# **Requirements Specification Document**

# **Safexec**

# **Code Signature Application**

# 

# 5.1 Introduction

Safexec is a useful code signature application designed to thwart attempts to modify executable code for malicious purposes. Anyone who uses Safexec on their executable code will leave a stamp with a special signature on it that can be verified at run-time to ensure that the code has not been modified.

# 5.2 CSCI Component Breakdown

Safexec is composed of the following Computer System Configurations:

5.2.1 Code Stamping CSC -- Many parts to the code stamping algorithm

5.2.1.1 Python code stamping algorithm -- Manages file reading and manipulation

5.2.1.2 SHA3-512 Hash function -- Hashes executable binary stream into a 512 bit signature

5.2.2 Input Handling CSC -- Input handling process

5.2.2.1 Windows Mouse and Key Input Drivers -- Reports mouse and key input

5.2.2.2 Python code stamping algorithm -- handles input given by the user

5.2.3 Linux Runtime Code Signature Verification CSC -- Verifies stamped code

5.2.3.1 Linux Kernel modification -- Detects stamped executable code

5.2.3.2 Python code hashing algorithm -- hashes stamped code to ensure it is safe to run

# 5.3 Functional Requirements

Safexec’s main functional goal should be to verify whether or not code is safe to run. The user is mostly removed from the functioning of the application unless the code is unsafe to execute. The programmer will need to have a decent understanding of using the Linux bash shell in order to utilize the code stamping algorithm.

5.3.1 The code stamping algorithm should access a provided file path with no problems.

5.3.2 The code stamping algorithm shall read and write to binary executable files.

5.3.3 The code stamping algorithm shall hash the binary executable code.

5.3.4 The code stamping algorithm shall write the stamp to the header of the file.

5.3.5 The input handling pipeline shall read input from the user in the bash shell.

5.3.6 The input handling pipeline shall interpret a correct file path provided by the user.

5.3.7 The Linux kernel modification shall detect modified file headers.

5.3.8 The Linux kernel modification should run unmodified code with no problems.

5.3.9 The Linux kernel modification shall run the Python hashing algorithm for modified code.

5.3.10 The Linux kernel modification should run code that is deemed safe by the python hashing algorithm.

5.3.11 The Linux kernel modification should not run code that is deemed unsafe by the python hashing algorithm.

5.3.12 The python hashing algorithm shall hash modified executable binary code.

5.3.13 The python hashing algorithm shall verify that the stamped signature does match the hashed executable code.

5.3.14 The python hashing algorithm shall verify that the stamped signature does not match the hashed executable code.

5.3.15 The python hashing algorithm should report back whether the code is safe to run or not.

# 5.4 Performance Requirements

Safexec’s performance goal is to remain as removed from the user’s experience as possible. The user should be able to run stamped code with little to no wait time and should not have to navigate annoying menus unless the code is unsafe to run. The programmer’s experience will be less user-friendly and is more oriented towards basic functionality.

5.4.1 The code stamping algorithm should sign files in a time proportional to their size.

5.4.2 The input handling pipeline should collect all user input within 1 second.

5.4.3 The Linux kernel modification should automatically detect stamped executable files.

5.4.4 The Linux kernel modification should notify the user if code is unsafe to run.

5.4.5 The Linux kernel modification should run safe or unsigned code as normal without interrupting the user’s workflow.

5.4.6 The python hashing algorithm should hash files in a time proportional to their size.

# 5.5 Project Environment Requirements

## 5.5.1 Development Environment Requirements

Following are the hardware requirements for development of Safexec:

|  |  |
| --- | --- |
| Category | Requirement |
| Processor | Intel Core i3, 2.30 GHz or Better |
| Hard Drive Space | 2 Gb |
| RAM | 1 Gb |
| Display | 800x600 or better |

Following are the software requirements for developing Safexec:

|  |  |
| --- | --- |
| Category | Requirement |
| Operating System | Safexec Linux kernel |
| Input Handling | Keyboard and Mouse Drivers |
| Source Code Management/Design | Atom |
| Compiler/Execution | Python 3 |

It is required that a modifiable version of Linux be used to develop Safexec in order create the Linux Kernel modification needed to detect signed code.

## 5.5.2 Execution Environment Requirements

The hardware requirements for executing Safexec are equivalent to those for developing Safexec.

Following are the software requirements for executing Safexec:

|  |  |
| --- | --- |
| Category | Requirement |
| Operating System | Safexec Linux kernel |
| Input Handling | Keyboard and Mouse Drivers |
| Compiler/Execution | Python 3 |

The modified Linux kernel must be used in order to get the entire Safexec system to work. It will still be possible to manually feed signed files into the Python code on other operating systems, but Safexec is designed to be run seamlessly by the user.