# **Software Development Plan**

# **Safexec**

# **Code Signature Application**

# 

# 4.0 Software Development Plan

## 4.1 Plan Introduction

This Software Development Plan provides the details of the planned development for the Safexec Software CSCI which provides an application to ensure that an ELF (Linux executable file format) file has not been maliciously modified.

Safexec is a code signature application used to ensure that code sent over the internet has not been modified for malicious purposes. Development of Safexec will begin with basic hashing which will lead into hashing entire files then modification of ELF files into ELFS files (secure ELF) and finally tweaking the Linux kernel to automatically check ELFS files.

|  |  |
| --- | --- |
| Sub-task Completion Dates | |
| Basic hashing is working | Week 4 |
| Basic Unit test suite is working | Week 7 |
| ELF signing and ELFS checking is working | Week 9 |
| Linux kernel modifications are working | Week 12 |
| Safexec is ready for release | Week 13 |

### 4.1.1 Project Deliverables

|  |  |
| --- | --- |
| Project Deliverables | |
| Project Proposal Presentation / Document   * Present PowerPoint and document outlining Safexec to class | Week 02 |
| Basic hashing working   * Unit tests and code for hashing strings and files (of all types) is working | Week 04 |
| Requirements Specification Document (initial)   * List out what will be needed for the project | Week 05 |
| Basic unit test suite is working   * Unit tests for all parts of Safexec have been created | Week 07 |
| Software Development Plan Document (initial)   * The formal plan for how Safexec will be developed * Includes an initial development schedule   ELF signing and ELFS checking are working   * Safexec can manually sign ELF files and can manually check ELFS files as well | Week 09 |
| Software Design Description Document (Updated)   * Implement changes made for the initial SDP | Week 11 |
| Linux kernel modifications are working   * Linux kernel now recognizes ELFS files and uses Safexec to check them before running them | Week 12 |
| Requirements Specification Document (resubmit)   * Implement changes made to initial SRS and add more detail about resources needed   Safexec is ready for release   * All unit tests have been completed and Safexec is ready to be released to the general public for use | Week 13 |
| Preliminary Demonstration Presentations   * Present completed version of Safexec to the class | Week 13-15 |
| FINAL Product Delivery (Final Report and Code)   * Final report on how development of Safexec turned out along with all the code   FINAL Project Presentation   * Final presentation of Safexec | Week 16 |

## 4.2 Project Resources

### 4.2.1 Hardware Resources

Note that all hardware delineated below is the hardware used for development and execution. These do not count as a “minimum systems requirement” (see requirements specification) but rather as an outline of appropriate system resources. Comparable hardware resources should be able to run Safexec as long as enough hard drive space is allocated appropriately.

|  |  |  |
| --- | --- | --- |
| Resource | Development | Execution |
| Intel Core i3, 2.3 GHz  8 GB ram  500 GB Hard Drive | X | X |

### 4.2.2 Software Resources

For development, a large amount of software was used to implement all of the features. Execution, however will only require small modifications to the Linux kernel and python 3 to be installed.

|  |  |  |
| --- | --- | --- |
| Resource | Development | Execution |
| Windows 10 | X |  |
| Ubuntu 18.4.1 | X | X |
| Oracle VM VirtualBox | X | (optional) |
| Atom | X |  |
| Python 3 (includes standard libraries) | X | X |
| HxD | X |  |
| Git | X |  |
| Safexec Linux kernel modifications | X | X |

## 4.3 Project Organization

* Unit test suite
  + This is a module used during the development of Safexec to ensure that all code runs correctly and as expected. The unit test suite will be updated and added to regularly throughout development, but a basic one will be completed by week 7.
  + The basic unit test suite will have at least one test for each function of Safexec including: hashing small amounts of data, hashing files, ensuring that errors are thrown correctly, testing ELF signing, testing ELFS checking, and testing manipulation of files.
* Data hashing module
  + The heart of the Safexec application. It uses SHA version 3 to hash a set of data into a 64 byte signature. Also comes with smaller methods for hashing strings or arbitrary chunks of data.
* File management module
  + This module handles file manipulation and controls the flow of the program. It attempts to open files provided by the user and throws custom errors otherwise. It also attaches signatures provided by the data hasher to the end of ELF files, making them into ELFS files. ELFS files are then converted back to ELF files by removing the signature.
* Linux kernel modifications
  + The Linux kernel modifications are a series of scripts to be added to the Ubuntu 18.4.1 kernel to handle ELFS files. The kernel will run the above python scripts when it detects that the user is trying to run an ELFS file and will not run the files if they are deemed unsafe.

## 4.4 Project Schedule

### 4.4.1 GANTT Chart

(See excel spreadsheet in repository)

### 4.4.2 Task / Resource Table

|  |  |  |
| --- | --- | --- |
| Task | Persons Assigned | Equipment |
| Basic Hashing | John Hardy | Computer, Win 10, Git, Python 3, Atom |
| Basic Unit Tests | John Hardy | Computer, Win 10, Git, Python 3, Atom |
| ELF signing | John Hardy | Computer, Win 10, Git, Python 3, Atom, VirtualBox, Ubuntu, HxD |
| ELFS checking | John Hardy | Computer, Win 10, Git, Python 3, Atom, VirtualBox, Ubuntu, HxD |
| Linux Kernel mods | John Hardy | Computer, Win 10, Git, Python 3, Atom, VirtualBox, Ubuntu |