Project Proposal:

Safexec

John Hardy 1/22/19

# Verbal Description

Safexec is a code signing application and Linux Kernel modification designed to demonstrate the importance of code signing. Many users assume that the code they are running on their machines has not been modified or tampered with during run time. Unfortunately, this is not the case. All too often, users download questionable code from the internet and are not bothered to validate that the code is safe. This is largely due to the fact that manually checking the code (i.e. with a disassembler or hex editor) is far too difficult for most users to do for themselves.

## Explanation

What is the solution? Code signing! A developer “signs their code” by running a hashing function on the binary executable code after compilation and then takes the string of bytes from the function (called a “digest”) and includes it inside the executable file or posts it somewhere else. The user then runs the same hash function before execution and verifies that the two digests indeed match each other. This verifies that the code has not been tampered with without going through the trouble of checking the code itself.

## Modern Applications

Many code signing applications are already implemented in many areas: Operating systems like Windows, Mac OS X, and some versions of Linux. Developers making apps for the iTunes store must also sign their code before uploading it. Some websites providing content to users often provide an MD5 digest with their files so that users can verify on their own that the files have not been tampered with during transmission.

## Features

The code-signing portion of the application will provide a Graphical User Interface for the user to utilize quickly. The user needs only to provide the location of the file on their file systems and wait for the application to finish generating the digest. The digest will then automatically be included into the header of the executable file. The code signing process is now finished.

The end-user need only run the operating system provided with Safexec. If they try to run an executable file, the OS will search the header for a digest. If it finds one it will run the hashing function on the binary code and verify that the two digests indeed match. If they do, the OS will continue running the executable as normal; otherwise, it will inform the user that the hashing failed and terminate execution.

## Requirements

Any computer that is capable of running Ubuntu at a reasonable speed will do. Obviously, you must have the Safexec Operating System running on the computer for the code-signing to work. No other hardware or software will be needed.

# Justification

1. This project will implement material that I have used in several classes including: Operating Systems, Programming Languages, Introduction to Cybersecurity and Introduction to Cryptography
2. I will be extending what I have learned by utilizing the same techniques that I was required to learn about in Operating Systems (modifying the Linux Kernel) and Programming Languages (utilize the Python programming language).
3. Difficulty can be scaled forward or backward as needed. My final product does not need to be a Linux Kernel modification and does not need to have a graphical user interface.
4. One semester is plenty of time for this project. The scale of the application does not require it to continue to be developed after 402 has finished.
5. I am familiar with all the tools presented. I have used Ubuntu before and have made modifications to it. I have written code in Python before and have imported modules as well. I have used a hex editor on my own to look a binary code.
6. I am fascinated with cybersecurity and I believe that this project will be very relevant to that field as well as my future career. I cannot speak for my fellow classmates however…