space is defined you can begin coding in C and include the OS's kernal (1). Then you can code the events, reading, writing, and loading actions of the driver and then compile with the use of a makefile (2). Different tools are used for Linux but a good one is the Linux Driver Verification (LDV) program (3). This program is an integrated platform that makes for a modular development approach in an open source environment (3). My reference are valid because they agree with one another and I found similar answers from the community on stack overflow.

References:

1. (n.d.). Retrieved September 11, 2017, from http://freesoftwaremagazine.com/articles/drivers_linux/
2. A. (2016, June 28). Device Drivers, Part 2: Writing Your First Linux Driver in the Classroom. Retrieved September 11, 2017, from http://opensourceforu.com/2010/12/writing-your-first-linux-driver/
3. Linux Driver Verification. (n.d.). Retrieved September 11, 2017, from http://linuxtesting.org/ldv

OSX

The preferred method of driver writing on OSX is to use C++ and communicate through a provider object using the IO Kit (1). OSX must also decide on whether the drivers must be written for the user or kernal space (2). Once the code is written you must update the kextd daemon to load the extension with a property list (2). Quality control is performed by checking permissions, checking for memory leaks, and are versions up to date (3). These references are all valid because they are from apple themselves.

References:

1. Kernel Extension Programming Topics. (2010, September 01). Retrieved September 11, 2017, from https://developer.apple.com/library/content/documentation/Darwin/Conceptual/KEXTConcept/KEXTConceptIOKit/iokit_tutorial.html
2. "Getting Started with Hardware and Drivers." *Introduction*, 28 May 2009, http://developer.apple.com/library/content/referencelibrary/GettingStarted/GS_HardwareDrivers/_index.html. Accessed 11 Sept. 2017.
3. IOKit Device Driver Design Guidelines. (2009, August 14). Retrieved September 11, 2017, from https://developer.apple.com/library/content/documentation/DeviceDrivers/Conceptual/WritingDeviceDriver/DeployingDrivers/DeployingDrivers.html

Part Two

Windows

Windows uses driver signatures to verify the driver is valid (1). The File Signature Verification tool can scan all drivers to make sure they are approved by windows (1). They use the RSA cryptography to allow for remote verifying using a message from the signature (2). My sources are valid because I have had some experience installing drivers and both sources agree.

References:

1. (n.d.). Retrieved September 11, 2017, from http://www.davidegrayson.com/signing/
2. Brinkmann, M. (2015, April 11). How to verify that system drivers are digitally signed - gHacks Tech News. Retrieved September 11, 2017, from https://www.ghacks.net/2015/04/11/how-to-verify-that-system-drivers-are-digitally-signed/

Linux

Linux checks driver signatures by having the creater sign them upon release (1). The keys are encrypted with RSA and SHA encryption methods (1). This public key encryption is similar to how windows validates signatures, although these signatures are commonly stored in elf files (2). These sources are valid because they are similar to the methods used by windows and both sources agree with each other.

References:

1. Kernel module signing facility. (n.d.). Retrieved September 11, 2017, from https://www.kernel.org/doc/html/v4.10/admin-guide/module-signing.html
2. Signed Kernel Modules. (n.d.). Retrieved September 11, 2017, from http://www.linuxjournal.com/article/7130

OSX

OSX checks driver signatures by validating SHA and MD5 fingerprints (1). Again a similar method of generating a secure hash is made for the driver which can then report if there is any tampering (2). They can use Gatekeeper and manually check the fingerprint to verify file integrety (2). My sources are valid because one is from apple and the other is from a known reputable source.

References:

1. How to verify the authenticity of manually downloaded Apple software updates. (2017, August 26). Retrieved September 11, 2017, from https://support.apple.com/en-us/HT202369
2. How to Show & Verify Code Signatures for Apps in Mac OS X. (2016, March 14). Retrieved September 11, 2017, from http://osxdaily.com/2016/03/14/verify-code-sign-apps-mac-os-x/